AAF01-A

ANALOG OUTPUT DIAG
CZAAFA0

AH-T981A-MC
1 OF 2 JUL 1985

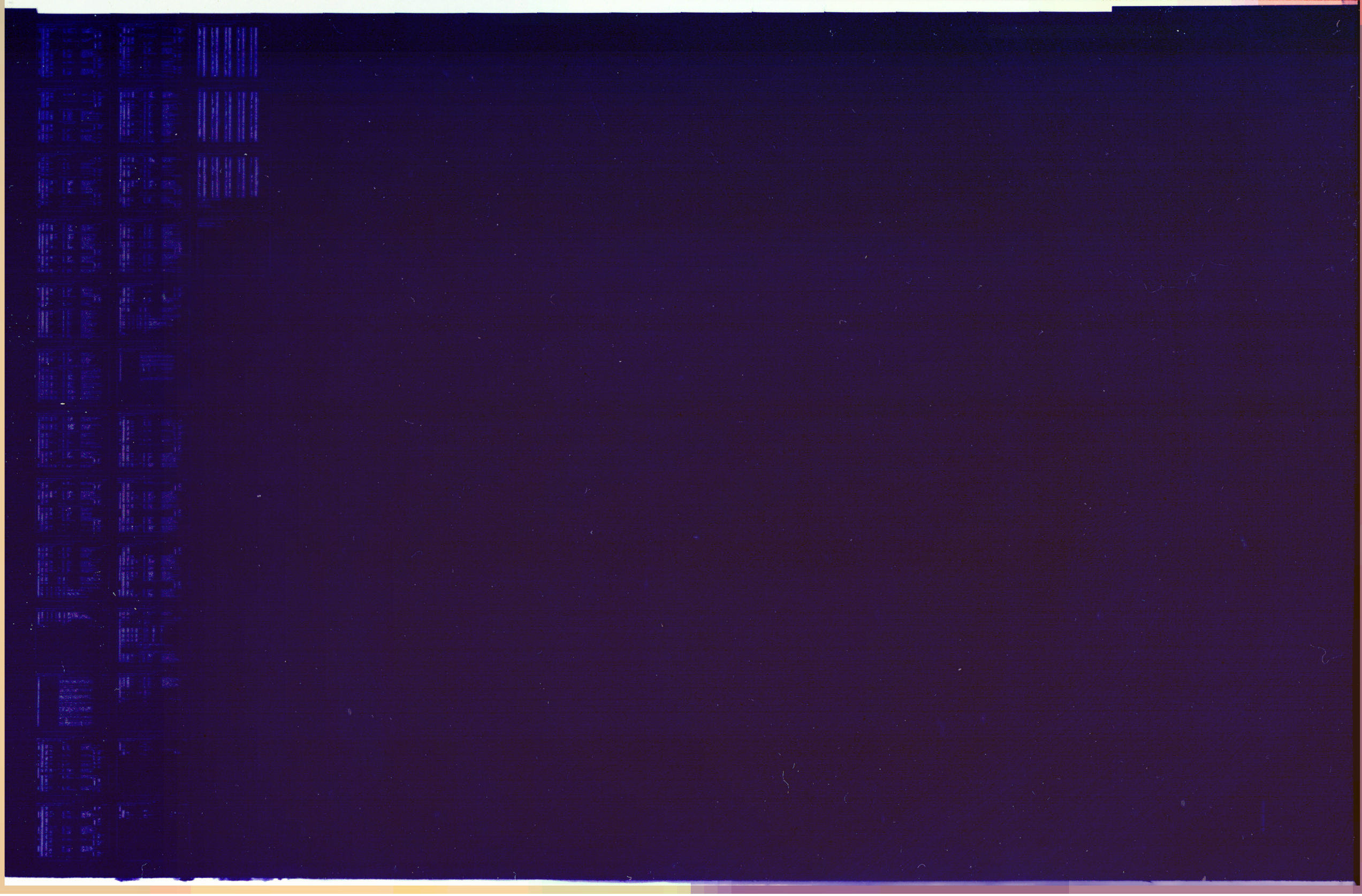
digital
MADE IN USA

AAF01-A

ANALOG OUTPUT DIAG
CZAAFA0

AH-T981A-MC
2 OF 2 JUL 1985

digital
MADE IN USA



.REM &

IDENTIFICATION

Product Code: AC-T980A-MC

Product Name: CZAAFAO ANALOGUE OUTPUT DIAGNOSTIC

Product Date: January 1985

Maintainer: CSS Munich

Author: Dave Hunter

TABLE OF CONTENTS

1.0 General information

1.1 Program abstract

The CZAAFA0 diagnostic provides a series of tests to verify the integrity and functionality of the AAF01-A Analogue Output module. This diagnostic can be used by field service for functional testing, by the engineer for design tests, and by manufacturing for checkout and repair. Special test procedures for design and manufacturing checkout are described in section 7.0.

The following special features are implemented.

The "PRINT" command can be used to obtain a list of test titles, or a printout of the error statistics accumulated by the diagnostic. For more information, see section 4.0.

If the evaluate flag "EVL" is set, any unit on which more than 5 errors are detected following a "START" command is dropped from testing.

The program supports up to 16 units, all selected tests being run on one unit before proceding to the next unit.

This diagnostic has been written for use with the diagnostic runtime services software (supervisor) running under XXDP+. These services provide the interface to the operator and to the software environment.

For a complete description of the runtime services, refer to the XXDP+ user's manual. A brief description is given in section 2 of this document.

1.2 System requirements

a. PDP-11 processor with a minimum of 28k of memory.

b. Console terminal with interface address 777560.

c. XXDP+ load device (RX,RK,RL etc.)

d. AAF01-A module(s) to be checked, each with a DRU11-C or DRQ11-C for Unibus or Q-bus respectively.

e. Field checkout : Digital voltmeter accurate to +/- 0.01% and test connector 2G-M00FA

   Manufacturing : Digital voltmeter accurate to +/- 0.01%, test connector 2G-M00FA, ADF01/DRX11-C, loopback cable BC05L-xx, and oscilloscope.

1.3 Running the diagnostic on a FALCON

To run the diagnostic on a FALCON based system, a bootstrap program is needed in addition to the above requirements. This could be in the FALCON MACRO ODT rom (KXT11-A2), or on an MXV-11 board.

NOTE:

A) Once the XXDP+ media is booted, the console "BREAK" key should not be pressed as it may cause error messages to be printed.

B) I/O Page addresses from 160000 to 173776 are used by the KXT11-A2 ODT prom, so the first DRX11-C address must be 174000 or higher.

C) FALCON does not support vectors over 374.

1.4 Related documents and standards

XXDP+ User manual (CHQUS)
AAF01-A Option description YG-C03ZC-00

1.5 Diagnostic hierarchy prerequisites

Before running this diagnostic, the appropriate PDP-11 CPU, memory and peripheral standard diagnostics should be run to verify correct operation of the system. The DRU11-C OR DRQ11-C should also be tested using diagnostic YG-Z01TB-00.

1.6 Execution time

Execution times vary with the CPU type. The following times are typical for one pass with one module on a PDP-11/23 using defaults for all input parameters :

Default tests (excluding calibration) :	45 seconds
Loopback test using ADF01 :	10 seconds
Loopback test using IEX11-A :	3 minutes, 35 seconds

2.0 Operating instructions

This section contains a brief description of the runtime services. for detailed information, refer to the XXDP+ user's manual (CHQUS).

2.1 Commands

There are eleven legal commands for the diagnostic runtime services (supervisor). This section lists the commands and gives a very brief description of them. The XXDP+ user's manual has more details.

COMMAND	EFFECT
START	Start the diagnostic from an initial state
RESTART	Start the diagnostic without initializing
CONTINUE	Continue at test that was interrupted (after tc)
PROCEED	Continue from an error halt
EXIT	Return to XXDP+ monitor (XXDP+ operation only!)
ADD	Activate a unit for testing (all units are considered to be active at start time
DROP	Deactivate a unit
PRINT	Print test titles or error statistics
DISPLAY	Type a list of all device information
FLAGS	Type the state of all flags (see section 2.3)
ZFLAGS	Clear all flags (see section 2.3)

A command can be recognized by the first three characters. So you may, for example, type "STA" instead of "START".

2.2 Switches

There are several switches which are used to modify supervisor operation. These switches are appended to the legal commands. All of the legal switches are tabulated below with a brief description of each. In the descriptions below, a decimal number is designated by "ddddd".

SWITCH	EFFECT
/TESTS:LIST	Execute only those tests specified in the list. list is a string of test numbers, for example - /tests:1:5:7-10. this list will cause tests 1,5,7,8,9,10 to be run. all other tests will not be run.
/PASS:ddddd	Execute ddddd passes (ddddd = 1 to 64000)
/FLAGS:FLGS	Set specified flags. flags are described in section 2.3.
/EOP:ddddd	Report end of pass message after every ddddd passes only. (ddddd = 1 to 64000)
/UNITS:LIST	TEST/ADD/DROP only those units specified in the list. list example - /UNITS:0:5:10-12 use units 0,5,10,11,12 (unit numbers = 0-63)

Example of switch usage:

START/TESTS:1-5/PASS:1000/EOP:100

The effect of this command will be: 1) tests 1 through 5 will be executed, 2) all units will tested 1000 times and 3) the end of pass messages will be printed after each 100 passes only. A switch can be recognized by the first three characters. You may, for example, type "/TES:1-5" instead of "/TESTS:1-5".

Below is a table that specifies which switches can be used by each command.

	TESTS	PASS	FLAGS	EOP	UNITS
START	X	X	X	X	X
RESTART	X	X	X	X	X
CONTINUE		X	X	X	
PROCEED			X		
DROP					X
ADD					X
PRINT					
DISPLAY					X
FLAGS					
ZFLAGS					
EXIT					

2.3 Flags

Flags are used to set up certain operational parameters such as looping on error. All flags are cleared at startup and remain cleared until explicitly set using the flags switch. Flags are also cleared after a start command unless set using the flag switch. The ZFLAGS command may also be used to clear all flags.

With the exception of the start and zflags commands, no commands affect the state of the flags; they remain set or cleared as specified by the last flag switch.

FLAG	EFFECT
HOE	Halt on error - control is returned to runtime services command mode
LOE	Loop on error
IER*	Inhibit all error reports
IBE*	Inhibit all error reports except first level (first level contains error type, number, pc, test and unit)
IXE*	Inhibit extended error reports (those called by PRINTX macros)
PRI	Direct messages to line printer
BOE	"Bell" on error
PNT	Print test number as test executes
UAM	Unattended mode (no manual intervention)
ISR	Inhibit statistical reports (not applicable)

IDR — Inhibit program dropping of units (not required since units are only dropped if EVL is used)
ADR — Execute autodrop code
LOT — Loop on test
EVL — Execute evaluation ie. drop unit if more than 5 errors occur after a START command.

*error messages are described in section 3.1

See the XXDP+ user's manual for more details on flags. You may specify more than one flag with the flag switch. For example, to cause the program to loop on error, inhibit error reports and type a "bell" on error, you may use the following string:

/FLAGS:LOE:IER:BOE

2.4 Hardware questions

When the diagnostic is started, the runtime services will prompt the user for hardware information by typing "CHANGE HW (L) ?"

To run the diagnostic with more than one module or with a module at an address other than the default, you must answer "Y" to the "CHANGE HARDWARE" question. The runtime services will then ask for the number of units (in decimal). To keep down memory requirements, the maximum number of units supported is 16. You will then be asked the following questions for each unit:

DRX11-C ADDRESS (O) 164200 ?

In reply, you should enter an address in octal in the range 160000 to 177770.

DRX11-C VECTOR ADDRESS (O) 270 ?

The following illustrates the response to the hardware questions. In this example, the user response is underlined :-

CHANGE HARDWARE (L) ? Y <cr>

#UNITS (D) ? 2 <cr>

UNIT 0

DRX11-C ADDRESS (O) 164200 ? 164400 <cr>

DRX11-C VECTOR ADDRESS (O) 270 ? 310 <cr>

UNIT 1

DRX11-C ADDRESS (O) 164400 ? 164600 <cr>

DRX11-C VECTOR ADDRESS (O) 310 ? 320 <cr>

In this example, two AAF01-A's using DRX11-C's at addresses 164400 and 164600 will be tested.

Notice that the default value for the address and vector changes when a non-default response is given. This is true for all of the hardware questions, so be careful when specifying multiple units!

2.5 Software questions

After you have answered the hardware questions or after a restart or continue command, the runtime services will ask for software parameters. These parameters govern the diagnostic operating modes. you will be prompted by "CHANGE SW (L) ?". The normal response is to type "N".

Typing "Y" causes the following questions to be asked :

FIRST CHANNEL (D) 0 ?
LAST CHANNEL (D) 15 ?

These allow the user to concentrate testing on a single channel or group of channels in tests 24, 25 and 26. By default, 16 channels will be tested. The diagnostic will however accept any value between 0 and 63 for either question.

CONVERSION TIME (100 NANOSEC STEPS) (D) 200 ?

This specifies the time between conversions in steps of 100 nanoseconds. Although the minimum working value is 25, the diagnostic will accept any value between 1 and 4095. This value is used in tests 10 to 13, and 15 to 26.

NOTE : Small conversion times (< 50 steps) may result in error printouts showing the "data buffer empty" to be set in the ACS. These probably do not indicate a genuine error, rather than the DRX11-c cannot supply data to the AAF01-A quickly enough.

TOLERANCE IN BITS (D) 7 ?

This value is used as the tolerance in the linearity and ADF01 loopback tests. In the former, the tolerance is used as the value by which channels 0 and 15 differ when a data comparison is made. In the loopback test, values which deviate from the expected by more than the tolerance result in an error message.

QUICK VERIFY MODE (L) N ?

If the answer to this question is "Y", only one iteration of each test will be performed. Otherwise, some testing is done more than once. Repeatedly testing a piece of logic in this way often detects faults which a single test would not. Therefore, to fully test the hardware, the answer to this question should be "N".

```
REGISTER DUMP AFTER EVERY ERROR (L) N ?
```

Answering this question with "Y" causes a dump of the DRX11-C and AAF01-A register contents each time an error is detected. This feature is primarily for design testing and is normally disabled. The format of the dump is shown in section 3.1.

The following illustrates the response to the software questions. The user response is underlined :

```
CHANGE SOFTWARE (L) ? Y <cr>
                      ------
FIRST CHANNEL (D) 0 ? 2 <cr>
                      ------
LAST CHANNEL (D) 15 ? 2 <cr>
                      ------
CONVERSION TIME (100 NANOSEC STEPS) (D) 200 ? 25 <cr>
                                              -------
TOLERANCE IN BITS (D) 2 ? 16 <cr>
                         -------
QUICK VERIFY MODE (L) N ? <cr>
                         ----
REGISTER DUMP AFTER EVERY ERROR (L) N ? Y <cr>
                                        ------
```

In this example, channel 2 of the AAF01-A is being tested at the highest working conversion rate. Since accuracy is reduced at this rate, the tolerance has been increased to 16 LSB's. If an error occurs, a dump of the AAF01-A and DRX11-C register contents will be printed.

2.6 Extended p-table dialogue

When you answer the hardware questions, you are building entries in a table that describes the devices under test. The simplest way to build this table is to answer all questions for each unit to be tested. If you are testing several identical devices, this becomes tedious since most of the answers are the same for each unit.

To illustrate a more efficient method, suppose you are testing three AAF01-A modules. You could answer the hardware questions for each of the three units as shown in section 2.4. However, the procedure can be made more efficient.

The runtime services can take multiple unit specifications. Let's build an example table using the multiple specification feature:

```
CHANGE HARDWARE (L) ? Y <cr>
                      ------
#UNITS (D) ? 3 <cr>
             ------
UNIT 0
```

```
DRX11-C ADDRESS (0) 164200 ? 164200,164210,164220 <cr>
                             -------------------------
DRX11-C VECTOR ADDRESS (0) 270 ? 270,274,300 <cr>
                                 ---------------
```

As you can see in the above dialogue, the runtime services will build as many entries as it can with the information given in any one pass through the questions. In this example, the 3 AAF01-A modules at addresses 164200, 164210 and 164220 are set up with vectors 270, 274 and 300.

2.7 Clock questions

If there is no line time clock on the system, the user is asked to type 2 characters 6 seconds apart on the console. This should be done as accurately as possible since the interval is used by the diagnostic to calculate values for device timeouts.

2.8 Quick start-up procedure (XXDP+)

To start-up this program:

1. Boot XXDP+
2. Give the date and answer XXDP + questions
3. Type "R ZAAF??". (Normally the revision and patch level are typed instead of the question marks. The form shown here causes the latest version to be run.)
4. Type "START"
5. For a module using the default DRX11-C address of 764200 and vector 270, answer the "CHANGE HW" question with "N". To test more than one module or one at a different address, type "Y" and answer all of the hardware questions.
6. Answer the "CHANGE SW" question with "N"

When you follow this procedure you will be using only the defaults for the software parameters and no flags will be set. Flags and defaults are described in sections 2.3 to 2.5.

3.0 Error information

3.1 Types of error messages

There are three levels of error messages that may be issued by a diagnostic : general, basic and extended. General error messages are always printed unless the "IER" flag is set (section 2.3). The general error message is of the form :

```
NAME  TYPE  NUMBER  ON UNIT NUMBER  TST NUMBER PC:XXXXXX
ERROR MESSAGE
```

where NAME = diagnostic name
TYPE = error type (SYS FATAL, DEV FATAL, HARD or SOFT)
NUMBER = error number
UNIT NUMBER = 0 - N (N is last unit in ptable)
TST NUMBER = test and subtest where error occurred
PC:XXXXXX = address of error message call

General error messages for this diagnostic are listed together with the test descriptions in section 6.0.

Basic error messages are messages that contain some additional information about the error. These are always printed unless the "IER" or "IBR" flags are set (section 2.3). These messages are printed after the associated general message. In this diagnostic, basic error messages contain information such as register contents or good/bad data.

Extended error messages contain supplementary error information. These are always printed unless the "IER", "IBR" or "IXR" flags are set (section 2.3). These messages are printed after the associated general error message and any associated basic error messages. The only extended error message in this diagnostic is in test 26, requesting the user to recalibrate the module if the loopback test fails.

If the "REGISTER DUMP" question in the software questions is answered affirmatively, a register dump will be printed after each error message as in the following example :

```
AAF01-A  ACS: xxxxxx, CTA: xxxxxx, CWR: xxxxxx
DRX11-C  SCR: xxxxxx, COR: xxxxxx, ADR: xxxxxx, DBR: xxxxxx
         BAR1:xxxxxx, WCR1:xxxxxx, BAR2:xxxxxx, WCR2:xxxxxx
         WCO: xxxxxx, ACO: xxxxxx
```

4.0 Performance and progress reports

At the end of each pass, the pass count is given along with the total number of errors reported since the diagnostic was started. The "EOP" switch can be used to control how often the end of pass message is printed. Section 2.2 describes switches.

4.1 Print command utilization

The "PRINT" command can be used to find out how many errors have occurred on each unit since the diagnostic was started.

In addition, the command can be used to to print a list of test titles. The following examples show how the print command can be used. User input is underlined :

```
PRINT <cr>
----------
TYPE S, T OR HELP (S) H ? <cr>
                          ----
THE FOLLOWING COMMANDS ARE ACCEPTED :-

S - PRINT STATISTICS TABLE
T - PRINT TEST TITLES
TYPE S, T OR HELP (S) H ?
```

If you type "H", "HELP" or any character other then "S" or "T", the routine prints the above help message listing the acceptable commands.

```
PRINT <cr>
----------
TYPE S, T OR HELP (S) H ? T <cr>
                          ------

TEST TITLES.

  1  DRX11-C Register NXM Test
  2  Reset Test
  3  ACS Read/write Test
  4  CTA Read/write Test
  5  Control Table Data Test
  6  Control Table Address Test
  7  COUT and SEQ CONT L Interrupt Test
  8  DMA to Registers Test
  9  DMA to Control Table Test
 10  CWR Mode 0 Test
 11  CWR Mode 1 Test
 12  CWR Mode 2 Test
 13  CWR Mode 3, RATE CLOCK OUT and SEQ CONT Test
 14  Programmable clock Test
 15  CWR Mode 4 Test
 16  CWR Mode 5 Test
 17  CWR Mode 6 Test
 18  CWR Mode 7 Test
```

```
 19  Data Buffer Empty Test
 20  External Clock Test
 21  Linearity Test
 22  DMA Output Comparison Test
 23  Memory Transfer Conversion Test
 24  Calibration Test
 25  Selectable Output Test
 26  Loopback Test
 26  Loopback Test using ADF01
 27  Loopback Test using IEX11-A Interface
 28  Direct Read/write Test

DR> PRINT <cr>
    ----------
TYPE S, T OR HELP (S) H ? S <cr>
                          ------

AAF01-A ERROR STATISTICS.

UNIT   ERRORS   DROPPED

  0       0       NO
  1       6       YES
  2   UNTESTED    NO
```

Here, unit 0 has shown no faults, unit 1 has had 6 errors and been dropped from testing, and unit 2 has not yet been tested. Unit 2 is shown as not dropped. If the diagnostic had not yet been started, the unit would still not be shown as dropped (unlike the display command).

5.0 Device information tables

The hardware p tables contain 2 words for each device. These are used to save the answers to the startup hardware questions, and can be displayed on the console by using the "DISPLAY" command described in section 2.1.

The hardware p table is set up for field service for 1 unit with the DRX11-C address at 764200 and the vector at 270. These parameters are used if the user types "NO" to the "CHANGE HARDWARE" question.

Using the XXDP+ SETUP utility, the tables can be preloaded to contain information for specific systems.

6.0 Test summaries

Tests are divided into 2 types - default and specifically selectable. Default tests are run if no test numbers are entered following a start or restart command. Specific tests are only run if specifically selected by starting or restarting with the test number.

All except tests 1, 25, 26 and 28 begin with a DRX11-C reset to put the module into a known state prior to testing.

The following default tests check the functionality of the module and are designed for fault detection :

```
 Test :  1  DRX11-C Register NXM Test
 Test :  2  Reset Test
 Test :  3  ACS Read/write Test
 Test :  4  CTA Read/write Test
 Test :  5  Control Table Data Test
 Test :  6  Control Table Address Test
 Test :  7  COUT and SEQ CONT L Interrupt Test
 Test :  8  DMA to Registers Test
 Test :  9  DMA to Control Table Test
 Test : 10  CWR Mode 0 Test
 Test : 11  CWR Mode 1 Test
 Test : 12  CWR Mode 2 Test
 Test : 13  CWR Mode 3, RATE CLOCK OUT and SEQ CONT Test
 Test : 14  Programmable clock Test
 Test : 15  CWR Mode 4 Test
 Test : 16  CWR Mode 5 Test
 Test : 17  CWR Mode 6 Test
 Test : 18  CWR Mode 7 Test
 Test : 19  Data Buffer Empty Test
 Test : 20  External Clock Test
 Test : 21  Linearity Test
 Test : 22  DMA Output Comparison Test
 Test : 23  Memory Transfer Conversion Test
 Test : 24  Calibration Test
```

The specific tests are only run if they are selected by test number (ie. not in sequence with other tests).

```
 Test : 25  Selectable Output Test
 Test : 26  Loopback Test using ADF01
 Test : 27  Loopback Test using IEX11-A Interface
 Test : 28  Direct Read/write Test
```

Test 1 - DRX11-C Register NXM Test.

This test checks that accessing the DRX11-C SCR, COR, ADR and DBR registers does not cause a NXM trap. The following error may be printed :

```
Error 100 :      DRX11-C REGISTER ADDRESSING ERROR
                 REGISTER AT xxxxxx DOES NOT RESPOND
```

This could mean that the DRX11-C address switch is incorrectly set, that the address was entered incorrectly in the startup questions, or that a hardware fault exists in the DRX11-C addressing logic. This is the only test specifically for the DRX11-C, which should be checked separately using diagnostic YG-Z01TB-00.

Test 2 - Reset Test.

This test checks that the AAF01-A ACS register is correctly set or reset after a DRX11-C reset and after a data system reset. Both are checked to complete within 1 millisecond.

Bit 11 of the ACS register is read and a message is output to show whether it indicates unipolar or bipolar output mode eg. :

MODULE IS SWITCHED TO UNIPOLAR MODE.

The following errors may be output :

Error 200 : ACS REGISTER INCORRECT 1 MILLISECOND AFTER DRX11-C RESET
GOOD:01x000, BAD:xxxxxx

Error 201 : ACS REGISTER INCORRECT 1 MILLISECOND AFTER DATA SYSTEM RESET
GOOD:01x000, BAD:xxxxxx

Test 3 - ACS Read/write Test.

This test checks that the read/write bits of the ACS register can all be set, all cleared and individually set. The following errors may be printed :

Error 300 : ACS READ/WRITE BITS COULD NOT BE SET
GOOD:01x417, BAD:xxxxxx

Error 301 : ACS READ/WRITE BITS COULD NOT BE CLEARED
GOOD:01xx00, BAD:xxxxxx

Error 302 : ACS READ/WRITE BITS COULD NOT BE INDIVIDUALLY SET
GOOD:01xxxx, BAD:xxxxxx

Test 4 - CTA Read/write Test.

This test checks that the read/write bits of the CTA register can all be set, all cleared and individually set. The following errors may be printed :

Error 400 : CTA READ/WRITE BITS COULD NOT BE SET
GOOD:021777, BAD:xxxxxx

Error 401 : CTA READ/WRITE BITS COULD NOT BE CLEARED
GOOD:020000, BAD:xxxxxx

Error 402 : CTA READ/WRITE BITS COULD NOT BE INDIVIDUALLY SET
GOOD:02xxxx, BAD:xxxxxx

Test 5 - Control Table Data Test.

This test checks that bits 0 to 8 of each control word can all be set, all cleared, and individually set. The following errors may be printed :

Error 500 : CONTROL WORD BITS COULD NOT BE SET
CTA:02xxxx, GOOD:030777, BAD:xxxxxx

Error 501 : CONTROL WORD BITS COULD NOT BE CLEARED
CTA:02xxxx, GOOD:030000, BAD:xxxxxx

Error 502 : CONTROL WORD BITS COULD NOT BE INDIVIDUALLY SET
CTA:02xxxx, GOOD:030xxx, BAD:xxxxxx

Test 6 - Control Table Address Test.

This checks for interaction between different words of the control table. All 1024 control words are loaded with a modulo 511 count. The entire control table is then read and the contents of each word compared with the data which was written to it. To check bit 9 of the address logic the first half of the control table is loaded with all zeros and the second half with all ones. The table is again read to check that the data was correctly written.

The following error may be printed :

Error 600 : CONTROL TABLE ADDRESSING ERROR, BITS 0-8
CTA:02xxxx, GOOD:030xxx, BAD:xxxxxx

Error 601 : CONTROL TABLE ADDRESSING ERROR, BIT 9
CTA:02xxxx, GOOD:030xxx, BAD:xxxxxx

Test 7 - COUT and SEQ CONT L Interrupt Test.

This tests that the command output (COUT) signal can be set and that the SEQ CONT L input signal can be used to generate an interrupt.

The interrupt enable bit (6) of the DRX11-C SCR register is set and in the AAF01-A's ACS register SBE is set. COUT is then set and cleared. COUT is looped back to the SEQ CONT L input signal, and the trailing edge sets SBR in the ACS register. This in turn sets EOC and STAT0 in the DRX11-C SCR causing an interrupt.

The following errors may be printed :

Error 700 : INTERRUPT TIMEOUT AFTER COUT LOOPBACK

Error 701 : ACS REGISTER INCORRECT AFTER COUT LOOPBACK
GOOD:01x250, BAD:xxxxxx

Error 702 : DRX11-C SCR REGISTER INCORRECT AFTER COUT LOOPBACK
GOOD:100700, BAD:xxxxxx

Test 8 - DMA to Registers Test.

This test performs a DMA write to the ACS and CTA registers. The register contents are then read back under program control and checked. The following errors may be printed :

Error 800 : NO INTERRUPT AFTER DMA TO REGISTERS

Error 801 : ACS INCORRECT AFTER DMA WRITE
GOOD:01x416, BAD:xxxxxx

Error 802 : CTA INCORRECT AFTER DMA WRITE
GOOD:020525, BAD:xxxxxx

Test 9 - DMA to Control Table Test.

This test performs a single block DMA to all 1024 words of the control table. The control table contents are then read back under program control and checked. The following errors may be printed :

Error 900 : NO INTERRUPT AFTER DMA TO CONTROL TABLE

Error 901 : CONTROL TABLE WORD INCORRECT AFTER DMA
CTA:02xxxx, GOOD:030xxx, BAD:xxxxxx

Test 10 - CWR Mode 0 Test.

This tests that after each conversion in mode 0, the CTA register is incremented. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage
0	0	0	1.0
1	0	1	2.0
2	0	2	3.0
3	0	3	0.0

Conversions are then initiated with a DMA count of 3. After the conversions are complete, the CTA register is checked to ensure that it was correctly incremented.

The following errors may be output :

Error 1000 : DRX11-C SCR INCORRECT AFTER MODE 0 CONVERSIONS
GOOD:106000, BAD:xxxxxx

Error 1001 : ACS INCORRECT AFTER MODE 0 CONVERSIONS
GOOD:01x000, BAD:xxxxxx

Error 1002 : CTA INCORRECT AFTER MODE 0 CONVERSIONS
GOOD:020003, BAD:xxxxxx

Test 11 - CWR Mode 1 Test.

This tests that after a conversion in mode 1, the CTA register is reset to 0. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage
0	0	0	1.0
1	0	1	2.0
2	0	2	3.0
3	0	3	4.0
4	0	4	5.0
5	1	5	6.0

Conversions are then initiated with a DMA count of 6. After the conversions are complete the CTA register is checked to ensure that it was correctly reset.

The following errors may be output :

Error 1100 : DRX11-C SCR INCORRECT AFTER MODE 1 CONVERSIONS
GOOD:106000, BAD:xxxxxx

Error 1101 : ACS CONTENTS INCORRECT AFTER MODE 1 CONVERSION
GOOD:01x000, BAD:xxxxxx

Error 1102 : CTA REGISTER INCORRECT AFTER MODE 1 CONVERSION
GOOD:020000, BAD:xxxxxx

Test 12 - CWR Mode 2 Test.

This tests that dummy control table loads can be made by using mode 2. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage
0	0	0	1.0
1	0	1	2.0
2	2	1	2.0
3	2	1	2.0
4	2	1	2.0
5	1	2	3.0

The SCR interrupt enable bit is set and the DMA is started with the DRX11-C WCR loaded for 3 DMA's. Conversions are then initiated and the program waits for the conversions to complete.

The following errors may be printed :

Error 1200 : DRX11-C SCR INCORRECT AFTER MODE 2 CONVERSIONS
GOOD:106000, BAD:xxxxxx

Error 1201 : ACS INCORRECT AFTER MODE 2 DMA OUTPUT
GOOD:01x000, BAD:xxxxxx

Error 1202 : CTA INCORRECT AFTER MODE 2 DMA OUTPUT
GOOD:020000, BAD:xxxxxx

Test 13 - CWR Mode 3, RATE CLOCK OUT and SEQ CONT Test.

This tests that the SEQ CONT signal can initiate and reinitiate conversions in mode 3. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage
0	3	0	0.0
1	0	0	1.0
2	3	0	1.0
3	1	1	2.0

The SCR interrupt enable bit is set and the DMA is started with the DRX11-C WCR loaded for 2 DMA's. The RATE CLK OUT signal is connected to the SEQ CONT input via the test connector and so triggers the conversions. The CTA register is checked to ensure that this occurred.

The following errors may be printed :

Error 1300 : DRX11-C SCR INCORRECT AFTER SEQ CONT CONVERSIONS
GOOD:106000, BAD:xxxxxx

Error 1301 : ACS CONTENTS INCORRECT AFTER SEQ CONT CONVERSION
GOOD:01x000, BAD:xxxxxx

Error 1302 : CTA REGISTER INCORRECT AFTER SEQ CONT CONVERSION
GOOD:020000, BAD:xxxxxx

Test 14 - Programmable clock test

This tests the speed and count functionality of the programmable clock. The PCR is loaded with the value 4000 to give a conversion time of 400 microseconds. The control table is set up to perform 2 conversions at this rate and a check is made that the conversions require between 1.0 and 1.5 milliseconds to complete.

The 2 conversions are made in mode 0, outputing a value of 5.0 volts on channels 0 and 1.

The following errors may be output :

Error 1400 : CONVERSIONS TOO FAST IN CLOCK TEST

Error 1401 : CONVERSIONS TOO SLOW IN CLOCK TEST

Error 1402 : ACS INCORRECT IN CLOCK TEST
GOOD:01x000, BAD:xxxxxx

Error 1403 : CTA INCORRECT IN CLOCK TEST
GOOD:020002, BAD:xxxxxx

Test 15 - CWR Mode 4 Test.

This tests that dummy control table loads can be made by using mode 4. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage
0	0	0	1.0
1	0	1	2.0
2	4	1	2.0
3	4	1	2.0
4	0	2	3.0
5	1	3	4.0

The SCR interrupt enable bit is set and the DMA is started with the DRX11-C WCR loaded for 4 DMA's. Conversions are then initiated and the program waits for the conversions to complete.

The following errors may be printed :

Error 1500 : DRX11-C SCR INCORRECT AFTER MODE 4 CONVERSIONS
GOOD:106000, BAD:xxxxxx

Error 1501 : ACS INCORRECT AFTER MODE 4 DMA OUTPUT
GOOD:01x000, BAD:xxxxxx

Error 1502 : CTA INCORRECT AFTER MODE 4 DMA OUTPUT
GOOD:020000, BAD:xxxxxx

Test 16 - CWR Mode 5 Test.

This tests that dummy control table loads can be made by using mode 5. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage (first pass)	Output voltage (second pass)
0	0	0	1.0	4.0
1	0	1	2.0	5.0
2	4	1	2.0	5.0
3	4	1	2.0	5.0
4	0	2	3.0	6.0
5	5	2	3.0	6.0

The SCR interrupt enable bit is set and the DMA is started with the DRX11-C WCR loaded for 6 DMA's to use the control table twice. Conversions are then initiated and the program waits for the conversions to complete.

The following errors may be printed :

Error 1600 : DRX11-C SCR INCORRECT AFTER MODE 5 CONVERSIONS
GOOD:106000, BAD:xxxxxx

Error 1601 : ACS INCORRECT AFTER MODE 5 DMA OUTPUT
GOOD:01x000, BAD:xxxxxx

Error 1602 : CTA INCORRECT AFTER MODE 5 DMA OUTPUT
GOOD:020005, BAD:xxxxxx

Test 17 - CWR Mode 6 Test.

This tests that dummy control table loads can be made by using mode 6. The PCR is loaded with the value 95 (complement of 4000) to give a conversion time of 400 microseconds. The control table is then set up to perform 5 conversions. All except the first and last conversions are made in mode 6. The first and last are made in mode 0 and mode 1, outputing a value of 1.0 volts and 2.0 volts respectively on channels 0 and 1.

The SCR interrupt enable bit is set and a DMA is started with the DRX11-C WCR loaded for 2 DMA's. Conversions are then initiated and the program checks that the conversions require between 1 and 1.5 milliseconds to complete.

The following errors may be printed :

Error 1700 : CONVERSIONS TOO FAST IN CWR MODE 6

Error 1701 : CONVERSIONS TOO SLOW IN CWR MODE 6

Error 1702 : ACS INCORRECT AFTER MODE 6 DMA OUTPUT
GOOD:01x000, BAD:xxxxxx

Error 1703 : CTA INCORRECT AFTER MODE 6 DMA OUTPUT
GOOD:02x000, BAD:xxxxxx

Test 18 - CWR Mode 7 Test.

This tests that dummy control table loads can be made by using mode 7. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage	
0	0	0	1.0	
1	7	0	1.0	
2	7	0	1.0	
3	0	1	2.0	- should
4	0	2	3.0	> not be
5	0	3	4.0	- output

The SCR interrupt enable and ACS SBE bits are set and the DMA is started with the DRX11-C WCR loaded for 3 DMA's. Conversions are then initiated. As the second conversion completes, COUT goes low causing the input SEQ CONT L to go low. The diagnostic checks that an SBR interrupt then occurs and that conversions stop with the CTA register pointing to the 4th control word.

The following errors may be printed :

Error 1800 : DRX11-C SCR INCORRECT AFTER MODE 7 CONVERSIONS
GOOD:100600, BAD:xxxxxx

Error 1801 : ACS INCORRECT AFTER MODE 7 DMA OUTPUT
GOOD:01x250, BAD:xxxxxx

Error 1802 : CTA INCORRECT AFTER MODE 7 DMA OUTPUT
GOOD:020003, BAD:xxxxxx

Test 19 - Data Buffer Empty Test.

This tests that an interrupt can be generated when the AAF01-A data buffer becomes empty.

The ACS external clock enable bit is set to prevent conversions. The control table is then set up for 66 conversions in mode 0, to output a value of 5.0 volts on sequential (modulo 15) channels.

DMA's are then initiated and the diagnostic waits for 1 millisecond for the FIFO to be fully loaded. The DRX11-C COR maintenance bit is then set to prevent further DMA's. The ECE bit is then cleared to allow conversions. A check is made that the DBE bit causes an interrupt within a specific timeout period and that the CTA register indicates that 64 conversions took place.

The following errors may be printed :

Error 1900 : NO DATA BUFFER EMPTY INTERRUPT

Error 1901 : DRX11-C SCR INCORRECT AFTER DATA BUFFER EMPTY TEST
GOOD:110600, BAD:xxxxxx

Error 1902 : ACS REGISTER INCORRECT AFTER DATA BUFFER EMPTY TEST
GOOD:01x300, BAD:xxxxxx

Error 1903 : CTA REGISTER INCORRECT AFTER DATA BUFFER EMPTY TEST
GOOD:020102, BAD:xxxxxx

Test 20 - External Clock Test.

This checks that the RATE CLOCK IN/OUT line can be used to trigger conversions. The control table and values for output are set up as follows :

Control word	Mode	Channel	Output voltage
0	0	0	1.0
1	0	1	2.0
2	1	2	3.0

The external clock enable and maintenance bits are set and a 3 word DMA is enabled. The diagnostic then waits for 1 millisecond for the data to be loaded into the FIFO, after which COUT is set and cleared 3 times. This signal is looped back to the RATE CLOCK IN/OUT line via the test connector. After the third pulse a check is made that an End of conversion interrupt occurred and that the SCR and ACS registers are correct.

The following errors may be printed :

Error 2000 : NO INTERRUPT AFTER EXTERNAL CLOCK TEST

Error 2001 : ACS REGISTER INCORRECT AFTER EXTERNAL CLOCK TEST
GOOD:01x004, BAD:xxxxxx

Test 21 - Linearity Test.

This checks the linearity of the AAF01-A. Each possible value is output to channel 15. For each of these outputs, two outputs are made on channel 0, the first using a value N bits greater than that on channel 15, and the second, N bits lower. N is equal to the value entered as "TOLERANCE" in the startup software questions. A check is made that the first output on channel 0 results in a logical 0 from the comparator logic, and that the second results in a 1. All outputs are made by program using the DBF register. At the range boundaries, only one comparison with channel 0 is made.

The following error may be printed :

Error 2100: ACS INCORRECT AFTER DBF OUTPUT COMPARISON
GOOD:01xxxx, BAD:01xxxx, CHAN15:xxxx, CHAN0:xxxx

Test 22 - DMA Output Comparison Test.

This checks that conversions can be made under DMA control. The control table is set up to perform 2 outputs, on channels 0 and 15. An internal buffer is set up to contain the data 1777 and 3777. These are selected so that the ACS CMP bit wil be set. The DRX11-C is then set up for a 2 word DMA and conversions are initiated by setting the GO bit of the ACS register. After the DMA has completed, the ACS register CMP bit is checked to be set. The test is then repeated but with the data words interchanged so that the ACS CMP bit is not set.

The following error may be printed :

Error 2200 : DRX11-C SCR INCORRECT AFTER DMA OUTPUT COMPARISON
GOOD:106000, BAD:xxxxxx

Error 2201 : ACS INCORRECT AFTER DMA OUTPUT COMPARISON
GOOD:01xxxx, BAD:01xxxx CHAN15:xxxx, CHAN0:xxxx

Test 23 - Memory Transfer Conversion Test.

This tests DMA conversions in memory transfer mode. An internal buffer is set up to contain the following control and data words :

Word	Contents	Description
0	0000	Control word, mode 0, channel 0
1	0000	Data, 9.9976V unipolar or 9.9951V bipolar
2	0001	Control word, mode 0, channel 1
3	1777	Data, 7.5000V unipolar or 5.0000V bipolar
4	0201	Control word, mode 2, channel 1
5	0201	Control word, mode 2, channel 1
6	0002	Control word, mode 0, channel 2
7	3777	Data, 5.0000V unipolar or 0.0000V bipolar
8	0003	Control word, mode 0, channel 3
9	5777	Data, 2.5000V unipolar or -5.0000V bipolar
10	0303	Control word, mode 3, channel 3
11	0004	Control word, mode 0, channel 4
12	7776	Data, 0.0024V unipolar or -9.9951V bipolar
13	0404	Control word, mode 4, channel 4
14	0005	Control word, mode 0, channel 5
15	7777	Data, 0.0000V unipolar or -10.0000V bipolar
16	0605	Control word, mode 6, channel 5
17	0017	Control word, mode 0, channel 15
18	3777	Data, 5.0000V unipolar or 0.0000V bipolar
19	0717	Control word, mode 7, channel 15

These are selected to test combinations of CWR modes and channels so that at the end of the conversion the ACS CMP bit will be set.

The DRX11-C is set up to start a 20 word DMA. The GO and MET bits of the ACS register are then set. After the DMA has completed, the ACS register MET and CMP bits are checked to be set.

The test is then repeated but with the data words for channels 0 and 15 complemented so that after the conversions the ACS CMP bit is not set.

The following errors may be printed :

Error 2300 : DRX11-C SCR INCORRECT AFTER MEMORY MODE DMA CONVERSION
GOOD:106000, BAD:xxxxxx

Error 2301 : ACS REGISTER INCORRECT AFTER MEMORY MODE DMA CONVERSION
GOOD:01x002, BAD:xxxxxx

Test 24 - Calibration Test.

This allows the AAF01-A analogue output to be calibrated. ACS bit 11 is first read and a message printed indicating whether the module is switched to unipolar or bipolar mode Eg. :

CALIBRATING FOR UNIPOLAR MODE

In bipolar mode, each of the channels selected in the startup questions is loaded with the value 7777 and the user is asked to check that the DVM reads -10.000 volts +/- 4.88 millivolts on the selected channels. If the output is not within this tolerance, the user is asked to adjust the offset potentiometer RV1 until the outputs are correct. Each channel is then loaded with the value 0000 and the user asked to check that the DVM now reads 9.9951 volts +/- 4.88 millivolts. If the output is not within this tolerance, he is asked to adjust the gain potentiometer RV2 until the output is correct.

In unipolar mode, each of the channels selected in the startup questions is loaded with the value 7777 and the user is asked to check that the DVM reads 0.0000 volts +/- 4.88 millivolts on the selected channels. If the output is not within this tolerance, the user is asked to adjust the DAC OFFS potentiometer RV1 until outputs are correct. Each channel is then loaded with the value 0000 and the user asked to check that the DVM now reads 9.9976 volts +/- 4.88 millivolts. If the output is not within this tolerance, he is asked to adjust the gain potentiometer RV2 until the output is correct.

All outputs are made directly through the DBF register.

After the module has been tested and/or calibrated, the following message is output describing how to continue running the diagnostic without the calibration test :

THIS TEST CAN BE DISABLED BY ABORTING THE TEST AND SETTING THE "UAM" FLAG. EG. "CONTROL C" THEN "CONTINUE/FLAGS:UAM".
TYPE "CARRIAGE RETURN TO CONTINUE OR "CONTROL C" TO ABORT (A) ?

No error messages are output by this test.

Test 25 - Selectable Output Test.

This test allows the user to select output patterns for monitoring with an oscilloscope.

The test begins by requesting the patterns for the test :

FIRST VALUE IN MILLIVOLTS (A) 0.0 ?

SECOND VALUE IN MILLIVOLTS (A) 10000.0 ?

After the user has typed in each value, the program prints out the actual value which will be output and its equivalent bit pattern in octal. The program then asks :

USE BOTH VALUES FOR EACH CHANNEL (L) N ?

Two outputs are made on each of the channels selected in the startup questions. Either both values are output for each channel or the values switch with each new channel according to the response to the third question. The control table is set up to make all outputs except the last in mode 0. The last is in mode 1 to force a return back to the begining of the table. Continuous outputs are made with the DRX11-C in double buffer mode until the user types control C.

The following error may be printed :

Error 2500 : SCR ATTENTION BIT SET DURING DOUBLE BUFFER DMA OUTPUT
SCR:xxxxxx, ACS:xxxxxx

Error 2501 : INTERRUPT TIMEOUT DURING DOUBLE BUFFER DMA OUTPUT
SCR:xxxxxx, ACS:xxxxxx

Test 26 - Loopback Test using ADF01.

This tests that the AAF01-A conversion accuracy is within a specified tolerance.

The test begins by requesting the following parameters :

ADF01 DRX11-C ADDRESS (O) 164210 ?

In reply, you should enter the address of the DRX11-C to which an ADF01 is connected. This ADF01 will be used as the other half of a looped pair. This question is only asked following a "START" command.

CHANGE LOOPBACK PARAMETERS (L) ?

This allows the parameters used for loopback testing to be changed. Answering "no" forces the use of the default parameters, which cause the most thorough testing, and eliminates the need for any more questions. If "Y" is typed, the following questions are also asked :

PATTERN (1=RAMP, 2=RANDOM, 3=PAIR) (D) 1 ?

This selects how output is made from the AAF01-A. A sequence of 2048 outputs is made. If "1" was typed in response to this question, the output ramps up from the lowest voltage to the highest and back again each pass of the test. If "2" is selected, random values are output. Responding with "3" results in the output alternating according to the answers to the next two questions.

NOTE : If "2" or "3" are selected, a conversion time of around 10 microseconds should be chosen to inhibit the refresh logic. If the refresh logic operates in this test, errors may be printed which do not reflect real hardware faults.

FIRST VALUE IN MILLIVOLTS (A) 0.0 ?

SECOND VALUE IN MILLIVOLTS (A) 10000.0 ?

After the user has typed in each value, the program prints out the actual value which will be output and its equivalent bit pattern in octal. The program then asks :

ALTERNATING CHANNELS (L) Y ?

Normally, 2048 values are output to the first of the selected channels and checked to be correct. Outputs are then made to the next channel and so on until each channel has been used. If the answer to the "alternating channels" question is "yes", 2048 outputs are made alternating between the first and second of the selected channels. After checking the results, a further 2048 outputs are made to the third and fourth of the selected channels and checked. This continues until all channels have

been checked. If the number of selected channels is odd, the last channel is paired with the subsequent channel.

If a paired data pattern and alternating channels are both selected, the effect is that the first of the data values is output 1024 times to the first of the channels in the pair and the second value to the second channel. This results in a static output on each channel, each second channel having the same voltage.

After the parameters have been input, a system reset is made and the output from each selected channel is read via the DRX11-C/ADF01 and checked to be zero.

Starting with the first of the selected channels, 2048 outputs are made under DMA and the program waits for DMA ready. The output is read back via the DRX11-C/ADF01 analogue input. The RATE CLK OUT signal is used to synchronize the two devices. When the transfer is complete, the received data is compared with that which was output and checked to be within the specified tolerance.

NOTE : The DMA is divided into 64 word blocks, the first half of which consists of dummy outputs. If the outputs are being observed on an oscilloscope, a delayed trigger is necessary to see the data changing.

The test then proceeds with the next channel and continues until all selected channels have been tested.

If "alternating channels" was selected at the start of the test, then the list of selected channels is split into pairs. The test sequences through the pairs alternating output between each channel in the pair.

The following errors may be printed :

Error 2600 : ADF01 DRX11-C REGISTER ADDRESSING ERROR
REGISTER AT xxxxxx DOES NOT RESPOND

Error 2601 : ADF01 NOT IN SAME MODE AS AAF01-A
ADF01:unipolar, AAF01-A:bipolar

Error 2602 : TIMEOUT ON DMA INPUT
SCR:xxxxxx, ACS:xxxxxx

Error 2603 : ERROR ON DMA INPUT
SCR:xxxxxx, ACS:xxxxxx

Error 2604 : OUTPUTS NOT ZERO AFTER A SYSTEM RESET
CHANNEL:xx, GOOD:xxxx, BAD:xxxx

Error 2605 : DRX11-C SCR INCORRECT IN LOOPBACK TEST
GOOD:106000, BAD:xxxxxx

Error 2606 : ACS REGISTER INCORRECT AFTER LOOPBACK TEST
GOOD:010000, BAD:xxxxxx

Error 2607 : LOOP BACK DATA NOT WITHIN SPECIFIED TOLERANCE
CHANNEL:xx, GOOD:xxxx, BAD:xxxx
RUN TEST 24 TO CALIBRATE THE MODULE

Test 27 - Loopback Test using IEX11-A Interface.

This tests that the AAF01-A conversion accuracy is within a specified tolerance. The output from the AAF01-A is read using a digital voltmeter connected to an IEX11-A interface. This provides more accurate measurement than the ADF01 used in test 26.

The test begins by requesting the following parameters :

IEX11-A ADDRESS (O) 164100 ?

In reply, you should enter the address of the IEX11-A to which a digital voltmeter is connected. This IEX11-A will be used to read back the outputs of the AAF01-A. This question is only asked following a "start" command.

CHANGE LOOPBACK PARAMETERS (L) ?

This allows the parameters used for loopback testing to be changed. Answering "no" forces the use of the default parameters, which cause the most thorough testing, and eliminates the need for any more questions. If "Y" is typed, the following questions are also asked :

LOOPBACK TOLERANCE IN MILLIVOLTS (A) 10.0 ?

If the difference between output and input values is greater than this value, an error message will be output.

PATTERN (1=RAMP, 2=RANDOM, 3=PAIR) (D) 1 ?

This selects how output is made from the AAF01-A. A sequence of 4095 decimal outputs is made. If "1" was typed in response to this question, the output ramps up from the lowest voltage to the highest and back again. If "2" is selected, random values are output. Responding with "3" results in the output alternating according to the answers to the next two questions.

FIRST VALUE IN MILLIVOLTS (A) 0.0 ?

SECOND VALUE IN MILLIVOLTS (A) 10000.0 ?

After the user has typed in each value, the program prints out the actual value which will be output and its equivalent bit pattern in octal.

After the parameters have been input, a system reset is made.

Using the channel selected as "FIRST CHANNEL" in the startup questions, 4096 outputs are made. After each output, the IEX11-A is used to check that the resulting voltage is within the specified tolerance. At the end of each pass, the maximum deviation from the expected value is printed.

The following errors may be printed :

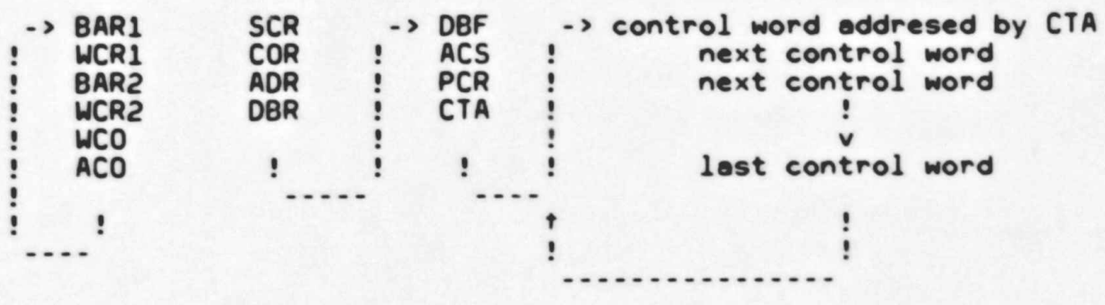
```
Error 2700 : IEX11-A REGISTER ADDRESSING ERROR
             REGISTER AT xxxxxx DOES NOT RESPOND

Error 2701 : ACS REGISTER INCORRECT AFTER OUTPUT TO IEX11-A
             GOOD:010000, BAD:xxxxxx

Error 2702 : ERROR ON IEX11-A INPUT
             ERROR NUMBER:x

Error 2703 : LOOP BACK DATA NOT WITHIN SPECIFIED TOLERANCE
             GOOD:xxxxx.xxx MILLIVOLTS, BAD:xxxxx.xxx MILLIVOLTS
```

Test 28 - Direct Read/write Test.

This allows the registers of the AAF01-A to be read from or written to (where the hardware allows). Multiplexing through the DRX11-C to the AAF01-A and further to the control table is performed by the program.

Normally, the program will use unit 0. However, any unit can be selected by starting the test with the /UNIT switch (see section 2.2).

Registers are specified using 3 or 4 character mnemonics. After a register has been read and if required a new value input, another register will automatically be opened according to the sequence shown below.



To exit from any sequence, "control Z" can be typed to return to the "REG" prompt. Typing "CONTROL C" returns the user to the supervisor. The following example shows how the test can be used. User input is underlined :

```
START/TEST:28/UNIT:3
--------------------
CHANGE HARDWARE (L) ? N <cr>
                      ------
CHANGE SOFTWARE (L) ? N <cr>
                      ------
REG (S) SCR ? PCR <cr>
              --------
PCR (0) xxxxxx ? <cr>
                 ----
CTA (0) xxxxxx ? 0 <cr>
                 ------
CWR 0000 (0) xxxx ? 123 <cr>
                    --------
CWR 0001 (0) xxxx ? 456 <cr>
                    --------
CWR 0002 (0) xxxx ? <control Z>
                    -----------
REG (S) CTA ? <control C>
              -----------
```

7.0 Design and Manufacturing Checkout.

Whilst the default tests described above should be enough to test most of the module functionality, it is recommended that additional testing is performed in the design and manufacturing stages of the module development.

In the design phase, the following should be checked :

a. The module works in unipolar and bipolar mode.

b. Conversion accuracy at rates from 80 to 400 KHz is as described in the Design Specification.

c. The module passes the loopback test with all legal parameter combinations.

d. In test 13 the logic waits for SEQ CONT when operating in CWR mode 3.

In the manufacturing phase, the following should be checked :

a. The module works in unipolar and bipolar mode.

b. The module passes the loopback test in unipolar mode with the default parameters.

c. The RATE CLOCK OUT output signal should be checked with an oscilloscope.

d. Using an oscilloscope, the SYS IN PROG L output signal should be verified to be set during conversions in test 12 (mode 2) and to remain not set during conversions in test 15 (mode 4).

The above manufacturing checks probably need to be made on a small sample of modules from each batch.

```
                                .TITLE PROGRAM HEADER AND TABLES
                                .SBTTL  PROGRAM HEADER
                                .ENABL  LC

                                        .ENABL  ABS,AMA
                        002000          .       =       2000

002000                                  BGNMOD

                                ;++
                                ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
                                ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
                                ;--

002000                                  POINTER BGNRPT,BGNSW,BGNSFT,BGNAU,BGNDU,BGNSETUP

002000                                  HEADER  CZAAF,A,0,45,0,340
002000                                                                  L$NAME::
002000  103                                                                     .ASCII /C/
002001  132                                                                     .ASCII /Z/
002002  101                                                                     .ASCII /A/
002003  101                                                                     .ASCII /A/
002004  106                                                                     .ASCII /F/
002005  000                                                                     .BYTE   0
002006  000                                                                     .BYTE   0
002007  000                                                                     .BYTE   0
002010                                                                  L$REV::
002010  101                                                                     .ASCII  /A/
002011                                                                  L$DEPO::
002011  060                                                                     .ASCII  /0/
002012                                                                  L$UNIT::
002012  000001                                                                  .WORD   T$PTHV
002014                                                                  L$TIML::
002014  000045                                                                  .WORD   45
002016                                                                  L$HPCP::
002016  002242                                                                  .WORD   L$HARD
002020                                                                  L$SPCP::
002020  002334                                                                  .WORD   L$SOFT
002022                                                                  L$HPTP::
002022  002216                                                                  .WORD   L$HW
002024                                                                  L$SPTP::
002024  002224                                                                  .WORD   L$SW
002026                                                                  L$LADP::
002026  100004                                                                  .WORD   L$LAST
002030                                                                  L$STA::
002030  000000                                                                  .WORD   0
002032                                                                  L$CO::
002032  000000                                                                  .WORD   0
002034                                                                  L$DTYP::
002034  000000                                                                  .WORD   0
002036                                                                  L$APT::
002036  000000                                                                  .WORD   0
002040                                                                  L$DTP::
002040  002124                                                                  .WORD   L$DISPATCH
002042                                                                  L$PRIO::
```

```
PROGRAM HEADER

002042  000340                                          .WORD   340
002044                                          L$ENVI::
002044  000000                                          .WORD   0
002046                                          L$EXP1::
002046  000000                                          .WORD   0
002050                                          L$MREV::
002050     003                                          .BYTE   C$REVISION
002051     003                                          .BYTE   C$EDIT
002052                                          L$EF::
002052  000000                                          .WORD   0
002054  000000                                          .WORD   0
002056                                          L$SPC::
002056  000000                                          .WORD   0
002060                                          L$DEVP::
002060  023202                                          .WORD   L$DVTYP
002062                                          L$REPP::
002062  032234                                          .WORD   L$RPT
002064                                          L$EXP4::
002064  000000                                          .WORD   0
002066                                          L$EXP5::
002066  000000                                          .WORD   0
002070                                          L$AUT::
002070  035450                                          .WORD   L$AU
002072                                          L$DUT::
002072  035362                                          .WORD   L$DU
002074                                          L$LUN::
002074  000000                                          .WORD   0
002076                                          L$DESP::
002076  023212                                          .WORD   L$DESC
002100                                          L$LOAD::
002100  104035                                          EMT     E$LOAD
002102                                          L$ETP::
002102  000000                                          .WORD   0
002104                                          L$ICP::
002104  033532                                          .WORD   L$INIT
002106                                          L$CCP::
002106  035266                                          .WORD   L$CLEAN
002110                                          L$ACP::
002110  035204                                          .WORD   L$AUTO
002112                                          L$PRT::
002112  033524                                          .WORD   L$PROT
002114                                          L$TEST::
002114  000000                                          .WORD   0
002116                                          L$DLY::
002116  000000                                          .WORD   0
002120                                          L$HIME::
002120  000000                                          .WORD   0
1825
```

DISPATCH TABLE

```
   1837                                 .SBTTL  DISPATCH TABLE
   1838
   1839                                 ;++
   1840                                 ; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
   1841                                 ; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
   1842                                 ;--
   1843
   1844 002122                                 DISPATCH 28.
        002122  000034                                                          .WORD   28
        002124                                                          L$DISPATCH::
        002124  035462                                                          .WORD   T1
        002126  035776                                                          .WORD   T2
        002130  036720                                                          .WORD   T3
        002132  037602                                                          .WORD   T4
        002134  040414                                                          .WORD   T5
        002136  041252                                                          .WORD   T6
        002140  042120                                                          .WORD   T7
        002142  043014                                                          .WORD   T8
        002144  043652                                                          .WORD   T9
        002146  044472                                                          .WORD   T10
        002150  045442                                                          .WORD   T11
        002152  046424                                                          .WORD   T12
        002154  047406                                                          .WORD   T13
        002156  050422                                                          .WORD   T14
        002160  051512                                                          .WORD   T15
        002162  052474                                                          .WORD   T16
        002164  053456                                                          .WORD   T17
        002166  054656                                                          .WORD   T18
        002170  055640                                                          .WORD   T19
        002172  057164                                                          .WORD   T20
        002174  060144                                                          .WORD   T21
        002176  061024                                                          .WORD   T22
        002200  062002                                                          .WORD   T23
        002202  062774                                                          .WORD   T24
        002204  065362                                                          .WORD   T25
        002206  067226                                                          .WORD   T26
        002210  073120                                                          .WORD   T27
        002212  075500                                                          .WORD   T28
   1845
```

DEFAULT HARDWARE P-TABLE

```
                                .SBTTL  DEFAULT HARDWARE P-TABLE

                                ;++
                                ; THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
                                ; THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
                                ; IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
                                ; AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
                                ;--

002214                                  BGNHW   DFPTBL
002214  000002                                                                 .WORD   L10000-L$HW/2
002216                                                                  L$HW::
002216                                                                  DFPTBL::

002216  164200                                  .WORD   164200          ; DRX11-C SCR address
002220  000270                                  .WORD   270             ; DRX11-C vector address

002222                                  ENDHW
002222                                                                  L10000:
```

SOFTWARE P-TABLE

```
1878                                    .SBTTL  SOFTWARE P-TABLE
1879
1880                                    ;++
1881                                    ; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
1882                                    ; PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
1883                                    ; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
1884                                    ; AT RUN TIME.
1885                                    ;--
1886
1887 002222                                    BGNSW   SFPTBL
     002222  000006                                                                .WORD   L10001-L$SW/2
     002224                                                                 L$SW::
     002224                                                                 SFPTBL::
1888
1896
1897 002224  000000                    FSTCHN::        .WORD   0       ; first channel
1898 002226  000017                    LSTCHN::        .WORD   15.     ; last channel
1899 002230  000310                    CTIME::         .WORD   200.    ; conversion time
1900 002232  000007                    TOL::           .WORD   7       ; tolerance
1901 002234  000000                    QVMODE::        .WORD   0       ; quick verify mode ?
1902 002236  000000                    RDUMP::         .WORD   0       ; register dump ?
1903
1904 002240                                    ENDSW
     002240                                                             L10001:
```

HARDWARE PARAMETER CODING SECTION

```
                                .SBTTL  HARDWARE PARAMETER CODING SECTION

                                ;++
                                ; THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
                                ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
                                ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
                                ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
                                ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
                                ; WITH THE OPERATOR.
                                ;--

002240                                  BGNHRD
002240  000010                                                            .WORD L10002-L$HARD/2
002242                                                                  L$HARD::

                                GPRMA   G1,0,0,160000,177776,YES        ; DRX11-C SCR address
002242  000031                                                          .WORD   T$CODE
002244  002262                                                          .WORD   G1
002246  160000                                                          .WORD   T$LOLIM
002250  177776                                                          .WORD   T$HILIM
                                GPRMA   G2,2,0,0,770,YES                ; DRX11-C vector address
002252  001031                                                          .WORD   T$CODE
002254  002302                                                          .WORD   G2
002256  000000                                                          .WORD   T$LOLIM
002260  000770                                                          .WORD   T$HILIM

                                ENDHRD
                                                                        .EVEN
002262                                                                  L10002:

                                .NLIST  BEX
002262  104  122  130  G1:      .ASCIZ  /DRX11-C ADDRESS/
002302  104  122  130  G2:      .ASCIZ  /DRX11-C VECTOR ADDRESS/
                                .LIST   BEX
                                .EVEN
```

```
                                .SBTTL  SOFTWARE PARAMETER CODING SECTION

                                ;++
                                ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
                                ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
                                ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
                                ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
                                ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
                                ; WITH THE OPERATOR.
                                ;--

002332                                  BGNSFT
002332  000032                                                          .WORD L10003-L$SOFT/2
002334                                                                L$SOFT::

002334                                  GPRMD   G3,0,D,-1,0,64.,YES             ; first channel
002334  000052                                                          .WORD   T$CODE
002336  002420                                                          .WORD   G3
002340  177777                                                          .WORD   -1
002342  000000                                                          .WORD   T$LOLIM
002344  000100                                                          .WORD   T$HILIM
002346                                  GPRMD   G4,2,D,-1,0,64.,YES             ; last channel
002346  001052                                                          .WORD   T$CODE
002350  002436                                                          .WORD   G4
002352  177777                                                          .WORD   -1
002354  000000                                                          .WORD   T$LOLIM
002356  000100                                                          .WORD   T$HILIM
002360                                  GPRMD   G5,4,D,-1,1,4095.,YES           ; conversion time
002360  002052                                                          .WORD   T$CODE
002362  002453                                                          .WORD   G5
002364  177777                                                          .WORD   -1
002366  000001                                                          .WORD   T$LOLIM
002370  007777                                                          .WORD   T$HILIM
002372                                  GPRMD   G6,6,D,-1,0,4095.,YES           ; tolerance
002372  003052                                                          .WORD   T$CODE
002374  002522                                                          .WORD   G6
002376  177777                                                          .WORD   -1
002400  000000                                                          .WORD   T$LOLIM
002402  007777                                                          .WORD   T$HILIM
002404                                  GPRML   G7,10,-1,YES                    ; quick verify mode ?
002404  004130                                                          .WORD   T$CODE
002406  002544                                                          .WORD   G7
002410  177777                                                          .WORD   -1
002412                                  GPRML   G8,12,-1,YES                    ; register dump ?
002412  005130                                                          .WORD   T$CODE
002414  002566                                                          .WORD   G8
002416  177777                                                          .WORD   -1

                                .EVEN

002420                                  ENDSFT
                                                                        .EVEN
002420                                                                L10003:

                                .NLIST  BEX
```

SOFTWARE PARAMETER CODING SECTION

```
1988 002420     106     111     122  G3:     .ASCIZ  /FIRST CHANNEL/
1989 002436     114     101     123  G4:     .ASCIZ  /LAST CHANNEL/
1990 002453     103     117     116  G5:     .ASCIZ  /CONVERSION TIME (100 NANOSECOND STEPS)/
1991 002522     124     117     114  G6:     .ASCIZ  /TOLERANCE IN BITS/
1992 002544     121     125     111  G7:     .ASCIZ  /QUICK VERIFY MODE/
1993 002566     122     105     107  G8:     .ASCIZ  /REGISTER DUMP AFTER EVERY ERROR/
1994                                         .LIST   BEX
1995                                         .EVEN
1996
1997
1998 002626                                 ENDMOD
```

SOFTWARE PARAMETER CODING SECTION

```
                                .TITLE GLOBAL AREAS
                                .SBTTL  GLOBAL EQUATES SECTION


002626                                  BGNMOD

                                ;++
                                ; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
                                ; ARE USED IN MORE THAN ONE TEST.
                                ;--


002626                                  EQUALS
                                ;
                                ; BIT DIFINITIONS
                                ;
100000                          BIT15== 100000
040000                          BIT14== 40000
020000                          BIT13== 20000
010000                          BIT12== 10000
004000                          BIT11== 4000
002000                          BIT10== 2000
001000                          BIT09== 1000
000400                          BIT08== 400
000200                          BIT07== 200
000100                          BIT06== 100
000040                          BIT05== 40
000020                          BIT04== 20
000010                          BIT03== 10
000004                          BIT02== 4
000002                          BIT01== 2
000001                          BIT00== 1
                                ;
001000                          BIT9==  BIT09
000400                          BIT8==  BIT08
000200                          BIT7==  BIT07
000100                          BIT6==  BIT06
000040                          BIT5==  BIT05
000020                          BIT4==  BIT04
000010                          BIT3==  BIT03
000004                          BIT2==  BIT02
000002                          BIT1==  BIT01
000001                          BIT0==  BIT00
                                ;
                                ; EVENT FLAG DEFINITIONS
                                ;   EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
                                ;
000040                          EF.START==      32.                     ; START COMMAND WAS ISSUED
000037                          EF.RESTART==    31.                     ; RESTART COMMAND WAS ISSUED
000036                          EF.CONTINUE==   30.                     ; CONTINUE COMMAND WAS ISSUED
000035                          EF.NEW==        29.                     ; A NEW PASS HAS BEEN STARTED
000034                          EF.PWR==        28.                     ; A POWER-FAIL/POWER-UP OCCURRED
                                ;
                                ;
                                ; PRIORITY LEVEL DEFINITIONS
                                ;
000340                          PRI07== 340
```

```
GLOBAL EQUATES SECTION

                000300                  PRI06== 300
                000240                  PRI05== 240
                000200                  PRI04== 200
                000140                  PRI03== 140
                000100                  PRI02== 100
                000040                  PRI01== 40
                000000                  PRI00== 0
                                        ;
                                        ;OPERATOR FLAG BITS
                                        ;
                000004                  EVL==         4
                000010                  LOT==        10
                000020                  ADR==        20
                000040                  IDU==        40
                000100                  ISR==       100
                000200                  UAM==       200
                000400                  BOE==       400
                001000                  PNT==      1000
                002000                  PRI==      2000
                004000                  IXE==      4000
                010000                  IBE==     10000
                020000                  IER==     20000
                040000                  LOE==     40000
                100000                  HOE==    100000
   2074
   2075                                 ; DRX11-C SCR bit definitions
   2076
   2077         100000                         ATT     == 100000       ; attention
   2078         040000                         NEX     == 40000        ; non existant memory
   2079         020000                         OFF     == 20000        ; off line
   2080         010000                         WAIT    == 10000        ; wait
   2081         004000                         STAT3   == 4000         ; status 3
   2082         002000                         STAT2   == 2000         ; status 2
   2083         001000                         STAT1   == 1000         ; status 1
   2084         000400                         STAT0   == 400          ; status 0
   2085         000200                         RDY     == 200          ; ready
   2086         000100                         IE      == 100          ; interrupt enable
   2087         000040                         RES     == 40           ; reset
   2088         000020                         DCH     == 20           ; direction change
   2089         000010                         FNCT3   == 10           ; function 3
   2090         000004                         FNCT2   == 4            ; function 2
   2091         000002                         FNCT1   == 2            ; function 1
   2092         000001                         FNCT0   == 1            ; function 0
   2093
   2094                                 ; DRX11-C COR bit definitions
   2095
   2096         100000                         BUF     == 100000       ; buffer mark
   2097         040000                         INOUT   == 40000        ; transfer direction
   2098         020000                         RUN     == 20000        ; DMA enable
   2099         010000                         MAINT   == 10000        ; maintenance
   2100         004000                         BURST   == 4000         ; burst enable
   2101         002400                         ACO     == 2400         ; address counter
   2102         002000                         WCO     == 2000         ; word counter
   2103         001400                         WCR2    == 1400         ; word count register 2
   2104         001000                         BAR2    == 1000         ; bus address register 2
   2105         000400                         WCR1    == 400          ; word count register 1
   2106         000000                         BAR1    == 0            ; bus address register 1
```

GLOBAL EQUATES SECTION

```
2107    000100                          ALT     == 100          ; alternate buffer mode
2108
2109                                ; AAF01-A Register subaddresses
2110
2111    000000                          DBF     == 0            ; data buffer/FIFO
2112    010000                          ACS     == 10000        ; command and status register
2113    020000                          CTA     == 20000        ; control table address register
2114    030000                          CWR     == 30000        ; control word register
2115    040000                          PCR     == 40000        ; programmable clock register
2116
2117                                ; ACS Register bit definitions
2118
2119    004000                          UNI     == 4000         ; unipolar operation
2120    002000                          CMP     == 2000         ; comparator
2121    001000                          MNT     == 1000         ; maintenance
2122    000400                          COUT    == 400          ; command output
2123    000200                          EOC     == 200          ; end of conversion
2124    000100                          DBE     == 100          ; data buffer empty
2125    000040                          SBR     == 40           ; seqence break
2126    000020                          DSR     == 20           ; data system reset
2127    000010                          SBE     == 10           ; sequence break enable
2128    000004                          ECE     == 4            ; external clock enable
2129    000002                          MET     == 2            ; memory transfer
2130    000001                          GO      == 1            ; go (start conversions)
```

GLOBAL MACROS SECTION

```
                        .SBTTL  GLOBAL MACROS SECTION
                        ;++
                        ; Macros to save or restore registers.
                        ;
                        ; Examples :
                        ;
                        ;       PUSH$   <R0,R1,R2,R3,R4,R5>
                        ;       POP$    <R5,R4,R3,R2,R1,R0>
                        ;--

                        .MACRO  PUSH$   args
                        .irp    arg,<args>
                                MOV     arg,-(SP)
                        .endr
                        .ENDM   PUSH$

                        .MACRO  POP$    args
                        .irp    arg,<args>
                                MOV     (SP)+,arg
                        .endr
                        .ENDM   POP$
```

GLOBAL DATA SECTION

```
                                .SBTTL  GLOBAL DATA SECTION

                                ;++
                                ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
                                ; IN MORE THAN ONE TEST.
                                ;--

002626  000000          DRXSCR::        .WORD   0       ; DRX11-C SCR address for current uut
002630  000000          DRXCOR::        .WORD   0       ; DRX11-C COR address for current uut
002632  000000          DRXADR::        .WORD   0       ; DRX11-C ADR address for current uut
002634  000000          DRXDBR::        .WORD   0       ; DRX11-C DBR address for current uut
002636  000000          DRXVEC::        .WORD   0       ; DRX11-C vector address for current uut

002640  000000          ADFSCR::        .WORD   0       ; ADF's DRX11-C SCR address for current uut
002642  000000          ADFCOR::        .WORD   0       ; ADF's DRX11-C COR address for current uut
002644  000000          ADFADR::        .WORD   0       ; ADF's DRX11-C ADR address for current uut
002646  000000          ADFDBR::        .WORD   0       ; ADF's DRX11-C DBR address for current uut

002650  164210          ADFADD::        .WORD   164210  ; ADF01 address - unit 0
002652  164210                          .WORD   164210  ; ADF01 address - unit 1
002654  164210                          .WORD   164210  ; ADF01 address - unit 2
002656  164210                          .WORD   164210  ; ADF01 address - unit 3
002660  164210                          .WORD   164210  ; ADF01 address - unit 4
002662  164210                          .WORD   164210  ; ADF01 address - unit 5
002664  164210                          .WORD   164210  ; ADF01 address - unit 6
002666  164210                          .WORD   164210  ; ADF01 address - unit 7
002670  164210                          .WORD   164210  ; ADF01 address - unit 8
002672  164210                          .WORD   164210  ; ADF01 address - unit 9
002674  164210                          .WORD   164210  ; ADF01 address - unit 10
002676  164210                          .WORD   164210  ; ADF01 address - unit 11
002700  164210                          .WORD   164210  ; ADF01 address - unit 12
002702  164210                          .WORD   164210  ; ADF01 address - unit 13
002704  164210                          .WORD   164210  ; ADF01 address - unit 14
002706  164210                          .WORD   164210  ; ADF01 address - unit 15

002710  100000          ECNT::          .WORD   100000  ; error count for uut 0 - bit 15 is set to
002712  100000                          .WORD   100000  ; error count for uut 1 - flag not tested.
002714  100000                          .WORD   100000  ; error count for uut 2
002716  100000                          .WORD   100000  ; error count for uut 3
002720  100000                          .WORD   100000  ; error count for uut 4
002722  100000                          .WORD   100000  ; error count for uut 5
002724  100000                          .WORD   100000  ; error count for uut 6
002726  100000                          .WORD   100000  ; error count for uut 7
002730  100000                          .WORD   100000  ; error count for uut 8
002732  100000                          .WORD   100000  ; error count for uut 9
002734  100000                          .WORD   100000  ; error count for uut 10
002736  100000                          .WORD   100000  ; error count for uut 11
002740  100000                          .WORD   100000  ; error count for uut 12
002742  100000                          .WORD   100000  ; error count for uut 13
002744  100000                          .WORD   100000  ; error count for uut 14
002746  100000                          .WORD   100000  ; error count for uut 15

002750                  DROPED::        .BLKB   16.     ; unit dropped flags

002770  000000          NXMFLG::        .WORD   0       ; set if nxm trap occurs
002772  000000          INTFLG::        .WORD   0       ; set by an interrupt to routines INT or INT1
002774                  PFLAG1::        .BLKB   16.     ; flags to control "unipolar/bipolar" printout
```

GLOBAL DATA SECTION

```
2212
2213 003014  000000                QFLAG1::        .WORD   0       ; after start command
2214 003016  000000                QFLAG2::        .WORD   0       ; flag to control questions after start command
2215                                                               ; flag to control questions after start,
2216                                                               ; restart or continue command
2217 003020  000000                GOOD::          .WORD   0       ; expected contents
2218 003022  000000                BAD::           .WORD   0       ; actual contents
2219
2220 003024  000000                PADD::          .WORD   0       ; address of prompt for decimal input routine
2221
2222 003026  000000                MODE::          .WORD   0       ; mode for digital/analogue conversion routines
2223
2224 003030  000000                STAFLG::        .WORD   0       ; flag = 1 after start command, cleared by each test
2225 003032  000000                ITRCNT::        .WORD   0       ; iteration counter
2226
2227 003034  007777                FVAL::          .WORD   7777    ; first value for output in tests 25 and 26
2228 003036  000000                SVAL::          .WORD   0       ; second value for output in tests 25 and 26
2229
2230 003040                        BUFOUT::        .BLKW   4096.   ; buffer for DMA output
2231 023040  000000                BUFIN::         .WORD   0       ; address of buffer for DMA input
```

GLOBAL DATA SECTION

```
2233          000012                          .RADIX  10
2234                                          .NLIST  BEX
2235
2236                                  ; Analogue/digital conversion tables used by routines DACON and ADCON.
2237
2238
2239
2240                                  ; Unipolar table - mode 0 (0-10v)
2241
2242                                  ; Bits    11   10    9    8    7    6    5    4    3    2    1    0
2243
2244 023042  011610  004704  002342  VUPTAB:: 5000,2500,1250, 625, 312, 156,  78,  39,  19,   9,   4,   2 ; mV
2245 023072  000000  000000  000000             0,   0,   0,   0, 500, 250, 125,  63, 531, 766, 883, 441 ; uV
2246
2247
2248
2249                                  ; Bipolar table - mode 1 (-10 - +10v)
2250
2251                                  ; Bits    11   10    9    8    7    6    5    4    3    2    1    0
2252
2253 023122  023420  011610  004704  VBPTAB::10000,5000,2500,1250, 625, 312, 156,  78,  39,  19,   9,   4 ; mV
2254 023152  000000  000000  000000             0,   0,   0,   0,   0, 500, 250, 125,  63, 531, 766, 883 ; uV
2255
2256
2257          000010                          .RADIX  8
2258                                          .LIST   BEX
```

GLOBAL TEXT SECTION

```
                                .SBTTL  GLOBAL TEXT SECTION

                                ;++
                                ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
                                ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
                                ; MORE THAN ONE TEST.
                                ;--

                                        .NLIST  BEX

                                ;
                                ; NAMES OF DEVICES SUPPORTED BY PROGRAM
                                ;
023202                                  DEVTYP  <AAF01-A>
023202                                                                  L$DVTYP::
023202  101     101     106                                                     .ASCIZ  /AAF01-A/
                                                                                .EVEN

                                ; TEST DESCRIPTION
                                ;
023212                                  DESCRIPT        <AAF01-A DIAGNOSTIC>
023212                                                                  L$DESC::
023212  101     101     106                                                     .ASCIZ  /AAF01-A DIAGNOSTIC/
                                                                                .EVEN

023236  106     111     122     FVALQ:: .ASCIZ  /FIRST VALUE IN MILLIVOLTS/             ; prompts for values
023270  123     105     103     SVALQ:: .ASCIZ  /SECOND VALUE IN MILLIVOLTS/            ; used in tests
023323  045     101     122     RNDOUT::.ASCIZ  /%AROUNDED TO /                         ; 25 to 27
023341  045     101     040     OCTOUT::.ASCIZ  /%A MILLIVOLTS - OCTAL VALUE %05%N/

                                        .LIST   BEX
                                        .EVEN
```

GLOBAL ERROR REPORT SECTION

```
                                .SBTTL  GLOBAL ERROR REPORT SECTION

                                ;++
                                ; THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS
                                ; USED BY MORE THAN ONE TEST TO OUTPUT ADDITIONAL ERROR INFORMATION.  PRINTB
                                ; (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.
                                ;--


                                        BGNMSG  ECHK
023404                                                                  ECHK::
2314 023404  004737  024760             JSR     PC,CHKMAX               ; check for too many errors
2315 023410  004737  025124             JSR     PC,DUMP                 ; dump registers if dumping selected
2316 023414                             ENDMSG
023414                                                                  L10004:
023414  104423                                                                  TRAP    C$MSG
2317
2318 023416                             BGNMSG  EGB
023416                                                                  EGB::
2319 023416                             PRINTB  #GOOBAD,GOOD,BAD        ; print "GOOD, BAD"
023416  013746  003022                                                          MOV     BAD,-(SP)
023422  013746  003020                                                          MOV     GOOD,-(SP)
023426  012746  024204                                                          MOV     #GOOBAD,-(SP)
023432  012746  000003                                                          MOV     #3,-(SP)
023436  010600                                                                  MOV     SP,R0
023440  104414                                                                  TRAP    C$PNTB
023442  062706  000010                                                          ADD     #10,SP
2320 023446  004737  024760             JSR     PC,CHKMAX               ; check for too many errors
2321 023452  004737  025124             JSR     PC,DUMP                 ; dump registers if dumping selected
2322 023456                             ENDMSG
023456                                                                  L10005:
023456  104423                                                                  TRAP    C$MSG
2323
2324 023460                             BGNMSG  ERDNR
023460                                                                  ERDNR::
2325 023460                             PRINTB  #EMG1,R1                ; print "REGISTER DOES NOT RESPOND"
023460  010146                                                                  MOV     R1,-(SP)
023462  012746  024234                                                          MOV     #EMG1,-(SP)
023466  012746  000002                                                          MOV     #2,-(SP)
023472  010600                                                                  MOV     SP,R0
023474  104414                                                                  TRAP    C$PNTB
023476  062706  000006                                                          ADD     #6,SP
2326 023502  004737  024760             JSR     PC,CHKMAX               ; check for too many errors
2327 023506  004737  025124             JSR     PC,DUMP                 ; dump registers if dumping selected
2328 023512                             ENDMSG
023512                                                                  L10006:
023512  104423                                                                  TRAP    C$MSG
2329
2330 023514                             BGNMSG  ECGB
023514                                                                  ECGB::
2331 023514                             PRINTB  #EMG2,R2,GOOD,BAD       ; print "CTA, GOOD, BAD"
023514  013746  003022                                                          MOV     BAD,-(SP)
023520  013746  003020                                                          MOV     GOOD,-(SP)
023524  010246                                                                  MOV     R2,-(SP)
023526  012746  024303                                                          MOV     #EMG2,-(SP)
023532  012746  000004                                                          MOV     #4,-(SP)
```

GLOBAL ERROR REPORT SECTION

```
                023536  010600                                                          MOV     SP,R0
                023540  104414                                                          TRAP    C$PNTB
                023542  062706  000012                                                  ADD     #12,SP
   2332         023546  004737  024760          JSR     PC,CHKMAX               ; check for too many errors
   2333         023552  004737  025124          JSR     PC,DUMP                 ; dump registers if dumping selected
   2334         023556                          ENDMSG
                023556                                                                  L10007:
                023556  104423                                                          TRAP    C$MSG
   2335
   2336         023560                          BGNMSG  EIGB
                023560                                                                  EIGB::
   2337         023560                          PRINTB  #EMG3,INTFLG,@DRXSCR    ; print "INTERRUPTS, GOOD, BAD"
                023560  017746  157042                                                  MOV     @DRXSCR,-(SP)
                023564  013746  002772                                                  MOV     INTFLG,-(SP)
                023570  012746  024347                                                  MOV     #EMG3,-(SP)
                023574  012746  000003                                                  MOV     #3,-(SP)
                023600  010600                                                          MOV     SP,R0
                023602  104414                                                          TRAP    C$PNTB
                023604  062706  000010                                                  ADD     #10,SP
   2338         023610  004737  024760          JSR     PC,CHKMAX               ; check for too many errors
   2339         023614  004737  025124          JSR     PC,DUMP                 ; dump registers if dumping selected
   2340         023620                          ENDMSG
                023620                                                                  L10010:
                023620  104423                                                          TRAP    C$MSG
   2341
   2342         023622                          BGNMSG  EGBCC
                023622                                                                  EGBCC::
   2343         023622                          PRINTB  #EMG4,GOOD,BAD,R1,R2    ; print "GOOD, BAD, CHAN15, CHAN0"
                023622  010246                                                          MOV     R2,-(SP)
                023624  010146                                                          MOV     R1,-(SP)
                023626  013746  003022                                                  MOV     BAD,-(SP)
                023632  013746  003020                                                  MOV     GOOD,-(SP)
                023636  012746  024443                                                  MOV     #EMG4,-(SP)
                023642  012746  000005                                                  MOV     #5,-(SP)
                023646  010600                                                          MOV     SP,R0
                023650  104414                                                          TRAP    C$PNTB
                023652  062706  000014                                                  ADD     #14,SP
   2344         023656  004737  024760          JSR     PC,CHKMAX               ; check for too many errors
   2345         023662  004737  025124          JSR     PC,DUMP                 ; dump registers if dumping selected
   2346         023666                          ENDMSG
                023666                                                                  L10011:
                023666  104423                                                          TRAP    C$MSG
   2347
   2348         023670                          BGNMSG  ESA
                023670                                                                  ESA::
   2349         023670                          PRINTB  #EMG5,R2,R3             ; print "SCR, ACS"
                023670  010346                                                          MOV     R3,-(SP)
                023672  010246                                                          MOV     R2,-(SP)
                023674  012746  024526                                                  MOV     #EMG5,-(SP)
                023700  012746  000003                                                  MOV     #3,-(SP)
                023704  010600                                                          MOV     SP,R0
                023706  104414                                                          TRAP    C$PNTB
                023710  062706  000010                                                  ADD     #10,SP
   2350         023714  004737  024760          JSR     PC,CHKMAX               ; check for too many errors
   2351         023720  004737  025124          JSR     PC,DUMP                 ; dump registers if dumping selected
   2352         023724                          ENDMSG
                023724                                                                  L10012:
```

GLOBAL ERROR REPORT SECTION

```
        023724  104423                                                          TRAP    C$MSG
  2353
  2354  023726                          BGNMSG  EAAAF
        023726                                                                  EAAAF::
  2355  023726                          PRINTB  #EMG6,R1,R2             ; print "AAF01, ADF01"
        023726  010246                                                          MOV     R2,-(SP)
        023730  010146                                                          MOV     R1,-(SP)
        023732  012746  024555                                                  MOV     #EMG6,-(SP)
        023736  012746  000003                                                  MOV     #3,-(SP)
        023742  010600                                                          MOV     SP,R0
        023744  104414                                                          TRAP    C$PNTB
        023746  062706  000010                                                  ADD     #10,SP
  2356  023752  004737  024760          JSR     PC,CHKMAX               ; check for too many errors
  2357  023756  004737  025124          JSR     PC,DUMP                 ; dump registers if dumping selected
  2358  023762                          ENDMSG
        023762                                                                  L10013:
        023762  104423                                                          TRAP    C$MSG
  2359
  2360  023764                          BGNMSG  ECHGB
        023764                                                                  ECHGB::
  2361  023764                          PRINTB  #EMG7,R1,GOOD,BAD       ; print "CHANNEL, GOOD, BAD"
        023764  013746  003022                                                  MOV     BAD,-(SP)
        023770  013746  003020                                                  MOV     GOOD,-(SP)
        023774  010146                                                          MOV     R1,-(SP)
        023776  012746  024610                                                  MOV     #EMG7,-(SP)
        024002  012746  000004                                                  MOV     #4,-(SP)
        024006  010600                                                          MOV     SP,R0
        024010  104414                                                          TRAP    C$PNTB
        024012  062706  000012                                                  ADD     #12,SP
  2362  024016  004737  024760          JSR     PC,CHKMAX               ; check for too many errors
  2363  024022  004737  025124          JSR     PC,DUMP                 ; dump registers if dumping selected
  2364  024026                          ENDMSG
        024026                                                                  L10014:
        024026  104423                                                          TRAP    C$MSG
  2365
  2366  024030                          BGNMSG  EN
        024030                                                                  EN::
  2367  024030                          PRINTB  #EMG8,ERFLA             ; print "ERROR NUMBER"
        024030  013746  032140                                                  MOV     ERFLA,-(SP)
        024034  012746  024657                                                  MOV     #EMG8,-(SP)
        024040  012746  000002                                                  MOV     #2,-(SP)
        024044  010600                                                          MOV     SP,R0
        024046  104414                                                          TRAP    C$PNTB
        024050  062706  000006                                                  ADD     #6,SP
  2368  024054  004737  024760          JSR     PC,CHKMAX               ; check for too many errors
  2369  024060  004737  025124          JSR     PC,DUMP                 ; dump registers if dumping selected
  2370  024064                          ENDMSG
        024064                                                                  L10015:
        024064  104423                                                          TRAP    C$MSG
  2371
  2372  024066                          BGNMSG  EGBDEC
        024066                                                                  EGBDEC::
  2373  024066                          RFLAGS  R0                      ; don't print anything if
        024066  104421                                                          TRAP    C$RFLA
  2374  024070  032700  020000          BIT     #IER,R0                 ; error reporting is inhibited
  2375  024074  001036                  BNE     10$                     ; ...
  2376  024076                          PRINTB  #EMG9A                  ; print "GOOD:"
```

GLOBAL ERROR REPORT SECTION

```
        024076  012746  024704                                                  MOV     #EMG9A,-(SP)
        024102  012746  000001                                                  MOV     #1,-(SP)
        024106  010600                                                          MOV     SP,R0
        024110  104414                                                          TRAP    C$PNTA
        024112  062706  000004                                                  ADD     #4,SP
  2377  024116  004737  027640                  JSR     PC,DECOUT       ; print "xxxxx.xxx"
  2378  024122                                  PRINTB  #EMG9B          ; "MILLIVOLTS, BAD:"
        024122  012746  024714                                                  MOV     #EMG9B,-(SP)
        024126  012746  000001                                                  MOV     #1,-(SP)
        024132  010600                                                          MOV     SP,R0
        024134  104414                                                          TRAP    C$PNTB
        024136  062706  000004                                                  ADD     #4,SP
  2379  024142  010301                          MOV     R3,R1           ; get BAD values
  2380  024144  010402                          MOV     R4,R2           ; in R1 and R2
  2381  024146  004737  027640                  JSR     PC,DECOUT       ; print "xxxxx.xxx"
  2382  024152                                  PRINTB  #EMG9C          ; "MILLIVOLTS"
        024152  012746  024740                                                  MOV     #EMG9C,-(SP)
        024156  012746  000001                                                  MOV     #1,-(SP)
        024162  010600                                                          MOV     SP,R0
        024164  104414                                                          TRAP    C$PNTB
        024166  062706  000004                                                  ADD     #4,SP
  2383  024172  004737  024760          10$:    JSR     PC,CHKMAX       ; check for too many errors
  2384  024176  004737  025124                  JSR     PC,DUMP         ; dump registers if dumping selected
  2385  024202                                  ENDMSG
        024202                                                          L10016:
        024202  104423                                                         TRAP    C$MSG
  2386
  2387                                          .NLIST BEX
  2388  024204      045     101     107  GOOBAD: .ASCIZ  /%AGOOD:%O6%A, BAD:%O6%N/
  2389  024234      045     101     122  EMG1:   .ASCIZ  /%AREGISTER AT %O6%A DOES NOT RESPOND%N/
  2390  024303      045     101     103  EMG2:   .ASCIZ  /%ACTA: %O6%A, GOOD:%O6%A, BAD:%O6%N/
  2391  024347      045     101     116  EMG3:   .ASCIZ  /%ANUMBER OF INTERRUPTS: %D2%A, SCR GOOD: 100600, BAD: %O6%N/
  2392  024443      045     101     107  EMG4:   .ASCIZ  /%AGOOD:%O6%A, BAD:%O6%A, CHAN15:%O4%A, CHAN0:%O4%N/
  2393  024526      045     101     123  EMG5:   .ASCIZ  /%ASCR:%O6%A, ACS:%O6%N/
  2394  024555      045     101     101  EMG6:   .ASCIZ  /%AADF01:%T%A, AAF01-A:%T%N/
  2395  024610      045     101     103  EMG7:   .ASCIZ  /%ACHANNEL:%D2%A, GOOD:%O4%A, BAD:%O4%N/
  2396  024657      045     101     105  EMG8:   .ASCIZ  /%AERROR NUMBER:%D1%N/
  2397  024704      045     101     107  EMG9A:  .ASCIZ  /%AGOOD:/
  2398  024714      045     101     040  EMG9B:  .ASCIZ  /%A MILLIVOLTS, BAD:/
  2399  024740      045     101     040  EMG9C:  .ASCIZ  /%A MILLIVOLTS%N/
  2400                                          .LIST BEX
  2401                                          .EVEN
  2402
```

```
                                .SBTTL  GLOBAL SUBROUTINES SECTION

                                ;++
                                ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
                                ; THAT ARE USED IN MORE THAN ONE TEST.
                                ;--

                                ; Subroutine CHKMAX - Error count checking routine.
                                ;
                                ;++
                                ; Functional description :
                                ;
                                ;       Subroutine to update unit error count. If the program is looping
                                ;       on an error, the subroutine does nothing. Otherwise, the error
                                ;       count for the unit is incremented. If the error count exceeds 5
                                ;       and the user flag EVL has been selected and the flag IDU is not
                                ;       selected, the unit is dropped from the test cycle.
                                ;
                                ; Inputs :
                                ;
                                ;       None.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       L$LUN contains the number of the unit currently being tested.
                                ;       ECNT is the address of the error count for unit 0.
                                ;
                                ; Outputs :
                                ;
                                ;       None.
                                ;
                                ; Implicit outputs :
                                ;
                                ;       The error count for the logical unit being tested is
                                ;       incremented if the program is not looping.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       None.
                                ;
                                ; Functional side effects :
                                ;
                                ;       If the error count exceeds 5 and the user EVL flag is selected,
                                ;       and the 'loop on test' and 'inhibit dropping of units' flags are
                                ;       not selected, the unit will be dropped from testing.
                                ;
                                ; Calling sequence :
                                ;
                                ;       JSR PC,CHKMAX
                                ;
                                ;--

024760                          CHKMAX::INLOOP                  ; looping on error?
024760  104420                          TRAP    C$INLP
024762                                  BCOMPLETE 10$           ; if yes, exit
024762  103436                          BCS     10$
```

```
GLOBAL SUBROUTINES SECTION

 2459 024764  013700  002074                 MOV     L$LUN,R0        ; get current unit
 2460 024770  006300                         ASL     R0              ; convert to error count offset
 2461 024772  005260  002710                 INC     ECNT(R0)        ; update the error count
 2462 024776  026027  002710  000005         CMP     ECNT(R0),#5     ; too many errors?
 2463 025004  003425                         BLE     10$             ; if not, jump
 2464
 2465 025006                                 RFLAGS  R0              ; get operator flags
      025006  104421                                                            TRAP    C$RFLA
 2466 025010  032700  000040                 BIT     #IDU,R0         ; is dropping inhibited?
 2467 025014  001021                         BNE     10$             ; if yes, exit
 2468 025016  032700  000004                 BIT     #EVL,R0         ; evaluate flag selected ?
 2469 025022  001416                         BEQ     10$             ; if not, exit
 2470
 2471 025024                                 PRINTF  #NERRS,L$LUN    ; 'TOO MANY ERRORS'
      025024  013746  002074                                                    MOV     L$LUN,-(SP)
      025030  012746  025062                                                    MOV     #NERRS,-(SP)
      025034  012746  000002                                                    MOV     #2,-(SP)
      025040  010600                                                            MOV     SP,R0
      025042  104417                                                            TRAP    C$PNTF
      025044  062706  000006                                                    ADD     #6,SP
 2472 025050                                 DODU    L$LUN           ; drop the unit
      025050  013700  002074                                                    MOV     L$LUN,R0
      025054  104451                                                            TRAP    C$DODU
 2473
 2474 025056                                 DOCLN                   ; end the subpass
      025056  104444                                                            TRAP    C$DCLN
 2475
 2476 025060  000207                 10$:    RTS     PC
 2477
 2478                                        .NLIST  BEX
 2479 025062     045     116     045 NERRS:  .ASCIZ  /%N%AMORE THAN 5 ERRORS ON UNIT%D2/
 2480                                        .LIST   BEX
 2481                                        .EVEN
```

GLOBAL SUBROUTINES SECTION

```
                                ; Subroutine DUMP - Register dump routine.
                                ;
                                ;**
                                ; Functional description :
                                ;
                                ;       Subroutine to print a dump of the DRX11-C and AAF01-A registers
                                ;       following an error printout. A dump is made only if RDUMP is
                                ;       set to 1 using the startup software questions.
                                ;
                                ; Inputs :
                                ;
                                ;       None.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       RDUMP - if 0, no dump is made.
                                ;
                                ; Outputs :
                                ;
                                ;       A dump of the AAF01-A and DRX11-C registers is printed.
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       None.
                                ;
                                ; Functional side effects :
                                ;
                                ;       None.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSR PC,DUMP
                                ;
                                ;--

025124  005737  002236          DUMP::  TST     RDUMP                   ; is dumping selected ?
025130  001001                          BNE     10$                     ; if yes, branch
025132  000207                          RTS     PC                      ; else return

025134  017737  155474  025556  10$:    MOV     @DRXDBR,RS+14           ; save the DBR
025142  012777  010000  155464          MOV     #ACS,@DRXDBR            ; address the ACS
025150  017737  155460  025542          MOV     @DRXDBR,RS              ; and save its contents
025156  012777  020000  155450          MOV     #CTA,@DRXDBR            ; address the CTA
025164  017737  155444  025544          MOV     @DRXDBR,RS+2            ; and save its contents
025172  012777  030000  155434          MOV     #CWR,@DRXDBR            ; address the CWR
025200  017737  155430  025546          MOV     @DRXDBR,RS+4            ; and save its contents
025206                                  PRINTB  #DMP1,RS,RS+2,RS+4      ; dump ACS, CTA and CWR
  025206  013746  025546                                                        MOV     RS+4,-(SP)
  025212  013746  025544                                                        MOV     RS+2,-(SP)
  025216  013746  025542                                                        MOV     RS,-(SP)
  025222  012746  025574                                                        MOV     #DMP1,-(SP)
  025226  012746  000004                                                        MOV     #4,-(SP)
  025232  010600                                                                MOV     SP,R0
```

```
GLOBAL SUBROUTINES SECTION

        025234  104414                                                          TRAP    C$PNTB
        025236  062706  000012                                                  ADD     #12,SP
  2534
  2535  025242  017737  155360  025550          MOV     @DRXSCR,RS+6            ; save the SCR
  2536  025250  017737  155354  025552          MOV     @DRXCOR,RS+10           ; COR
  2537  025256  017737  155350  025554          MOV     @DRXADR,RS+12           ; and ADR
  2538  025264                                  PRINTB  #DMP2,RS+6,RS+10,RS+12,RS+14 ; dump SCR, COR, ADR and DBR
        025264  013746  025556                                                  MOV     RS+14,-(SP)
        025270  013746  025554                                                  MOV     RS+12,-(SP)
        025274  013746  025552                                                  MOV     RS+10,-(SP)
        025300  013746  025550                                                  MOV     RS+6,-(SP)
        025304  012746  025652                                                  MOV     #DMP2,-(SP)
        025310  012746  000005                                                  MOV     #5,-(SP)
        025314  010600                                                          MOV     SP,R0
        025316  104414                                                          TRAP    C$PNTB
        025320  062706  000014                                                  ADD     #14,SP
  2539
  2540  025324  012777  000000  155276          MOV     #BAR1,@DRXCOR           ; address BAR1
  2541  025332  017737  155274  025560          MOV     @DRXADR,RS+16           ; and save its contents
  2542  025340  012777  000400  155262          MOV     #WCR1,@DRXCOR           ; address WCR1
  2543  025346  017737  155260  025562          MOV     @DRXADR,RS+20           ; and save its contents
  2544  025354  012777  001000  155246          MOV     #BAR2,@DRXCOR           ; address BAR2
  2545  025362  017737  155244  025564          MOV     @DRXADR,RS+22           ; and save its contents
  2546  025370  012777  001400  155232          MOV     #WCR2,@DRXCOR           ; address WCR2
  2547  025376  017737  155230  025566          MOV     @DRXADR,RS+24           ; and save its contents
  2548  025404                                  PRINTB  #DMP3,RS+16,RS+20,RS+22,RS+24 ; dump BAR1, WCR1, BAR2 and WCR2
        025404  013746  025566                                                  MOV     RS+24,-(SP)
        025410  013746  025564                                                  MOV     RS+22,-(SP)
        025414  013746  025562                                                  MOV     RS+20,-(SP)
        025420  013746  025560                                                  MOV     RS+16,-(SP)
        025424  012746  025744                                                  MOV     #DMP3,-(SP)
        025430  012746  000005                                                  MOV     #5,-(SP)
        025434  010600                                                          MOV     SP,R0
        025436  104414                                                          TRAP    C$PNTB
        025440  062706  000014                                                  ADD     #14,SP
  2549
  2550  025444  012777  002000  155156          MOV     #WC0,@DRXCOR            ; address WC0
  2551  025452  017737  155154  025570          MOV     @DRXADR,RS+26           ; and save its contents
  2552  025460  012777  002400  155142          MOV     #AC0,@DRXCOR            ; address AC0
  2553  025466  017737  155140  025572          MOV     @DRXADR,RS+30           ; and save its contents
  2554  025474                                  PRINTB  #DMP4,RS+26,RS+30       ; dump  C0 and AC0
        025474  013746  025572                                                  MOV     RS+30,-(SP)
        025500  013746  025570                                                  MOV     RS+26,-(SP)
        025504  012746  026036                                                  MOV     #DMP4,-(SP)
        025510  012746  000003                                                  MOV     #3,-(SP)
        025514  010600                                                          MOV     SP,R0
        025516  104414                                                          TRAP    C$PNTB
        025520  062706  000010                                                  ADD     #10,SP
  2555
  2556  025524  013777  025552  155076          MOV     RS+10,@DRXCOR           ; restore the COR
  2557  025532  013777  025556  155074          MOV     RS+14,@DRXDBR           ; and the DBR
  2558
  2559  025540  000207                          RTS     PC
  2560
  2561  025542  000000                  RS:     .WORD   0                       ; store for ACS contents
  2562  025544  000000                          .WORD   0                       ; store for CTA contents
  2563  025546  000000                          .WORD   0                       ; store for CWR contents
```

GLOBAL SUBROUTINES SECTION

```
 2564 025550  000000                          .WORD   0                       ; store for SCR contents
 2565 025552  000000                          .WORD   0                       ; store for COR contents
 2566 025554  000000                          .WORD   0                       ; store for ADR contents
 2567 025556  000000                          .WORD   0                       ; store for DBR contents
 2568 025560  000000                          .WORD   0                       ; store for BAR1 contents
 2569 025562  000000                          .WORD   0                       ; store for WCR1 contents
 2570 025564  000000                          .WORD   0                       ; store for BAR2 contents
 2571 025566  000000                          .WORD   0                       ; store for WCR2 contents
 2572 025570  000000                          .WORD   0                       ; store for WCO contents
 2573 025572  000000                          .WORD   0                       ; store for ACO contents
 2574
 2575                                         .NLIST  BEX
 2576 025574     045     101     101  DMP1:   .ASCIZ  /%AAAF01-A  ACS: %06%A, CTA: %06%A, CWR: %06%N/
 2577 025652     045     101     104  DMP2:   .ASCIZ  /%ADRX11-C  SCR: %06%A, COR: %06%A, ADR: %06%A, DBR: %06%N/
 2578 025744     045     101     040  DMP3:   .ASCIZ  /%A          BAR1:%06%A, WCR1:%06%A, BAR2:%06%A, WCR2:%06%N/
 2579 026036     045     101     040  DMP4:   .ASCIZ  /%A          WCO: %06%A, ACO: %06%N/
 2580                                         .LIST   BEX
 2581                                         .EVEN
 2582
```

GLOBAL SUBROUTINES SECTION

```
                                        ; Subroutines WT25M, WT500 AND WT25 - Delay routines.
                                        ;
                                        ;**
                                        ; Functional description :
                                        ;
                                        ;       Subroutine to wait for 25 milliseconds, 500 microseconds or 25
                                        ;       microseconds.
                                        ;
                                        ;       Note. Because of the small number of program wait loops used for
                                        ;             the 25 microsecond counter, the accuracy of the WT25 routine
                                        ;             is low. The delay may last up to 50 microseconds on some
                                        ;             slow processors.
                                        ;
                                        ; Inputs :
                                        ;
                                        ;       None.
                                        ;
                                        ; Implicit inputs :
                                        ;
                                        ;       The variables CNT25M, CNT500, and CNT25 must have been set up by
                                        ;       routine SETCLK.
                                        ;
                                        ; Outputs :
                                        ;
                                        ;       None.
                                        ;
                                        ; Implicit outputs :
                                        ;
                                        ;       None.
                                        ;
                                        ; Subordinate routines used :
                                        ;
                                        ;       None.
                                        ;
                                        ; Functional side effects :
                                        ;
                                        ;       None.
                                        ;
                                        ; Calling sequence :
                                        ;
                                        ;       JSR     PC,WT25M        ; wait for 25 milliseconds
                                        ;    OR JSR     PC,WT500        ; wait for 500 microseconds
                                        ;    OR JSR     PC,WT25         ; wait for 25 microseconds
                                        ;
                                        ;--

026100  013700  026126          WT25M:: MOV     CNT25M,R0       ; get 25 millisecond wait count
026104  000405                          BR      WT              ;

026106  013700  026130          WT500:: MOV     CNT500,R0       ; get 500 microsecond wait count
026112  000402                          BR      WT              ;

026114  013700  026132          WT25::  MOV     CNT25,R0        ; get 25 microsecond wait count

026120  005300                  WT:     DEC     R0              ; all done?
026122  001376                          BNE     WT              ; if not, wait some more
026124  000207                          RTS     PC              ; else return
```

GLOBAL SUBROUTINES SECTION

```
   2641
   2642 026126  000000                CNT25M::.WORD   0          ; counter for 25 millisecond delay
   2643 026130  000000                CNT500::.WORD   0          ; counter for 500 microsecond delay
   2644 026132  000000                CNT25:: .WORD   0          ; counter for 25 microsecond delay
   2645
```

GLOBAL SUBROUTINES SECTION

```
                                ; Subroutine CRLF - Routine to print carriage return, line feed.
                                ;
                                ;++
                                ; Functional description :
                                ;
                                ;       Prints a carriage return and line feed.
                                ;
                                ; Inputs :
                                ;
                                ;       None.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       None.
                                ;
                                ; Outputs :
                                ;
                                ;       A carriage return and line feed are printed.
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       Supervisor PRINTF macro.
                                ;
                                ; Functional side effects :
                                ;
                                ;       None.
                                ;
                                ; Calling sequence :
                                ;
                                ;       JSR     PC,CRLF
                                ;
                                ;--
                                CRLF::
                                        PRINTF  #LF
026134  012746  026156                                          MOV     #LF,-(SP)
026140  012746  000001                                          MOV     #1,-(SP)
026144  010600                                                  MOV     SP,R0
026146  104417                                                  TRAP    C$PNTF
026150  062706  000004                                          ADD     #4,SP
026154  000207                         RTS     PC

026156      045     116     000  LF:    .ASCIZ  /%N/
                                        .EVEN
```

GLOBAL SUBROUTINES SECTION

```
                                ; Subroutine WRDY - Subroutine to wait for operator ready
                                ;
                                ;++
                                ; Functional description :
                                ;
                                ;       This prints a message for the operator to type 'carriage return'
                                ;       to continue. The routine is normally used to allow a message to
                                ;       be read before proceding.
                                ;
                                ;       If manual intervention is not allowed, the routine does nothing.
                                ;
                                ; Inputs :
                                ;
                                ;       None.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       None.
                                ;
                                ; Outputs :
                                ;
                                ;       "TYPE 'CARRIAGE RETURN' TO CONTINUE OR 'CONRTOL C' TO ABORT"
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       Supervisor GMANID macro.
                                ;
                                ; Functional side effects :
                                ;
                                ;       None.
                                ;
                                ; Calling sequence :
                                ;
                                ;       JSR     PC,WRDY
                                ;
                                ;--
                                WRDY::
                                        MANUAL                                  ; is manual intervention allowed ?
026162  104450                                                                          TRAP    C$MANI
                                        BNCOMPLETE 10$                          ; if not, exit
026164  103010                                                                          BCC     10$
                                        GMANID  WRDY1,WFLG,A,377,0,1,YES        ; 'type return to continue'
026166  104443                                                                          TRAP    C$GMAN
026170  000406                                                                          BR      10000$
026172  026210                                                                          .WORD   WFLG
026174  000152                                                                          .WORD   T$CODE
026176  026212                                                                          .WORD   WRDY1
026200  000377                                                                          .WORD   377
026202  000000                                                                          .WORD   T$LOLIM
026204  000001                                                                          .WORD   T$HILIM
026206                                                                          10000$:
026206  000207                  10$:    RTS     PC
```

GLOBAL SUBROUTINES SECTION

```
   2737
   2738 026210  000000                  WFLG:   .WORD   0                       ; flag for input
   2739
   2740                                         .NLIST  BEX
   2741 026212     124     131     120  WRDY1:  .ASCIZ  /TYPE "CARRIAGE RETURN" TO CONTINUE OR "CONTROL C" TO ABORT./
   2742                                         .LIST   BEX
   2743                                         .EVEN
```

GLOBAL SUBROUTINES SECTION

```
                                ; Subroutine INSERT - Subroutine to force error printouts
                                ;
                                ;++
                                ; Functional description :
                                ;
                                ;       This subroutine can be used to force printout of error messages
                                ;       for quality checking. If the instruction at location "INSERT" is
                                ;       equal to 402, the branch instruction following the subroutine
                                ;       call is skipped over, causing the error message to be printed.
                                ;       If the location contains 401, the address of the subroutine call
                                ;       is compared with that of the last call and, if it has not changed,
                                ;       the message is not printed.
                                ;
                                ; Inputs :
                                ;
                                ;       Location "INSERT"  - if 400 the routine does nothing.
                                ;                            if 401 error messages are printed once.
                                ;                            if 402 error messages are always printed.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       See calling sequence.
                                ;
                                ; Outputs :
                                ;
                                ;       LASTFA - if the error message is to be printed once only, LASTFA
                                ;                is loaded with this subroutine return address.
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       None.
                                ;
                                ; Functional side effects :
                                ;
                                ;       R0 is destroyed.
                                ;
                                ; Calling sequence :
                                ;
                                ;       A one word branch instruction must follow the subroutine call
                                ;       before the error print call.
                                ;
                                ;       Eg.     CMP     BAD,GOOD        ; register correct ?
                                ;               CALL    INSERT          ; skip branch if error insert selected
                                ;               BEQ     10$             ; branch if register correct
                                ;               ERROR                   ; else print out error message
                                ;
                                ;--
                                
026306                          INSERT::
026306  000400                          BR      10$                     ; 400=10$, 401=20$, 402=30$
026310  000207                  10$:    RTS     PC                      ; don't force error printing
026312  000401                  20$:    BR      40$                     ; print each error once
026314  000412                  30$:    BR      50$                     ; print every error message
```

GLOBAL SUBROUTINES SECTION

```
 2802 026316  013700  026350          40$:    MOV     LASTFA,R0               ; get last fault address
 2803 026322  011637  026350                  MOV     (SP),LASTFA             ; save new fault address
 2804 026326  021600                          CMP     (SP),R0                 ; are they the same ?
 2805 026330  001004                          BNE     50$                     ; if not, print the error
 2806 026332  117600  000000                  MOVB    @(SP),R0                ; else get the return branch offset
 2807 026336  006300                          ASL     R0                      ; convert to a byte offset
 2808 026340  060016                          ADD     R0,(SP)                 ; skip over the error printout
 2809 026342  062716  000002          50$:    ADD     #2,(SP)                 ; add 2 to the return address
 2810 026346  000207                          RTS     PC                      ; and return
 2811
 2812 026350  000000                  LASTFA: .WORD   0                       ; address of routine call
 2813
```

GLOBAL SUBROUTINES SECTION

```
                                ; Subroutine DACON - Digital to analogue conversion routine.
                                ;
                                ;++
                                ; Functional description :
                                ;
                                ;       This converts a 12 bit digital pattern into a 2 word analogue
                                ;       output value.
                                ;
                                ; Inputs :
                                ;
                                ;       MODE : 0 = unipolar (0 to 10 volts)     complementary binary coded
                                ;              1 = bipolar  (-10 to +10 volts)  complementary offset binary coded
                                ;
                                ;       R1   : 12 bit input pattern.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       VITAB and ITAB : digital/analogue conversion tables.
                                ;
                                ; Outputs :
                                ;
                                ;       R1 - millivolts
                                ;
                                ;       R2 - microvolts
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       None.
                                ;
                                ; Functional side effects :
                                ;
                                ;       None.
                                ;
                                ; Calling sequence:
                                ;
                                ;  Eg.  MOV     #1,MODE                 ; bipolar conversion
                                ;       MOV     #7777,R1                ; all bits set
                                ;       JSR     PC,DACON
                                ;
                                ;--
                                DACON::
                                        MOV     R3,-(SP)                ; save R3
                                        MOV     R4,-(SP)                ; and R4
                                        COM     R1                      ; complememt the data
                                        MOV     #VUPTAB,R0              ; get conversion table for mode 0
                                        TST     MODE                    ; mode 0 ?
                                        BEQ     20$                     ; if yes, branch
                                        MOV     #VBPTAB,R0              ; get conversion table for mode 1
                                
                                20$:    MOV     (R0),R4                 ; save high bit value
                                        MOV     R1,R3                   ; save the bit pattern
                                        CLR     R1                      ; clear the output registers
```

GLOBAL SUBROUTINES SECTION

```
2872 026404  005002                        CLR     R2                      ;
2873 026406  006303                        ASL     R3                      ; shift out unused bits
2874 026410  006303                        ASL     R3                      ; ( bits 15 - 12 )
2875 026412  006303                        ASL     R3                      ;
2876 026414  006303                        ASL     R3                      ;
2877
2878 026416  006303                 50$:   ASL     R3                      ; test a bit
2879 026420  103011                        BCC     70$                     ; if clear, branch
2880 026422  066002  000030                ADD     24.(R0),R2              ; else add in low value
2881 026426  020227  001750                CMP     R2,#1000.               ; overflow of low word ?
2882 026432  002403                        BLT     60$                     ; if not, branch
2883 026434  162702  001750                SUB     #1000.,R2               ; else carry from low word
2884 026440  005201                        INC     R1                      ; to high word
2885 026442  061001                 60$:   ADD     (R0),R1                 ; and add in high value
2886 026444  062700  000002         70$:   ADD     #2,R0                   ; get next table entry
2887 026450  005703                        TST     R3                      ; all bits processed ?
2888 026452  001361                        BNE     50$                     ; if not, do more bits
2889
2890 026454  023727  003026  000001  80$:   CMP     MODE,#1                 ; bipolar voltage conversion ?
2891 026462  001007                        BNE     90$                     ; if not,branch
2892 026464  160401                        SUB     R4,R1                   ; else make bipolar
2893 026466  002005                        BGE     90$                     ; if still positive, branch
2894 026470  005702                        TST     R2                      ; decimal part zero ?
2895 026472  001403                        BEQ     90$                     ; if yes, branch
2896 026474  162702  001750                SUB     #1000.,R2               ; else borrow from high part
2897 026500  005201                        INC     R1                      ;
2898
2899 026502  012604                 90$:   MOV     (SP)+,R4                ; restore R4
2900 026504  012603                        MOV     (SP)+,R3                ; and R3
2901 026506  000207                        RTS     PC                      ;
```

GLOBAL SUBROUTINES SECTION

```
                                        ; Subroutine ADCON - Analogue to digital conversion routine.
                                        ;
                                        ;**
                                        ; Functional description :
                                        ;
                                        ;       This converts a 2 word analogue value into a 12 bit digital output
                                        ;       pattern. The input is rounded up or down to the nearest lsb value.
                                        ;
                                        ; Inputs :
                                        ;
                                        ;       mode : 0 = unipolar (0 to 10 volts)     complementary binary coded
                                        ;              1 = bipolar  (-10 to +10 volts)  complementary offset binary coded
                                        ;
                                        ;       R1 - millivolts
                                        ;
                                        ;       R2 - microvolts
                                        ;
                                        ; Implicit inputs :
                                        ;
                                        ;       VITAB and ITAB : digital/analogue conversion tables.
                                        ;
                                        ; Outputs :
                                        ;
                                        ;       R1    : 12 bit input pattern.
                                        ;
                                        ; Implicit outputs :
                                        ;
                                        ;       None.
                                        ;
                                        ; Subordinate routines used :
                                        ;
                                        ;       None.
                                        ;
                                        ; Functional side effects :
                                        ;
                                        ;       R2 is destroyed.
                                        ;
                                        ; Calling sequence :
                                        ;
                                        ;  Eg.  MOV     #1,MODE                 ; bipolar conversion
                                        ;       MOV     #-4,R1                  ; -4.001 MILLIVOLTS
                                        ;       MOV     #-1,R2                  ; in R1/R2
                                        ;       JSR     PC,ADCON
                                        ;
                                        ;--
                                        ADCON::
026510  010346                                  MOV     R3,-(SP)                ; save R3
026512  010446                                  MOV     R4,-(SP)                ; and R4
026514  012700  023042                          MOV     #VUPTAB,R0              ; get conversion table for mode 0
026520  005737  003026                          TST     MODE                    ; unipolar mode ?
026524  001413                                 BEQ     30$                     ; if yes, branch
026526  012700  023122                          MOV     #VBPTAB,R0              ; else get conversion table for mode 1
026532  061001                                  ADD     (R0),R1                 ; and convert bipolar to unipolar
026534  020127  023420                          CMP     R1,#10000.              ; was previous value negative ?
026540  001005                                 BNE     30$                     ; if not, branch
026542  005702                                  TST     R2                      ; is decimal part zero ?
```

GLOBAL SUBROUTINES SECTION

```
2960 026544  001403                    BEQ     30$                     ; if yes, branch
2961 026546  062702  001750              ADD     #1000.,R2               ; else borrow from high part
2962 026552  005301                    DEC     R1                      ;
2963
2964 026554  016003  000026    30$:    MOV     22.(R0),R3              ; get rounding values from lowest
2965 026560  016004  000056            MOV     46.(R0),R4              ; significant bit
2966 026564  006203                    ASR     R3                      ; divide by 2
2967 026566  103002                    BCC     40$                     ; if no carry skip next command
2968 026570  062704  001000            ADD     #1000,R4                ; add carry
2969 026574  006204            40$:    ASR     R4                      ; dididе by 2
2970 026576  060402                    ADD     R4,R2                   ; round up the input value
2971 026600  020227  001750            CMP     R2,#1000.               ; lower part is modulo 1000
2972 026604  002403                    BLT     50$                     ;
2973 026606  162702  001750            SUB     #1000.,R2               ; if overflow, carry over to
2974 026612  005201                    INC     R1                      ; high part
2975 026614  060301            50$:    ADD     R3,R1                   ; add in high part of rounding factor
2976 026616  012703  000020            MOV     #20,R3                  ; initialise working register
2977
2978 026622  020110            60$:    CMP     R1,(R0)                 ; compare high value with table entry
2979 026624  002415                    BLT     90$                     ; if less, don't set bit
2980 026626  003003                    BGT     70$                     ; if more, set the bit
2981 026630  020260  000030            CMP     R2,24.(R0)              ; otherwise, must check the low value
2982 026634  002411                    BLT     90$                     ; if less, don't set the bit
2983
2984 026636  166002  000030    70$:    SUB     24.(R0),R2              ; subtract the table entries
2985 026642  002003                    BGE     80$                     ; branch if no borrow needed
2986 026644  062702  001750            ADD     #1000.,R2               ; else add to low word
2987 026650  005301                    DEC     R1                      ; from high word
2988 026652  161001            80$:    SUB     (R0),R1                 ; and low words
2989 026654  052703  000001            BIS     #1,R3                   ; and set the output bit
2990 026660  062700  000002    90$:    ADD     #2,R0                   ; and next table entry
2991 026664  006303                    ASL     R3                      ; ready for next bit
2992 026666  103355                    BCC     60$                     ; if 12 bits not done, go back
2993
2994 026670  006203            100$:   ASR     R3                      ; get the pattern again
2995 026672  010301                    MOV     R3,R1                   ; set up output register
2996 026674  005101                    COM     R1                      ; and complement it
2997 026676  042701  170000            BIC     #170000,R1              ; isolate low 12 bits
2998 026702  012604                    MOV     (SP)+,R4                ; restore R4
2999 026704  012603                    MOV     (SP)+,R3                ; and R3
3000 026706  000207                    RTS     PC                      ;
```

```
                        ; Subroutine DECIN - Signed decimal input routine.
                        ;
                        ;**
                        ; Functional description :
                        ;
                        ;       This solicits a signed decimal number from the operator.
                        ;
                        ; Inputs :
                        ;
                        ;       The locations following the routine call should contain the
                        ;       address of the input prompt string followed by an address at which
                        ;       to store the new input string (11 bytes). The prompt string should
                        ;       be in ASCIZ FORMAT ( eg. .ASCIZ /INPUT VALUE/ ). The store for
                        ;       the input string should contain a default string in ASCIZ format.
                        ;
                        ;       The operator is prompted for a number which can be up to 10 digits
                        ;       long including an optional + or - sign and decimal point.
                        ;
                        ; Implicit inputs :
                        ;
                        ;       None.
                        ;
                        ; Outputs :
                        ;
                        ;       R1 - integer part of operator input
                        ;       R2 - decimal part of operator input
                        ;
                        ;
                        ; Implicit outputs :
                        ;
                        ;       Error messages are printed if the operator types an illegal character,
                        ;       an integer part over 32767, or a decimal part with more than 3 digits.
                        ;
                        ; Subordinate routines used :
                        ;
                        ;       None.
                        ;
                        ; Functional side effects :
                        ;
                        ;       None.
                        ;
                        ; Calling sequence :
                        ;
                        ;   EG. JSR     PC,DECIN
                        ;       PADD                            ; address of prompt message
                        ;       SADD                            ; address for input string
                        ;
                        ;--

026710                  DECIN::
026710  017600  000000          MOV     @(SP),R0                ; get the prompt string address
026714  062716  000002          ADD     #2,(SP)                 ; and skip to number string address
026720  012701  027361          MOV     #GETNUM,R1              ; copy prompt string to
026724  112021          10$:    MOVB    (R0)+,(R1)+             ; address GETNUM
026726  001376          BNE     10$                             ; until nul byte is found
026730  012700  027346          MOV     #SNUM,R0                ; get address of number store
026734  012701  000012          MOV     #10.,R1                 ; contains 10 bytes
```

GLOBAL SUBROUTINES SECTION

```
3059 026740  105020              20$:  CLRB   (R0)+                  ; clear the number store
3060 026742  005301                    DEC    R1                     ; all cleared ?
3061 026744  001375                    BNE    20$                    ; if not, clear another byte
3062 026746  017600  000000       30$:  MOV    @(SP),R0               ; get the current number string address
3063 026752  012701  027346            MOV    #SNUM,R1               ; copy current number to address SNUM
3064 026756  121027  000040       40$:  CMPB   (R0),#'                ; don't copy spaces
3065 026762  001402                    BEQ    50$                    ; ...
3066 026764  111021                    MOVB   (R0),(R1)+             ; copy everything else
3067 026766  001402                    BEQ    60$                    ; stop after nul byte copied
3068 026770  005200              50$:  INC    R0                     ; point to next input character
3069 026772  000771                    BR     40$                    ; and copy it
3070 026774                      60$:  GMANID GETNUM,SNUM,A,-1,0,10.,YES ; get the number string
     026774  104443                                                        TRAP   C$GMAN
     026776  000406                                                        BR     10001$
     027000  027346                                                        .WORD  SNUM
     027002  000152                                                        .WORD  T$CODE
     027004  027361                                                        .WORD  GETNUM
     027006  177777                                                        .WORD  -1
     027010  000000                                                        .WORD  T$LOLIM
     027012  000012                                                        .WORD  T$HILIM
     027014                                                          10001$:
3071 027014  005037  027342            CLR    NR1                    ; clear the number stores
3072 027020  005037  027344            CLR    NR2                    ; ...
3073 027024  012700  027346            MOV    #SNUM,R0               ; point to the start
3074 027030  012701  027342            MOV    #NR1,R1                ; assume integer part first
3075
3076 027034  121027  000053            CMPB   (R0),#'+               ; is 1st character a + ?
3077 027040  001430                    BEQ    110$                   ; if yes, branch
3078 027042  121027  000055            CMPB   (R0),#'-               ; is it a - ?
3079 027046  001425                    BEQ    110$                   ; if yes, branch
3080 027050  121027  000056       70$:  CMPB   (R0),#'.               ; is it a . ?
3081 027054  001017                    BNE    100$                   ; if not, branch
3082
3083 027056  012701  027344       80$:  MOV    #NR2,R1                ; start on decimal part
3084 027062  105760  000002            TSTB   2(R0)                  ; force to 3 digits
3085 027066  001003                    BNE    90$                    ;
3086 027070  112760  000060  000002    MOVB   #'0,2(R0)              ; ie. replace nulls
3087 027076  105760  000003       90$:  TSTB   3(R0)                  ;
3088 027102  001007                    BNE    110$                   ; with zeros
3089 027104  112760  000060  000003    MOVB   #'0,3(R0)              ;
3090 027112  000403                    BR     110$                   ;
3091
3092 027114  105710              100$: TSTB   (R0)                   ; end of string ?
3093 027116  001451                    BEQ    160$                   ; if yes, finish up
3094 027120  000402                    BR     120$                   ; else get next digit
3095
3096 027122  005200              110$: INC    R0                     ; skip over the sign or point
3097 027124  000751                    BR     70$                    ;
3098
3099 027126  121027  000060       120$: CMPB   (R0),#60               ; is character a valid number ?
3100 027132  002403                    BLT    130$                   ; if too low, ask again
3101 027134  121027  000071            CMPB   (R0),#71               ;
3102 027140  003411                    BLE    140$                   ; if not too high, branch
3103
3104 027142                      130$: PRINTF #DECIN3                 ; print 'illegal character'
     027142  012746  027610                                                MOV    #DECIN3,-(SP)
     027146  012746  000001                                                MOV    #1,-(SP)
```

GLOBAL SUBROUTINES SECTION

```
        027152  010600                                                          MOV     SP,R0
        027154  104417                                                          TRAP    C$PNTF
        027156  062706  000004                                                  ADD     #4,SP
  3105  027162  000671                    BR      30$                     ; and ask again
  3106
  3107  027164  021127  006314  140$:     CMP     (R1),#3276.             ; number too high ?
  3108  027170  101013                    BHI     150$                    ; if yes, branch
  3109
  3110  027172  006311                    ASL     (R1)                    ; else multiply by 10
  3111  027174  011102                    MOV     (R1),R2                 ;
  3112  027176  006311                    ASL     (R1)                    ; ready for next character
  3113  027200  006311                    ASL     (R1)                    ;
  3114  027202  060211                    ADD     R2,(R1)                 ;
  3115
  3116  027204  112002                    MOVB    (R0)+,R2                ; save the character
  3117  027206  162702  000060            SUB     #60,R2                  ; convert to number
  3118  027212  060211                    ADD     R2,(R1)                 ; and add to accumulator
  3119  027214  100401                    BMI     150$                    ; if overflow, report error
  3120
  3121  027216  000714                    BR      70$                     ; and get next character
  3122
  3123  027220                    150$:   PRINTF  #DECIN1                 ; print 'number too big'
        027220  012746  027471                                                  MOV     #DECIN1,-(SP)
        027224  012746  000001                                                  MOV     #1,-(SP)
        027230  010600                                                          MOV     SP,R0
        027232  104417                                                          TRAP    C$PNTF
        027234  062706  000004                                                  ADD     #4,SP
  3124  027240  000642                    BR      30$                     ; and get another
  3125
  3126  027242  023727  027344  001750  160$:     CMP     NR2,#1000.              ; decimal part too big ?
  3127  027250  103411                    BLO     170$                    ; if not, branch
  3128  027252                            PRINTF  #DECIN2                 ; print 'only 3 digits allowed'
        027252  012746  027527                                                  MOV     #DECIN2,-(SP)
        027256  012746  000001                                                  MOV     #1,-(SP)
        027262  010600                                                          MOV     SP,R0
        027264  104417                                                          TRAP    C$PNTF
        027266  062706  000004                                                  ADD     #4,SP
  3129  027272  000625                    BR      30$
  3130
  3131  027274  017601  000000          170$:   MOV     @(SP),R1                ; get the current number string address
  3132  027300  012700  027346                  MOV     #SNUM,R0                ; copy current number to
  3133  027304  112021                  180$:   MOVB    (R0)+,(R1)+             ; string store
  3134  027306  001376                          BNE     180$                    ; until nul byte is found
  3135  027310  013701  027342                  MOV     NR1,R1                  ; set up output registers
  3136  027314  013702  027344                  MOV     NR2,R2                  ;
  3137  027320  123727  027346  000055          CMPB    SNUM,#'-                ; was string negative ?
  3138  027326  001002                          BNE     190$                    ; if not, branch
  3139  027330  005401                          NEG     R1                      ; else negate the output
  3140  027332  005402                          NEG     R2                      ;
  3141
  3142  027334  062716  000002          190$:   ADD     #2,(SP)                 ; skip over the second argument
  3143  027340  000207                          RTS     PC                      ; and return
  3144
  3145                                    .NLIST  BEX
  3146
  3147  027342  000000                  NR1:    .WORD   0                       ; store for integer part
  3148  027344  000000                  NR2:    .WORD   0                       ; store for decimal part
```

GLOBAL SUBROUTINES SECTION

```
   3149 027346     055     061     062  SNUM:    .ASCIZ  /-12345.678/            ; store for input string
   3150
   3151 027361                              GETNUM: .BLKB   72.                   ; store for input prompt
   3152 027471     045     116     045  DECIN1: .ASCIZ  /%N%AMUST BE LESS THAN 32768%N/
   3153 027527     045     116     045  DECIN2: .ASCIZ  /%N%AONLY 3 DIGITS MAY FOLLOW THE DECIMAL POINT%N/
   3154 027610     045     116     045  DECIN3: .ASCIZ  /%N%AILLEGAL CHARACTER%N/
   3155
   3156                                     .LIST   BEX
   3157                                     .EVEN
```

GLOBAL SUBROUTINES SECTION

```
                                        ; Subroutine DECOUT - Signed decimal output routine
                                        ;
                                        ;++
                                        ; Functional description :
                                        ;
                                        ;       Routine to print a signed decimal number.
                                        ;
                                        ; Inputs :
                                        ;
                                        ;       R1 - integer part of number to be printed
                                        ;       R2 - decimal part of number to be printed
                                        ;
                                        ; Implicit inputs :
                                        ;
                                        ;       None.
                                        ;
                                        ; Outputs :
                                        ;
                                        ;       The number is printed as follows : -12345.678
                                        ;       ( no sign is printed for positive numbers )
                                        ;       if the decimal part (R2) is zero, the number
                                        ;       is printed as -12345
                                        ;
                                        ; Implicit outputs :
                                        ;
                                        ;       None.
                                        ;
                                        ; Subordinate routines used :
                                        ;
                                        ;       None.
                                        ;
                                        ; Functional side effects :
                                        ;
                                        ;       None.
                                        ;
                                        ; Calling sequence :
                                        ;
                                        ;   EG. MOV     #-10.,R1            ; print -10.001
                                        ;       MOV     #-1,R2              ;
                                        ;       JSR     PC,DECOUT
                                        ;
                                        ;--
                                        DECOUT::
                                                MOV     R1,-(SP)            ; save R1
                                                MOV     R2,-(SP)            ; save R2
                                                TST     R1                  ; R1 > 0 ?
                                                BLT     10$                 ; if not, branch
                                                TST     R2                  ; R2 positive ?
                                                BGE     20$                 ; if yes, branch
                                        10$:    NEG     R1                  ; else make positive
                                                NEG     R2                  ;
                                                PRINTF  #DEC01              ; and print '-'
027660  012746  027756                                                              MOV     #DEC01,-(SP)
027664  012746  000001                                                              MOV     #1,-(SP)
027670  010600                                                                      MOV     SP,R0
```

GLOBAL SUBROUTINES SECTION

```
        027672  104417                                                          TRAP    C$PNTF
        027674  062706  000004                                                  ADD     #4,SP
  3213
  3214  027700                          20$:    PRINTF  #DEC02,R1               ; print the integer part
        027700  010146                                                          MOV     R1,-(SP)
        027702  012746  027762                                                  MOV     #DEC02,-(SP)
        027706  012746  000002                                                  MOV     #2,-(SP)
        027712  010600                                                          MOV     SP,R0
        027714  104417                                                          TRAP    C$PNTF
        027716  062706  000006                                                  ADD     #6,SP
  3215  027722  005702                          TST     R2                      ; decimal part = 0 ?
  3216  027724  001411                          BEQ     DECEX                   ; if yes branch
  3217  027726                                  PRINTF  #DEC03,R2               ; else, print decimal part
        027726  010246                                                          MOV     R2,-(SP)
        027730  012746  027766                                                  MOV     #DEC03,-(SP)
        027734  012746  000002                                                  MOV     #2,-(SP)
        027740  010600                                                          MOV     SP,R0
        027742  104417                                                          TRAP    C$PNTF
        027744  062706  000006                                                  ADD     #6,SP
  3218  027750  012602                  DECEX:  MOV     (SP)+,R2                ; restore R2
  3219  027752  012601                          MOV     (SP)+,R1                ; and R1
  3220  027754  000207                          RTS     PC                      ; and return
  3221
  3222                                          .NLIST  BEX
  3223  027756      045     101     055 DEC01:  .ASCIZ  /%A-/
  3224  027762      045     104     065 DEC02:  .ASCIZ  /%D5/
  3225  027766      045     101     056 DEC03:  .ASCIZ  /%A.%Z3/
  3226                                          .LIST   BEX
  3227                                          .EVEN
```

```
                        ; Subroutine MUL - Multiplication routine.
                        ;
                        ;++
                        ; Functional description :
                        ;
                        ;       This multiplies a 32 bit value in R2 and R1 by a value either in R0 or
                        ;       in the location following the subroutine call.
                        ;
                        ; Inputs :
                        ;
                        ;       R2 - high word of multiplicand
                        ;       R1 - low word of multiplicand
                        ;
                        ;       If entered at location MUL, the location following the subroutine
                        ;       call must contain the value of the multiplier.
                        ;
                        ;       If entered at location MUL1, the multiplier should be placed in R0.
                        ;
                        ;       In either case, the multiplier must be a positive number.
                        ;
                        ; Implicit inputs :
                        ;
                        ;       None.
                        ;
                        ; Outputs :
                        ;
                        ;       R2 - high word of result
                        ;       R1 - low word of result
                        ;
                        ; Implicit outputs :
                        ;
                        ;       If an overflow occurs, the carry bit is set.
                        ;
                        ; Subordinate routines used :
                        ;
                        ;       None.
                        ;
                        ; Functional side effects :
                        ;
                        ;       Locations RESH and RESL are used as storage and are altered.
                        ;
                        ;       R0 is destroyed.
                        ;
                        ; Calling sequence :
                        ;
                        ;       Either a fixed multiplier follows the subroutine call :
                        ;
                        ;       Eg.     MOV     HIGH,R2         ; set up multiplicand high word
                        ;               MOV     LOW,R1          ; and low word
                        ;               JSR     PC,MUL          ; multiply
                        ;               100.                    ; by 100.
                        ;
                        ;       or the multiplier can be set dynamically :
                        ;
                        ;       Eg.     MOV     HIGH,R2         ; set up multiplicand high word
                        ;               MOV     LOW,R1          ; and low word
                        ;               MOV     #100.,R0        ; multiplier is 100.
```

GLOBAL SUBROUTINES SECTION

```
                                ;               JSR     PC,MUL1         ; multiply
                                ;
                                ;--
                                MUL::
                                        MOV     @(SP),R0                ; get the multiplier
                                        ADD     #2,(SP)                 ; jump over the argument
                                MUL1::
                                        CLR     HRES                    ; clear the result high word
                                        CLR     LRES                    ; and low word
                                10$:    ASR     R0                      ; get a bit from the multiplier
                                        BCC     20$                     ; branch if bit was 0
                                        ADD     R1,LRES                 ; else add multiplicand * bit value
                                        ADC     HRES                    ; to result
                                        ADD     R2,HRES                 ;
                                        BCS     40$                     ; branch if overflow
                                20$:    TST     R0                      ; all bits of multiplier used ?
                                        BEQ     30$                     ; if yes, branch
                                        ASL     R1                      ; else double the multiplicand
                                        ROL     R2                      ; high and low words
                                        BCS     40$                     ; branch if overflow
                                        BR      10$                     ; else do next bit

                                30$:    MOV     HRES,R2                 ; save high word of result
                                        MOV     LRES,R1                 ; and low word

                                40$:    RTS     PC                      ; and return

                                HRES:   .WORD   0                       ; high word of result
                                LRES:   .WORD   0                       ; low word of result
```

```
                        ; Subroutine DIV - Division routine.
                        ;
                        ;++
                        ; Functional description :
                        ;
                        ;       This divides a 32 bit value in R2 and R1 by a value either in R0 or
                        ;       in the location following the subroutine call. The result is rounded
                        ;       up or down to the nearest integer value.
                        ;
                        ; Inputs :
                        ;
                        ;       R2 - high word of dividend
                        ;       R1 - low word of dividend
                        ;
                        ;       If entered at location DIV, the location following the subroutine
                        ;       call must contain the value of the divisor.
                        ;
                        ;       If entered at location DIV1, the divisor should be placed in R0.
                        ;
                        ; Implicit inputs :
                        ;
                        ;       None.
                        ;
                        ; Outputs :
                        ;
                        ;       R2 - high word of result
                        ;       R1 - low word of result
                        ;
                        ; Implicit outputs :
                        ;
                        ;       None.
                        ;
                        ; Subordinate routines used :
                        ;
                        ;       None.
                        ;
                        ; Functional side effects :
                        ;
                        ;       Locations RESH, RESL, DIVH and DIVL are used as storage and are
                        ;       altered.
                        ;
                        ;       R0 is loaded with the divisor.
                        ;
                        ; Calling sequence :
                        ;
                        ;       Either a fixed divisor follows the subroutine call :
                        ;
                        ;       Eg.     MOV     HIGH,R2         ; set up dividend high word
                        ;               MOV     LOW,R1          ; and low word
                        ;               JSR     PC,DIV          ; divide
                        ;               100.                    ; by 100.
                        ;
                        ;       or the divisor can be set dynamically :
                        ;
                        ;       Eg.     MOV     HIGH,R2         ; set up dividend high word
                        ;               MOV     LOW,R1          ; and low word
                        ;               MOV     #100.,R0        ; divisor is 100.
```

GLOBAL SUBROUTINES SECTION

```
3375                                    ;               JSR     PC,DIV1         ; divide
3376                                    ;
3377                                    ;--
3378
3379 030072                            DIV::
3380 030072  017600  000000                    MOV     @(SP),R0                ; get the divisor
3381 030076  062716  000002                    ADD     #2,(SP)                 ; jump over the argument
3382 030102                            DIV1::
3383 030102  005037  030066                    CLR     HRES                    ; clear the result high word
3384 030106  005037  030070                    CLR     LRES                    ; and low word
3385
3386 030112  010046                            MOV     R0,-(SP)                ; save the divisor
3387 030114  006216                            ASR     (SP)                    ; divide it by 2
3388 030116  062601                            ADD     (SP)+,R1                ; add to the dividend to round
3389 030120  005502                            ADC     R2                      ; the result up or down
3390
3391 030122  010037  030312                    MOV     R0,LDIV                 ; save divisor for multiplying up
3392 030126  005037  030310                    CLR     HDIV                    ; high word is zero
3393 030132  012737  000001  030316            MOV     #1,LMUL                 ; initialize multiplier for divisor
3394 030140  005037  030314                    CLR     HMUL                    ; high word is 0
3395
3396 030144  023702  030310            10$:    CMP     HDIV,R2                 ; is multiplied divisor greater than
3397 030150  101015                            BHI     30$                     ; or equal to the dividend ?
3398 030152  103403                            BLO     20$                     ;
3399 030154  023701  030312                    CMP     LDIV,R1                 ;
3400 030160  103011                            BHIS    30$                     ; if yes, branch
3401 030162  006337  030312            20$:    ASL     LDIV                    ; else, double the divisor
3402 030166  006137  030310                    ROL     HDIV                    ;
3403 030172  006337  030316                    ASL     LMUL                    ; and its multiplier
3404 030176  006137  030314                    ROL     HMUL                    ;
3405 030202  000760                            BR      10$                     ; and keep multiplying the divisor
3406
3407 030204  023702  030310            30$:    CMP     HDIV,R2                 ; can we subtract multiplied divisor
3408 030210  101021                            BHI     50$                     ; from the remainder ?
3409 030212  103403                            BLO     40$                     ;
3410 030214  023701  030312                    CMP     LDIV,R1                 ;
3411 030220  101015                            BHI     50$                     ; if not, branch
3412 030222  163701  030312            40$:    SUB     LDIV,R1                 ; else subtract the multiplied divisor
3413 030226  005602                            SBC     R2                      ; from the remainder
3414 030230  163702  030310                    SUB     HDIV,R2                 ;
3415 030234  063737  030316  030070            ADD     LMUL,LRES               ; and add the multiplier to the
3416 030242  005537  030066                    ADC     HRES                    ; result
3417 030246  063737  030314  030066            ADD     HMUL,HRES               ;
3418
3419 030254  006237  030310            50$:    ASR     HDIV                    ; half the multiplied divisor
3420 030260  006037  030312                    ROR     LDIV                    ;
3421 030264  006237  030314                    ASR     HMUL                    ; and its multiplier
3422 030270  006037  030316                    ROR     LMUL                    ;
3423 030274  103343                            BCC     30$                     ; go back if multiplier is not zero
3424
3425 030276  013702  030066            60$:    MOV     HRES,R2                 ; save high word of result
3426 030302  013701  030070                    MOV     LRES,R1                 ; and low word
3427
3428 030306  000207                            RTS     PC                      ; and return
3429
3430 030310  000000                    HDIV:   .WORD   0                       ; high word of multiplied divisor
3431 030312  000000                    LDIV:   .WORD   0                       ; low word of multiplied divisor
```

GLOBAL SUBROUTINES SECTION

```
   3432 030314  000000                HMUL:   .WORD   0               ; high word of multiplier for divisor
   3433 030316  000000                LMUL:   .WORD   0               ; low word of multiplier for divisor
```

```
                                ; Subroutine SETTAB - Routine to set up the control table and a DMA buffer.
                                ;
                                ;**
                                ; Functional description :
                                ;
                                ;       This takes a table and uses it to set up the control table of the
                                ;       AAF01-A and to set up a DMA buffer containing the desired output
                                ;       values.
                                ;
                                ; Inputs :
                                ;
                                ;       The locations following the subroutine call must contain the
                                ;       number of control words to set up and the address of the table
                                ;       to be used for the set up.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       Table to be used for set up.
                                ;
                                ;       Eg.     ; Outputs :   Mode   Channel  Voltage
                                ;                                             (mV)
                                ;
                                ;               T10TAB: .WORD   0,      0,      1000.   ; table for outputs
                                ;                       .WORD   0,      1,      2000.
                                ;                       .WORD   0,      2,      3000.
                                ;                       .WORD   0,      3,      4000.
                                ;
                                ; Outputs :
                                ;
                                ;       The control table of the AAF01-A is set up and data is set up in
                                ;       the buffer BUFOUT.
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       ADCON - analogue to digital conversion routine.
                                ;
                                ; Functional side effects :
                                ;
                                ;       The CTA register is left pointing at control word 0.
                                ;
                                ; Calling sequence :
                                ;
                                ;  Eg.  JSR     PC,SETTAB                  ; set up control table and data
                                ;       4                                  ; for 4 control words
                                ;       T10TAB                             ; using table at T10TAB
                                ;
                                ;--
030320                          SETTAB::
030320                                  PUSH$   <R1,R2,R3,R4,R5>           ; save the registers
030332  017637  000012  030520          MOV     @12(SP),NWORDS             ; get number of words to set up
030340  062766  000002  000012          ADD     #2,12(SP)                  ; jump over the argument
```

GLOBAL SUBROUTINES SECTION

```
3492 030346  017605  000012                  MOV     @12(SP),R5              ; point to the table
3493 030352  062766  000002  000012          ADD     #2,12(SP)               ; and jump over the argument
3494
3495 030360  005002                          CLR     R2                      ; 0 microvolts in conversions
3496 030362  012703  003040                  MOV     #BUFOUT,R3              ; DMA buffer address in R3
3497 030366  012704  020000                  MOV     #CTA,R4                 ; control word number in R4
3498
3499 030372  010477  152236          10$:    MOV     R4,@DRXDBR              ; address the CTA
3500 030376  052777  000001  152222          BIS     #FNCTO,@DRXSCR          ; and transfer the control word address
3501 030404  011500                          MOV     (R5),R0                 ; get the mode from the table
3502 030406  000300                          SWAB    R0                      ; put it into bits 6-8
3503 030410  006200                          ASR     R0                      ;
3504 030412  006200                          ASR     R0                      ;
3505 030414  056500  000002                  BIS     2(R5),R0                ; now include the channel number
3506 030420  052700  030000                  BIS     #CWR,R0                 ; multiplex to the control word
3507 030424  010077  152204                  MOV     R0,@DRXDBR              ; address the control word register
3508 030430  052777  000001  152170          BIS     #FNCTO,@DRXSCR          ; and transfer to the control word
3509
3510 030436  016501  000004                  MOV     4(R5),R1                ; get the voltage from the table
3511 030442  004737  026510                  JSR     PC,ADCON                ; convert to 12 bit data
3512 030446  052701  070000                  BIS     #70000,R1               ; set RSA bits for output with MNT set
3513 030452  010123                          MOV     R1,(R3)+                ; save the data in the buffer
3514 030454  005204                          INC     R4                      ; point to next control word
3515 030456  062705  000006                  ADD     #6,R5                   ; move to next line of input table
3516 030462  005337  030520                  DEC     NWORDS                  ; all channels done ?
3517 030466  001341                          BNE     10$                     ; if not, do another
3518
3519 030470  012777  020000  152136          MOV     #CTA,@DRXDBR            ; address control word 0
3520 030476  052777  000001  152122          BIS     #FNCTO,@DRXSCR          ; transfer to CTA register
3521 030504                                  POP     <R5,R4,R3,R2,R1>        ; restore the registers
3522 030516  000207                          RTS     PC                      ; and return
3523
3524 030520  000000          NWORDS: .WORD   0                               ; number of control words to set up
```

GLOBAL SUBROUTINES SECTION

```
                        ; Subroutine ADFIN - Routine to Input from the ADF01.
                        ;
                        ;**
                        ; Functional description :
                        ;
                        ;       This sets up the ADF01 connected to the unit currently under test to
                        ;       perform a DMA input, placing the data in the buffer at address BUFIN.
                        ;       If the required number of conversions is greater than the number of
                        ;       channels, the channel sequence is used repeatedly until all of the
                        ;       inputs have been made.
                        ;
                        ; Inputs :
                        ;
                        ;       FCHIN   - first channel for input
                        ;       LCHIN   - last channel for input
                        ;       NCONS   - number of conversions to make
                        ;       EXTCLK  - set to 4 if the external clock is to be used (this enables
                        ;                 synchronization with the AAF01-A).
                        ;
                        ; Implicit inputs :
                        ;
                        ;       ADFSCR  - contains address of ADF's DRX11-C SCR register
                        ;       ADFCOR  - contains address of ADF's DRX11-C COR register
                        ;       ADFADR  - contains address of ADF's DRX11-C ADR register
                        ;       ADFDBR  - contains address of ADF's DRX11-C DBR register
                        ;       CTIME   - conversion time selected by startup questions
                        ;
                        ; Outputs :
                        ;
                        ;       The control table of the ADF01 is set up and the DRX11-C is set to DMA
                        ;       data in buffer DATIN.
                        ;
                        ; Implicit outputs :
                        ;
                        ;       None.
                        ;
                        ; Subordinate routines used :
                        ;
                        ;       None.
                        ;
                        ; Functional side effects :
                        ;
                        ;       None.
                        ;
                        ; Calling sequence :
                        ;
                        ;  Eg.  MOV     #1,FCHIN        ; use channels 1
                        ;       MOV     #2,LCHIN        ; and 2 repeatedly
                        ;       MOV     2048.,NCONS     ; to input 2048 words
                        ;       MOV     #ECE,EXTCLK     ; trigger inputs from the external clock
                        ;       JSR     PC,ADFIN        ; start the input
                        ;
                        ;--
030522                  ADFIN::
030522                          PUSH$   <R0,R1,R2,R3,R4,R5>     ; save the registers
```

GLOBAL SUBROUTINES SECTION

```
3583 030536  052777  000040  152074          BIS     #RES,@ADFSCR            ; reset the ADF01
3584 030544  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
3585 030550  052701  040000                  BIS     #PCR,R1                 ; address the PCR
3586 030554  010177  152066                  MOV     R1,@ADFDBR              ; and enter the conversion time
3587 030560  052777  000001  152052          BIS     #FNCTO,@ADFSCR          ; transfer to PCR
3588
3589 030566  013701  031006                  MOV     FCHIN,R1                ; save the first channel to use
3590 030572  013702  031010                  MOV     LCHIN,R2                ; and the last channel
3591 030576  013703  031012                  MOV     NCONS,R3                ; and the number of conversions
3592
3593 030602  052701  030000                  BIS     #CWR,R1                 ; convert first channel to CWR
3594 030606  052702  030000                  BIS     #CWR,R2                 ; convert last channel to CWR
3595 030612  010104                          MOV     R1,R4                   ; start with first channel
3596 030614  012705  020000                  MOV     #CTA,R5                 ; and first control table address
3597
3598 030620  010577  152022          10$:    MOV     R5,@ADFDBR              ; get the current CWR address
3599 030624  052777  000001  152006          BIS     #FNCTO,@ADFSCR          ; and transfer to the CTA
3600 030632  010477  152010                  MOV     R4,@ADFDBR              ; get the current channel
3601 030636  052777  000001  151774          BIS     #FNCTO,@ADFSCR          ; and transfer to the CWR
3602 030644  005205                          INC     R5                      ; next control table address
3603 030646  005204                          INC     R4                      ; next channel
3604 030650  020402                          CMP     R4,R2                   ; all channels done ?
3605 030652  101762                          BLOS    10$                     ; if not, go back
3606
3607 030654  005304                          DEC     R4                      ; reset to previous channel
3608 030656  052704  000100                  BIS     #100,R4                 ; set up to mode 1
3609 030662  010477  151760                  MOV     R4,@ADFDBR              ; and rewrite the previous channel
3610 030666  052777  000001  151744          BIS     #FNCTO,@ADFSCR          ; transfer to the CWR
3611 030674  012777  020000  151744          MOV     #CTA,@ADFDBR            ; reset the CTA to point
3612 030702  052777  000001  151730          BIS     #FNCTO,@ADFSCR          ; back to the first control word
3613
3614 030710  012777  000000  151724          MOV     #BAR1,@ADFCOR           ; point to BAR1
3615 030716  013777  023040  151720          MOV     BUFIN,@ADFADR           ; and insert DMA buffer address
3616 030724  012777  000400  151710          MOV     #WCR1,@ADFCOR           ; point to WCR1
3617 030732  005403                          NEG     R3                      ; form wordcount
3618 030734  010377  151704                  MOV     R3,@ADFADR              ; and insert into WCR1
3619 030740  012777  060000  151674          MOV     #INOUT!RUN,@ADFCOR      ; set the data direction (from @ADF)
3620 030746  013701  031014                  MOV     EXTCLK,R1               ; set the external clock if selected
3621 030752  052701  010001                  BIS     #ACS!GO,R1              ; start the DMA
3622 030756  010177  151664                  MOV     R1,@ADFDBR              ; transfer to the DBR
3623 030762  052777  000001  151650          BIS     #FNCTO,@ADFSCR          ; and start the input
3624
3625 030770                                  POP     <R5,R4,R3,R2,R1,R0>     ; restore the registers
3626 031004  000207                          RTS     PC                      ; and return
3627
3628 031006  000000          FCHIN:  .WORD   0                               ; first channel for input
3629 031010  000000          LCHIN:  .WORD   0                               ; last channel for input
3630 031012  000000          NCONS:  .WORD   0                               ; number of conversions
3631 031014  000000          EXTCLK: .WORD   0                               ; external clock (set to 400 to enable)
3632
```

GLOBAL SUBROUTINES SECTION

```
                        ; Subroutine IEXIN - Routine to input from an IEx11-A using an HP3455.
                        ;
                        ;**
                        ; Functional description :
                        ;
                        ;       This routine allows the output from a HP3455 digital voltmeter
                        ;       to be read via an IEx11-A interface. The result is saved in
                        ;       memory at location HPSAVE and in registers R1 and R2.
                        ;
                        ;       Channel 1 of the IEx11-A is set up as system controller and an
                        ;       IFC sent over the IEEE bus. The IEx11-A is then selected as
                        ;       talker and the HP3455 as listener in order to set up the HP3455.
                        ;
                        ;       The IEx11-A is then selected as listener with the HP3455 as
                        ;       talker and the voltage from the HP3455 is read over the IEEE bus
                        ;       into memory as an Ascii string. Finally, the string is converted
                        ;       to an integer and decimal value in R1 and R2.
                        ;
                        ; Inputs :
                        ;
                        ;       None.
                        ;
                        ; Implicit inputs :
                        ;
                        ;       IEXADD  - address of the IEx11-A to which the HP3455 is
                        ;                 connected (default is set to 164100).
                        ;
                        ; Outputs :
                        ;
                        ;       HPSAVE  - starting address of the memory area in which the
                        ;                 measurement is saved. The format is the same as that
                        ;                 output by the HP3455 (eg. +12.123450e+01).
                        ;
                        ;       R1      - integer part of the measurement (mV)
                        ;
                        ;       R2      - decimal part of the measurement (uV)
                        ;
                        ;       ERFLA   - this location is loaded with an error number if an
                        ;                 error is detected in the subroutine.
                        ;
                        ; Implicit outputs :
                        ;
                        ;       None.
                        ;
                        ; Subordinate routines used :
                        ;
                        ;       None.
                        ;
                        ; Functional side effects :
                        ;
                        ;       None.
                        ;
                        ; Calling sequence :
                        ;
                        ;       JSR     PC,IEXIN
                        ;
                        ;--
```

GLOBAL SUBROUTINES SECTION

```
                                ; IEx11-A register offsets :

000000                                  IEXISR = 00
000002                                  IEXIIR = 02
000004                                  IEXICR = 04
000006                                  IEXIDR = 06
000010                                  IEXCSR = 10
000012                                  IEXBDR = 12
000014                                  IEXBCR = 14
000016                                  IEXMCR = 16

031016                          IEXIN::
031016  010046                          MOV     R0,-(SP)                ; save R0 and R3
031020  010346                          MOV     R3,-(SP)                ; ...

031022  013701  032022                  MOV     IEXADD,R1               ; save address of IEx11-A

031026  005061  000010                  CLR     IEXCSR(R1)              ; clear IEX CSR (select cha.1)
031032  005061  000010                  CLR     IEXCSR(R1)              ; clear IEX CSR
031036  112761  000200  000005  10$:    MOVB    #200,IEXICR+1(R1)       ; set SWRST in ICRH (initialize cha. 1)
031044  105061  000005                  CLRB    IEXICR+1(R1)            ; clear ICRH (SWRST)
031050  105061  000000                  CLRB    IEXISR(R1)              ; disable all interrupts (mask reg.0,ISR0)
031054  105061  000003                  CLRB    IEXIIR+1(R1)            ; load DPA "0" into IIRH (IEEE bus adr.)
031060  012761  000002  000010          MOV     #2,IEXCSR(R1)           ; set SYS CONT bit in CSR
031066  112761  000217  000005          MOVB    #217,IEXICR+1(R1)       ; set IFC line true into ICRH
031074  012700  000500                  MOV     #500,R0                 ; delay for release of IFC
031100  005300                  20$:    DEC     R0                      ; (100 ms)
031102  001376                          BNE     20$                     ; ...
031104  112761  000017  000005          MOVB    #17,IEXICR+1(R1)        ; clear IFC (SIC command into ICRH)

                                ; Select the HP3455 as listener and set remote enable

031112  112761  000212  000005          MOVB    #212,IEXICR+1(R1)       ; select cha.1 as talker (TON to ICRH)
031120  112761  000220  000005          MOVB    #220,IEXICR+1(R1)       ; send remote enable (SRE) to HP3455
031126  112761  000066  000007          MOVB    #66,IEXIDR+1(R1)        ; address HP and switch to listener
031134  112761  000213  000005          MOVB    #213,IEXICR+1(R1)       ; go into standby state (CACS->CSBS)
031142  132761  000020  000002  30$:    BITB    #20,IEXIIR(R1)          ; is CONTR. changed into standby state
031150  001774                          BEQ     30$                     ; wait if bo bit was set

                                ; Set up the HP3455

031152  112761  000124  000007          MOVB    #124,IEXIDR+1(R1)       ; switch HP3455 to external trigger
031160  132761  000020  000002  40$:    BITB    #20,IEXIIR(R1)          ; data out reg. ready to send a byte ?
031166  001774                          BEQ     40$                     ; wait if BO bit was set
031170  112761  000062  000007          MOVB    #62,IEXIDR+1(R1)        ; second byte for external trigger
031176  132761  000020  000002  50$:    BITB    #20,IEXIIR(R1)          ; data out reg. ready to send a byte ?
031204  001774                          BEQ     50$                     ; wait if BO bit was set
031206  112761  000122  000007          MOVB    #122,IEXIDR+1(R1)       ; select 10V range
031214  132761  000020  000002  60$:    BITB    #20,IEXIIR(R1)          ; data out reg. ready to send a byte ?
031222  001774                          BEQ     60$                     ; wait if BO bit was set
031224  112761  000063  000007          MOVB    #63,IEXIDR+1(R1)        ; second byte for 10V range
031232  132761  000020  000002  70$:    BITB    #20,IEXIIR(R1)          ; data out reg. ready to send a byte ?
031240  001774                          BEQ     70$                     ; wait if BO bit was set
031242  112761  000014  000005          MOVB    #14,IEXICR+1(R1)        ; take control TCA (CSBS->CACS)
                                                                        ; ready for command - addr.
031250  112761  000010  000007          MOVB    #10,IEXIDR+1(R1)        ; trigger command to HP (get into DOUT)
```

GLOBAL SUBROUTINES SECTION

```
3748 031256  112761  000211  000005          MOVB    #211,IEXICR+1(R1)       ; select cha.1 as listener (ICRH)
3749 031264  132761  000020  000002  80$:    BITB    #20,IEXIIR(R1)          ; data out reg. ready to send a byte ?
3750 031272  001774                          BEQ     80$                     ; wait if BO bit was set
3751 031274  112761  000126  000007          MOVB    #126,IEXIDR+1(R1)       ; address HP and switch to talker
3752 031302  132761  000020  000002  90$:    BITB    #20,IEXIIR(R1)          ; data out reg. ready to send a byte ?
3753 031310  001774                          BEQ     90$                     ; wait if BO bit was set
3754 031312  112761  000213  000005          MOVB    #213,IEXICR+1(R1)       ; go into standby state (CACS->CSBS)
3755                                                                          ; ready for recive data from HP
3756
3757                                 ; Save all received data in the receive buffer
3758
3759 031320  012700  032024                  MOV     #HPSAVE,R0              ; load start address of receive buf.
3760 031324  162700  000002                  SUB     #2,R0                   ; ...
3761 031330  062700  000002          100$:   ADD     #2,R0                   ; increment buffer address
3762 031334  032761  000040  000002  110$:   BIT     #40,IEXIIR(R1)          ; BI in IRR set (data received from HP)
3763 031342  001774                          BEQ     110$                    ; branch if not
3764 031344  016110  000006                  MOV     IEXIDR(R1),(R0)         ; save recived data
3765 031350  000310                          SWAB    (R0)                    ; swap high byte into low byte
3766 031352  121027  000012                  CMPB    (R0),#12                ; all data recived ?
3767 031356  001364                          BNE     100$                    ; branch if not
3768
3769
3770                                 ; Clean up the IEEE bus interfaces to leave in a neutral state after exit
3771
3772 031360  112761  000014  000005          MOVB    #14,IEXICR+1(R1)        ; take control TCA (CSBS->CACS)
3773                                                                          ; ready for command + addr.
3774 031366  112761  000024  000007          MOVB    #24,IEXIDR+1(R1)        ; send deveice clear (DCL) to HP
3775 031374  132761  000020  000002  120$:   BITB    #20,IEXIIR(R1)          ; data out reg. ready to send a byte ?
3776 031402  001774                          BEQ     120$                    ; wait if BO bit was set
3777 031404  112761  000217  000005          MOVB    #217,IEXICR+1(R1)       ; set IFC line true into ICRH
3778 031412  012700  000500                  MOV     #500,R0                 ; delay for release of IFC
3779 031416  005300                  130$:   DEC     R0                      ; (100 ms)
3780 031420  001376                          BNE     130$                    ; ...
3781 031422  112761  000017  000005          MOVB    #17,IEXICR+1(R1)        ; clear IFC (SIC command into ICRH)
3782 031430  112761  000200  000005          MOVB    #200,IEYICR+1(R1)       ; set SWRST in ICRH (initialize cha. 1)
3783 031436  105061  000005                  CLRB    IEXICR+1(R1)            ; clear ICRH (SWRST)
3784
3785
3786                                 ; Convert the contents of the receive buffer and place in R1 and R2
3787
3788 031442  012700  032024          HPPS1:  MOV     #HPSAVE,R0              ; receive buffer (HP3455 output)
3789 031446  012701  032070                  MOV     #CNSAVE,R1              ; changed receive buffer
3790
3791 031452  121027  000053          10$:    CMPB    (R0),#'+                ; is 1st character a + ?
3792 031456  001403                          BEQ     20$                     ; if yes, branch
3793 031460  121027  000055                  CMPB    (R0),#'-                ; is it a - ?
3794 031464  001002                          BNE     30$                     ; if not, branch
3795 031466  062700  000002          20$:    ADD     #2,R0                   ; skip over the sign
3796 031472  121027  000056          30$:    CMPB    (R0),#'.                ; is it a point ?
3797 031476  001402                          BEQ     40$                     ; if yes, branch
3798 031500  112021                          MOVB    (R0)+,(R1)+             ; build intger part of changed buffer
3799 031502  000773                          BR      30$                     ;
3800
3801 031504  062700  000002          40$:    ADD     #2,R0                   ; skip over the point
3802 031510  005003                          CLR     R3                      ;
3803 031512  005002                          CLR     R2                      ;
3804 031514  116003  000022                  MOVB    22(R0),R3               ; get exponent from receive buffer
```

GLOBAL SUBROUTINES SECTION

```
3805 031520  162703  000060              SUB     #60,R3                  ; build number
3806 031524  020302              4$:    CMP     R3,R2                   ;
3807 031526  001403                     BEQ     3$                      ;
3808 031530  012021                     MOV     (R0)+,(R1)+             ;
3809 031532  005202                     INC     r2                      ;
3810 031534  000773                     BR      4$                      ;
3811
3812 031536  012021              3$:    MOV     (R0)+,(R1)+             ; shift piont 3 digits to the
3813 031540  012021                     MOV     (R0)+,(R1)+             ; right
3814 031542  012021                     MOV     (R0)+,(R1)+             ; ...
3815 031544  012721  000056             MOV     #'.,(R1)+               ; ...
3816
3817 031550  121027  000105       50$:  CMPB    (R0),#'E                ; is found character an E ?
3818 031554  001402                     BEQ     60$                     ; branch if yes
3819 031556  112021                     MOVB    (R0)+,(R1)+             ; build decimal part of changed buffer
3820 031560  000773                     BR      50$                     ;
3821 031562  005021              60$:   CLR     (R1)+                   ; clear following three locations
3822 031564  005021                     CLR     (R1)+                   ; ...
3823 031566  005011                     CLR     (R1)                    ; ...
3824 031570  012700  032070             MOV     #CNSAVE,R0              ; point to the start of the changed buf
3825 031574  012701  032134             MOV     #HPNR1,R1               ; assume integer part
3826 031600  005011                     CLR     (R1)                    ; clean up location
3827
3828 031602  042710  177400       70$:  BIC     #177400,(R0)            ; clear high byte of changed buffer
3829 031606  105710                     TSTB    (R0)                    ; end of string ?
3830 031610  001460                     BEQ     110$                    ; if yes, finish up
3831 031612  121027  000060             CMPB    (R0),#60                ; is received char. a valid number ?
3832 031616  002403                     BLT     80$                     ; if too low, mask for error
3833 031620  121027  000071             CMPB    (R0),#71                ;
3834 031624  003404                     BLE     90$                     ; if not too high, branch
3835
3836 031626  012737  000001  032140 80$:  MOV    #1,ERFLA                ; "illegal character" mark it
3837 031634  000467                     BR      130$                    ; exit routine
3838
3839 031636  162710  000060       90$:  SUB     #60,(R0)                ; convert to number
3840
3841 031642  021127  006314             CMP     (R1),#3276.             ; number too high ?
3842 031646  101367                     BHI     80$                     ; if yes, branch
3843
3844 031650  006311                     ASL     (R1)                    ; else multiply by 10
3845 031652  011102                     MOV     (R1),R2                 ;
3846 031654  006311                     ASL     (R1)                    ; ready for next character
3847 031656  006311                     ASL     (R1)                    ;
3848 031660  060211                     ADD     R2,(R1)                 ;
3849
3850 031662  012002                     MOV     (R0)+,R2                ; save the character
3851 031664  060211                     ADD     R2,(R1)                 ; and add to accumulator
3852 031666  100425                     BMI     100$                    ; if overflow, report error
3853
3854 031670  121027  000056             CMPB    (R0),#'.                ; is next charcter a point ?
3855 031674  001342                     BNE     70$                     ; if not, branch
3856 031676  062700  000002             ADD     #2,R0                   ; skip over the point
3857 031702  012701  032136             MOV     #HPNR2,R1               ; start no decimal part
3858 031706  005011                     CLR     (R1)                    ; clean up location
3859 031710  105760  000002             TSTB    2(R0)                   ; is this loaction loaded with data
3860 031714  001003                     BNE     44$                     ; branch if yes
3861 031716  112760  000060  000002     MOVB    #'0,2(R0)               ; otherwise fill with ascii 0 = 60
```

GLOBAL SUBROUTINES SECTION

```
3862 031724  105760  000004          44$:    TSTB    4(R0)                   ; is this location loaded with data
3863 031730  001324                          BNE     70$                     ; branch if yes
3864 031732  112760  000060  000004          MOVB    #'0,4(R0)               ; otherwise fill with ascii 0 = 60
3865 031740  000720                          BR      70$                     ; get next character
3866
3867 031742  012737  000002  032140  100$:   MOV     #2,ERFLA                ; "number too big" load error code
3868 031750  000421                          BR      130$                    ; exit routine
3869
3870 031752  013701  032134          110$:   MOV     HPNR1,R1                ; set up output registers
3871 031756  013702  032136                  MOV     HPNR2,R2                ;
3872 031762  020227  001750                  CMP     R2,#1000.               ; decimal part too big ?
3873 031766  103404                          BLO     120$                    ; if not, branch
3874 031770  012737  000003  032140          MOV     #3,ERFLA                ; save error messages
3875 031776  000406                          BR      130$                    ; exit routine
3876
3877 032000  123727  032024  000055  120$:   CMPB    HPSAVE,#'-              ; was string negative ?
3878 032006  001002                          BNE     130$                    ; if not, branch
3879 032010  005401                          NEG     R1                      ; else negate the output
3880 032012  005402                          NEG     R2                      ;
3881 032014                          130$:
3882 032014  012603                          MOV     (SP)+,R3                ; restore R3 and R0
3883 032016  012600                          MOV     (SP)+,R0                ; ...
3884
3885 032020  000207                          RTS     PC                      ; and return
3886
3887 032022  164100                  IEXADD::.WORD   164100                  ; IEx11-A address
3888
3889 032024                          HPSAVE: .BLKW   22                      ; buffer for received data
3890 032070                          CNSAVE: .BLKW   22                      ; buffer for conversion
3891 032134  000000                  HPNR1:  .WORD   0                       ; store for integer part
3892 032136  000000                  HPNR2:  .WORD   0                       ; store for decimal part
3893 032140  000000                  ERFLA:  .WORD   0                       ;
3894
3895                                         .EVEN
```

GLOBAL SUBROUTINES SECTION

```
                                ; Subroutine RANDOM - Routine to generate a pseudo random number.
                                ;
                                ; Functional description :
                                ;
                                ;       This routine generates a random pattern. The pattern is stored
                                ;       in location RB and in R0.
                                ;
                                ; Inputs :
                                ;
                                ;       None.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       RA and RB.
                                ;
                                ; Outputs :
                                ;
                                ;       RB - contains the random pattern
                                ;       RA - contains a second random pattern
                                ;       R0 - same as RB
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routines used :
                                ;
                                ;       None.
                                ;
                                ; Calling sequence :
                                ;
                                ;       JSR PC,RANDOM
                                ;
                                ;--

032142                          RANDOM::
032142  013746  032200                  MOV     RA,-(SP)        ; push RA to stack
032146  013700  032202                  MOV     RB,R0           ; get the last random pattern
032152  006316                          ASL     @SP             ; sift SP (=RA) left
032154  005500                          ADC     R0              ; if carry is set add to R0 (=RB)
032156  006200                          ASR     R0              ; then shift the result Rith
032160  005516                          ADC     @SP             ; if carry is set add to SP (=RA)
032162  061600                          ADD     @SP,R0          ; add SP (=RA) and R0 (=RB)
032164  005600                          SBC     R0              ; subtract carry if set from RB
032166  012637  032200                  MOV     (SP)+,RA        ; load new value into location RA
032172  010037  032202                  MOV     R0,RB           ; load location RB with new pat.
032176  000207                          RETURN                  ;

032200  135753                  RA::    .WORD   135753          ; start pattern for RB
032202  024674                  RB::    .WORD   24674           ; storage for random pattern
```

GLOBAL SUBROUTINES SECTION

```
                                        ;*****************************;
                                        ; INTERRUPT SERVICE ROUTINES ;
                                        ;*****************************;

                                ; Interrupt service routine NXM - non existant memory trap.

                                ;++
                                ; This routine sets a flag NXMFLG to 1. It is executed when a non
                                ; existant memory trap occurs if vector 4 has been loaded with the
                                ; address NXM.
                                ;
                                ; NXMFLG should be cleared immediately before executing code which may
                                ; address non existant memory.
                                ;--

032204                                  BGNSRV  NXM
032204                                                                          NXM::
032204  012737  000001  002770          MOV     #1,NXMFLG               ; flag NXM trap
032212                                  ENDSRV
032212                                                                          L10017:
032212  000002                                                                    RTI

                                ; Dummy clock interrupt service routine CLINT.

                                ;++
                                ; This routine is a dummy service for line time clock interrupts and
                                ; will be executed when an interrupt to vector 100 occurs.
                                ;--

032214                                  BGNSRV  CLINT                   ; do nothing
032214                                                                          CLINT::
032214                                  ENDSRV
032214                                                                          L10020:
032214  000002                                                                    RTI

                                ; DRX11-C interrupt service routine INT.

                                ;++
                                ; This routine increments a flag INTFLG. It is entered when a DRX11-C
                                ; interrupt occurs if the vector has been loaded with the address INT.
                                ;
                                ; INTFLG should be cleared immediately before executing code which may
                                ; cause a DRX11-C interrupt.
                                ;--

032216                                  BGNSRV  INT
032216                                                                          INT::
032216  005237  002772                  INC     INTFLG          ; flag DRX11-C interrupt
032222                                  ENDSRV
032222                                                                          L10021:
032222  000002                                                                    RTI

                                ; DRX11-C interrupt service routine INT1.
```

GLOBAL SUBROUTINES SECTION

```
   3997
   3998                                 ;++
   3999                                 ; This routine sets a flag INTFLG. It is entered when a DRX11-C
   4000                                 ; interrupt occurs if the vector has been loaded with the address INT1.
   4001                                 ;
   4002                                 ; INTFLG should be cleared immediately before executing code which may
   4003                                 ; cause a DRX11-C interrupt.
   4004                                 ;--
   4005
   4006 032224                                 BGNSRV  INT1
        032224                                                                          INT1::
   4007 032224  012737  000001  002772         MOV     #1,INTFLG       ; flag DRX11-C interrupt
   4008 032232                                 ENDSRV
        032232                                                                          L10022:
        032232  000002                                                                          RTI
   4009
```

GLOBAL SUBROUTINES SECTION

```
4011
4012
4013
4014 032234                                    ENDMOD
4015
```

```
                                .TITLE MISCELLANEOUS SECTIONS
                                .SBTTL  REPORT CODING SECTION

032234                                  BGNMOD

                                ; Print routine
                                ;++
                                ; Functional description :
                                ;
                                ;       Prints out test titles, or a statistics table for the units
                                ;       under test. Which to print is determined by user input.
                                ;
                                ;       The statistics table displays the number of errors which the
                                ;       diagnostic has detected for each unit, and whether the unit has
                                ;       been dropped from testing.
                                ;
                                ; Inputs :
                                ;
                                ;       The user is asked to type a character indicating whether to
                                ;       print the test titles or the statistics table.
                                ;
                                ; Implicit inputs :
                                ;
                                ;       The error table 'ECNT' is used for the statistics printout.
                                ;
                                ;       Test titles are assumed to be labelled with the format TDHDnn,
                                ;       where nn is the test number. NTESTS at the start of the routine
                                ;       must equal the number of tests in the diagnostic.
                                ;
                                ; Outputs :
                                ;
                                ;       Either a list of test titles or a statistics table are output.
                                ;
                                ; Implicit outputs :
                                ;
                                ;       None.
                                ;
                                ; Subordinate routires used :
                                ;
                                ;       CRLF    - lire feed print routine.
                                ;
                                ; Functional side effects :
                                ;
                                ;       Registers R1 to R5 are corrupted.
                                ;
                                ; Calling sequence :
                                ;
                                ;       Invoked by the operator print command.
                                ;
                                ;--

000034                                  NTESTS=28.                      ; 28 tests for title printout

032234                                  BGNRPT
032234                                                                  L$RPT::
032234                          ASK:    MANUAL                          ; manual intervention disabled ?
```

REPORT CODING SECTION

```
        032234  104450                                                          TRAP    C$MANI
4111    032236                          BNCOMPLETE STAT                 ; if yes, print statistics
        032236  103126                                                          BCC     STAT
4112    032240                          GMANID  PR1,CHAR,A,377,1,4,YES  , prompt for a command
        032240  104443                                                          TRAP    C$GMAN
        032242  000406                                                          BR      10000$
        032244  032704                                                          .WORD   CHAR
        032246  000152                                                          .WORD   T$CODE
        032250  033112                                                          .WORD   PR1
        032252  000377                                                          .WORD   377
        032254  000001                                                          .WORD   T$LOLIM
        032256  000004                                                          .WORD   T$HILIM
        032260                                                          10000$:
4113    032260  023727  032704  000124          CMP     CHAR,#'T        ; test list requested ?
4114    032266  001427                          BEQ     TITLE           ; if yes, output titles
4115    032270  023727  032704  000123          CMP     CHAR,#'S        ; statistics requested ?
4116    032276  001002                          BNE     HEL             ; if not, print the help message
4117    032300  000137  032514                  JMP     STAT            ; if yes, output statistics
4118
4119    032304                          HEL:    PRINTF  #PR2            ; otherwise, print the help message
        032304  012746  033133                                                  MOV     #PR2,-(SP)
        032310  012746  000001                                                  MOV     #1,-(SP)
        032314  010600                                                          MOV     SP,R0
        032316  104417                                                          TRAP    C$PNTF
        032320  062706  000004                                                  ADD     #4,SP
4120    032324                          PRINTF  #PR2A           ; ...
        032324  012746  033247                                                  MOV     #PR2A,-(SP)
        032330  012746  000001                                                  MOV     #1,-(SP)
        032334  010600                                                          MOV     SP,R0
        032336  104417                                                          TRAP    C$PNTF
        032340  062706  000004                                                  ADD     #4,SP
4121    032344  000733                  BR      ASK             ; and prompt for command again
4122
4123    032346                          TITLE:  PRINTF  #TT             ; test list header
        032346  012746  033004                                                  MOV     #TT,-(SP)
        032352  012746  000001                                                  MOV     #1,-(SP)
        032356  010600                                                          MOV     SP,R0
        032360  104417                                                          TRAP    C$PNTF
        032362  062706  000004                                                  ADD     #4,SP
4124    032366  012701  000001                  MOV     #1,R1           ; start with test 1
4125    032372  012702  032714                  MOV     #TADS,R2        ; start of list of title addresses
4126
4127    032376  012703  000022          10$:    MOV     #18.,R3         ; wait after 18 lines
4128
4129    032402                          20$:    PRINTF  #TNUM,R1        ; print test number
        032402  010146                                                          MOV     R1,-(SP)
        032404  012746  033106                                                  MOV     #TNUM,-(SP)
        032410  012746  000002                                                  MOV     #2,-(SP)
        032414  010600                                                          MOV     SP,R0
        032416  104417                                                          TRAP    C$PNTF
        032420  062706  000006                                                  ADD     #6,SP
4130    032424                          PRINTF  (R2)            ; and title
        032424  011246                                                          MOV     (R2),-(SP)
        032426  012746  000001                                                  MOV     #1,-(SP)
        032432  010600                                                          MOV     SP,R0
        032434  104417                                                          TRAP    C$PNTF
        032436  062706  000004                                                  ADD     #4,SP
```

REPORT CODING SECTION

```
4131 032442  062702  000002                ADD     #2,R2                   ; get address of next title
4132 032446  005201                        INC     R1                      ; and next test number
4133 032450  020127  000034                CMP     R1,#NTESTS              ; all printed ?
4134 032454  003015                        BGT     30$                     ; if yes, exit
4135 032456  005303                        DEC     R3                      ; 18 lines output ?
4136 032460  001350                        BNE     20$                     ; if not, branch
4137 032462                                GMANID  RET,RFLG,A,377,0,1,YES  ; else wait for operator to read
     032462  104443                                                                TRAP    C$GMAN
     032464  000406                                                                BR      10001$
     032466  032712                                                                .WORD   RFLG
     032470  000152                                                                .WORD   T$CODE
     032472  033050                                                                .WORD   RET
     032474  000377                                                                .WORD   377
     032476  000000                                                                .WORD   T$LOLIM
     032500  000001                                                                .WORD   T$HILIM
     032502                                                                  10001$:
4138 032502  004737  026134                JSR     PC,CRLF                 ; print a line feed
4139 032506  000733                        BR      10$                     ; and then continue
4140
4141 032510  000137  032700          30$:  JMP     PREX                    ; exit
4142
4143 032514                        STAT:   PRINTF  #PR3                    ; print statistics header
     032514  012746  033303                                                        MOV     #PR3,-(SP)
     032520  012746  000001                                                        MOV     #1,-(SP)
     032524  010600                                                                MOV     SP,R0
     032526  104417                                                                TRAP    C$PNTF
     032530  062706  000004                                                        ADD     #4,SP
4144 032534                                PRINTF  #PR3A                   ; ...
     032534  012746  033401                                                        MOV     #PR3A,-(SP)
     032540  012746  000001                                                        MOV     #1,-(SP)
     032544  010600                                                                MOV     SP,R0
     032546  104417                                                                TRAP    C$PNTF
     032550  062706  000004                                                        ADD     #4,SP
4145
4146 032554  005001                        CLR     R1                      ; start with first unit
4147
4148 032556  020137  002012          20$:  CMP     R1,L$UNIT               ; all units reported ?
4149 032562  001444                        BEQ     60$                     ; if yes, exit
4150 032564  010104                        MOV     R1,R4                   ; form offset to error count
4151 032566  006304                        ASL     R4                      ;
4152 032570  016405  002710                MOV     ECNT(R4),R5             ; get unit's error count
4153 032574  005705                        TST     R5                      ; is it negative ?
4154 032576  100423                        BMI     40$                     ; if yes, report untested
4155
4156 032600  012703  033516                MOV     #NO,R3                  ; assume unit is not dropped
4157 032604  105761  002750                TSTB    DROPED(R1)              ; check if it is
4158 032610  001402                        BEQ     30$                     ; if it is not, branch
4159 032612  012703  033512                MOV     #YES,R3                 ; otherwise print yes
4160 032616                          30$:  PRINTF  #PR5,R1,R5,R3           ; else print statistics
     032616  010346                                                                MOV     R3,-(SP)
     032620  010546                                                                MOV     R5,-(SP)
     032622  010146                                                                MOV     R1,-(SP)
     032624  012746  033471                                                        MOV     #PR5,-(SP)
     032630  012746  000004                                                        MOV     #4,-(SP)
     032634  010600                                                                MOV     SP,R0
     032636  104417                                                                TRAP    C$PNTF
     032640  062706  000012                                                        ADD     #12,SP
```

REPORT CODING SECTION

```
4161 032644  000411                        BR      50$                     ; and look for more units
4162
4163 032646                          40$:   PRINTF  #PR4,R1                 ; print 'UNTESTED'
     032646  010146                                                                MOV     R1,-(SP)
     032650  012746  033440                                                        MOV     #PR4,-(SP)
     032654  012746  000002                                                        MOV     #2,-(SP)
     032660  010600                                                                MOV     SP,R0
     032662  104417                                                                TRAP    C$PNTF
     032664  062706  000006                                                        ADD     #6,SP
4164
4165 032670  005201                          50$:   INC     R1                  ; prepare for next unit
4166 032672  000731                                 BR      20$                 ; if not, report the next
4167
4168 032674  004737  026134                 60$:   JSR     PC,CRLF             ; print a line feed
4169
4170 032700                          PREX:  EXIT    RPT
     032700  000167                                                                .WORD   J$JMP
     032702  000616                                                                .WORD   L10023-2-.
4171
4172 032704     110     040     040  CHAR:  .ASCIZ  /H    /                     ; store for operator input
     032707     040     040     000
4173 032712  000000                  RFLG:  .WORD   0                           ; flag for 'TYPE RETURN FOR MORE TITLES
4174
4198
4199 032714                          TADS:  TITLES                              ; list of test title addresses

     032714  035700                        .WORD   TSHD1
     032716  036362                        .WORD   TSHD2
     032720  037344                        .WORD   TSHD3
     032722  040156                        .WORD   TSHD4
     032724  041016                        .WORD   TSHD5
     032726  041734                        .WORD   TSHD6
     032730  042534                        .WORD   TSHD7
     032732  043452                        .WORD   TSHD8
     032734  044310                        .WORD   TSHD9
     032736  045214                        .WORD   TSHD10
     032740  046200                        .WORD   TSHD11
     032742  047162                        .WORD   TSHD12
     032744  050130                        .WORD   TSHD13
     032746  051252                        .WORD   TSHD14
     032750  052250                        .WORD   TSHD15
     032752  053232                        .WORD   TSHD16
     032754  054402                        .WORD   TSHD17
     032756  055414                        .WORD   TSHD18
     032760  056632                        .WORD   TSHD19
     032762  057756                        .WORD   TSHD20
     032764  060722                        .WORD   TSHD21
     032766  061602                        .WORD   TSHD22
     032770  062544                        .WORD   TSHD23
     032772  064036                        .WORD   TSHD24
     032774  066572                        .WORD   TSHD25
     032776  072100                        .WORD   TSHD26
     033000  074757                        .WORD   TSHD27
     033002  076734                        .WORD   TSHD28
4203
4204                                        .NLIST BEX
4205
```

REPORT CODING SECTION

```
   4206 033004      045     116     045  TT:     .ASCII  /%N%ATEST TITLES./
   4207 033024      045     116     045          .ASCIZ  /%N%A------------%N2/
   4208
   4209 033050      124     131     120  RET:    .ASCIZ  /TYPE "RETURN" FOR MORE TITLES/
   4210
   4211 033106      045     104     063  TNUM:   .ASCIZ  /%D3/
   4212
   4213 033112      124     131     120  PR1:    .ASCIZ  /TYPE S,T OR HELP/
   4214
   4215 033133      045     116     045  PR2:    .ASCII  /%N%ATHE FOLLOWING COMMANDS ARE ACCEPTED :-/
   4216 033205      045     116     062          .ASCIZ  /%N2%AS - PRINT STATISTICS TABLE%N/
   4217 033247      045     116     045  PR2A:   .ASCIZ  /%N%AT - PRINT TEST TITLES%N/
   4218
   4219 033303      045     116     062  PR3:    .ASCII  /%N2%AAAF01-A MODULE STATISTICS./
   4220 033342      045     116     045          .ASCIZ  /%N%A-------------------------/
   4221 033401      045     116     062  PR3A:   .ASCIZ  /%N2%AUNIT   ERRORS   DROPPED%N/
   4222
   4223 033440      045     116     045  PR4:    .ASCIZ  /%N%D3%A   UNTESTED    NO/
   4224
   4225 033471      045     116     045  PR5:    .ASCIZ  /%N%D3%S5%D3%S7%T/
   4226
   4227 033512      131     105     123  YES:    .ASCIZ  /YES/
   4228 033516      116     117     000  NO:     .ASCIZ  /NO/
   4229
   4230                                         .LIST BEX
   4231                                         .EVEN
   4232
   4233 033522                                  ENDRPT
        033522                                                  L10023:
        033522  104425                                                  TRAP    C$RPT
```

PROTECTION TABLE

```
                                .SBTTL  PROTECTION TABLE

                                ;++
                                ; THIS TABLE IS USED BY THE RUNTIME SERVICES
                                ; TO PROTECT THE LOAD MEDIA.
                                ;--

033524                                  BGNPROT
033524                                                          L$PROT::

033524  000000                          0               ;OFFSET INTO P-TABLE FOR DRX11-C ADDRESS
033526  177777                          -1              ;OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
033530  177777                          -1              ;OFFSET INTO P-TABLE FOR DRIVE NUMBER

033532                                  ENDPROT
```

INITIALIZE SECTION

```
                                .SBTTL  INITIALIZE SECTION

                                ;++
                                ; THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
                                ; AT THE BEGINNING OF EACH PASS.
                                ;--

033532                                  BGNINIT
033532                                                                  L$INIT::

                                START:  READEF  #EF.START       ; is this a  new start ?
033532  012700  000040                                                  MOV     #EF.START,R0
033536  104447                                                         TRAP    C$REFG
033540                                  BNCOMPLETE RESTRT       ; if not, branch
033540  103064                                                         BCC     RESTRT
033542                                  SETVEC  #100,#CLINT,#340; ignore further interrupts to vector 100
033542  012746  000340                                                 MOV     #340,-(SP)
033546  012746  032214                                                 MOV     #CLINT,-(SP)
033552  012746  000100                                                 MOV     #100,-(SP)
033556  012746  000003                                                 MOV     #3,-(SP)
033562  104437                                                         TRAP    C$SVEC
033564  062706  000010                                                 ADD     #10,SP
033570                                  SETVEC  #14,#113316,#340; *** breakpoint vector for debugging ***
033570  012746  000340                                                 MOV     #340,-(SP)
033574  012746  113316                                                 MOV     #113316,-(SP)
033600  012746  000014                                                 MOV     #14,-(SP)
033604  012746  000003                                                 MOV     #3,-(SP)
033610  104437                                                         TRAP    C$SVEC
033612  062706  000010                                                 ADD     #10,SP
033616                          10$:    BRESET                  ; reset the system
033616  104433                                                         TRAP    C$RESET
033620  004737  034254                  JSR     PC,SETCLK       ; set up clock counter

033624  012700  002774                  MOV     #PFLAG1,R0      ; clear flags for "unipolar/bipolar" printouts
033630  012701  000020                  MOV     #16.,R1         ; after start command
033634  105020                  20$:    CLRB    (R0)+           ; 16 1 byte flags
033636  005301                          DEC     R1              ;
033640  001375                          BNE     20$             ;

033642  005037  003014                  CLR     QFLAG1          ; clear flag for questions after start command
033646  005037  003016                  CLR     QFLAG2          ; clear flag for questions after start, restart
                                                                ; or continue commands

033652  012700  002750                  MOV     #DROPED,R0      ; get unit dropped table adderss
033656  012701  000020                  MOV     #16.,R1         ; there are 16 units
033662  105020                  30$:    CLRB    (R0)+           ; clear all 16 dropped unit flags
033664  005301                          DEC     R1              ; ...
033666  001375                          BNE     30$             ; ...

033670  012700  002710                  MOV     #ECNT,R0        ; get error count for uut 0
033674  012701  000020                  MOV     #16.,R1         ; there are 16 uut's
033700  012720  100000          40$:    MOV     #100000,(R0)+   ; initialize the error count
033704  005301                          DEC     R1              ; all cleared ?
033706  001374                          BNE     40$             ; if not, clear next count

033710  000427                          BR      INIUUT          ; and start testing with first uut
```

INITIALIZE SECTION

```
   4327
   4328 033712                          RESTRT: READEF  #EF.RESTART     ; is this a restart ?
        033712  012700  000037                                                  MOV     #EF.RESTART,R0
        033716  104447                                                          TRAP    C$REFG
   4329 033720                                  BNCOMPLETE NEWST         ; if not, branch
        033720  103003                                                          BCC     NEWST
   4330 033722  005037  003016                  CLR     QFLAG2           ; else flag some questions should be asked
   4331 033726  000420                          BR      INIUUT           ; and start with first uut
   4332
   4333 033730                          NEWST:  READEF  #EF.NEW          ; is this a new pass ?
        033730  012700  000035                                                  MOV     #EF.NEW,R0
        033734  104447                                                          TRAP    C$REFG
   4334 033736                                  BNCOMPLETE CONT          ; if not, branch
        033736  103000                                                          BCC     CONT
   4335
   4336 033740                          CONT:   READEF  #EF.CONTINUE     ; is this a continue ?
        033740  012700  000036                                                  MOV     #EF.CONTINUE,R0
        033744  104447                                                          TRAP    C$REFG
   4337 033746                                  BNCOMPLETE PWRFL         ; if not, branch
        033746  103003                                                          BCC     PWRFL
   4338 033750  005037  003016                  CLR     QFLAG2           ; else flag some questions should be asked
   4339 033754  000466                          BR      END              ; and continue
   4340
   4341 033756                          PWRFL:  READEF  #EF.PWR          ; is this a power fail
        033756  012700  000034                                                  MOV     #EF.PWR,R0
        033762  104447                                                          TRAP    C$REFG
   4342 033764                                  BCOMPLETE END            ; if yes, just continue
        033764  103462                                                          BCS     END
   4343
   4344 033766  000405                          BR      NXTUUT           ; else test next uut
   4345
   4346 033770  012737  177777  002074  INIUUT: MOV     #-1,L$LUN        ; initialize logical unit number
   4347 033776  005037  034250                  CLR     UNITS            ; no units dropped yet
   4348 034002  005237  002074          NXTUUT: INC     L$LUN            ; next logical unit to be tested ?
   4349 034006  023737  002074  002012          CMP     L$LUN,L$UNIT     ; all units tried ?
   4350 034014  002365                          BGE     INIUUT           ; if yes, start again
   4351 034016                                  GPHARD  L$LUN,R1         ; get parameter table address in R1
        034016  013700  002074                                                  MOV     L$LUN,R0
        034022  104442                                                          TRAP    C$GPHRD
        034024  010001                                                          MOV     R0,R1
   4352 034026                                  BCOMPLETE 10$            ; if not dropped, set up offsets
        034026  103407                                                          BCS     10$
   4353 034030  005237  034250                  INC     UNITS            ; else flag another unit dropped
   4354 034034  023737  034250  002012          CMP     UNITS,L$UNIT     ; all dropped ?
   4355 034042  001500                          BEQ     END1             ; if yes, exit
   4356 034044  000756                          BR      NXTUUT           ; else get the next unit
   4357
   4358 034046  013700  002074          10$:    MOV     L$LUN,R0         ; get unit number
   4359 034052  006300                          ASL     R0               ; convert to an offset
   4360 034054  042760  100000  002710          BIC     #100000,ECNT(R0); flag unit is being tested
   4361
   4362 034062  012100                          MOV     (R1)+,R0         ; get first hardware parameter
   4363 034064  010037  002626                  MOV     R0,DRXSCR        ; save as new SCR address
   4364 034070  062700  000002                  ADD     #2,R0            ; add 2
   4365 034074  010037  002630                  MOV     R0,DRXCOR        ; to give new COR address
   4366 034100  062700  000002                  ADD     #2,R0            ; add 2
   4367 034104  010037  002632                  MOV     R0,DRXADR        ; to give new ADR address
```

INITIALIZE SECTION

```
4368 034110  062700  000002                ADD     #2,R0           ; add 2
4369 034114  010037  002634                MOV     R0,DRXDBR       ; to give new DBR address
4370
4371 034120  011137  002636                MOV     (R1),DRXVEC     ; save new vector address
4372
4373 034124  012737  000001  003030        MOV     #1,STAFLG       ; flag start of pass
4374
4375 034132  005037  003026          END:  CLR     MODE            ; assume module is in unipolar mode
4376 034136                                SETVEC  #4,#NXM,#340    ; set NXM vector in case there's no module
     034136  012746  000340                                                  MOV     #340,-(SP)
     034142  012746  032204                                                  MOV     #NXM,-(SP)
     034146  012746  000004                                                  MOV     #4,-(SP)
     034152  012746  000003                                                  MOV     #3,-(SP)
     034156  104437                                                          TRAP    C$SVEC
     034160  062706  000010                                                  ADD     #10,SP
4377 034164  012777  010000  146442        MOV     #ACS,@DRXDBR    ; address the ACS register
4378 034172  017701  146436                MOV     @DRXDBR,R1      ; read the ACS
4379 034176                                CLRVEC  #4              ; restore the DRS NXM trap catcher
     034176  012700  000004                                                  MOV     #4,R0
     034202  104436                                                          TRAP    C$CVEC
4380 034204  032701  004000                BIT     #UNI,R1         ; module in unipolar mode ?
4381 034210  001002                        BNE     10$             ; if yes, branch
4382 034212  005237  003026                INC     MODE            ; else flag bipolar
4383
4384 034216  023737  002226  002224  10$:  CMP     LSTCHN,FSTCHN   ; is last channel greater than the first ?
4385 034224  003007                        BGT     END1            ; if yes, branch
4386 034226  013700  002224                MOV     FSTCHN,R0       ; else, swap the first and last
4387 034232  013737  002226  002224        MOV     LSTCHN,FSTCHN   ; channel numbers
4388 034240  010037  002226                MOV     R0,LSTCHN       ;
4389
4390 034244                          END1: EXIT    INIT
     034244  104432                                                  TRAP    C$EXIT
     034246  000004                                                  .WORD   L10025-.
4391
4392 034250  000000                  UNITS: .WORD   0               ; units dropped count - this prevents infinite
4393                                                                 ; looping in the INIT code if all units are
4394                                                                 ; dropped.
4395
4396 034252                                ENDINIT
     034252                                                       L10025:
     034252  104411                                                          TRAP    C$INIT
4397
```

```
                        ;*****************************************
                        ; SUBROUTINES USED DURING INITIALISATION. *
                        ;*****************************************

                        .SBTTL  SETCLK - Routine to set up delay counts

                        ;++
                        ; functional description :
                        ;
                        ;       This routine sets up 3 delay variables called CNT25M, CNT500, and
                        ;       CNT25. These give delays of approximately 25 milliseconds, 500
                        ;       microseconds or 25 microseconds respectively if used as follows :
                        ;
                        ;               MOV     CNTXXX,R0
                        ;           1$: DEC     R0
                        ;               BNE     1$
                        ;
                        ;       The counts are derived from an L clock if there is one.
                        ;       Otherwise, the operator is asked to type 2 characters on the
                        ;       console 6 seconds apart.
                        ;
                        ; Inputs :
                        ;
                        ;       None.
                        ;
                        ; Implicit inputs :
                        ;
                        ;       If CNT25M is not zero (already set up), the routine does nothing.
                        ;
                        ; Outputs :
                        ;
                        ;       Console message if there is no L clock on the system.
                        ;
                        ; Implicit outputs :
                        ;
                        ;       CNT25M contains the count required for 25 milliseconds.
                        ;       CNT500 contains the count required for 500 microseconds.
                        ;       CNT25 contains the count required for 25 microseconds.
                        ;
                        ; Subordinate routines used :
                        ;
                        ;       CRLF  - line feed print routine.
                        ;       CLINT - dummy clock interrupt service routine
                        ;
                        ; Functional side effects :
                        ;
                        ;       R0 to R5 are corrupted.
                        ;
                        ;       If a line time clock is found, vector 100 is set up so that
                        ;       interrupts to it are ignored. The SETVEC macro can be used to
                        ;       set up the vector for a device interrupt.
                        ;
                        ; Calling sequence :
                        ;
                        ;       JSR     PC,SETCLK
                        ;
                        ;--
```

SETCLK - Routine to set up delay counts

```
 4456
 4457 034254  005737  026126          SETCLK: TST     CNT25M          ; counters already set up?
 4458 034260  001402                          BEQ     10$             ; if not, branch
 4459 034262  000137  035130                  JMP     SETEX           ; if yes, exit
 4460
 4461 034266  005004                  10$:    CLR     R4              ; clear a counter
 4462 034270                                  GETPRI  R2              ; save current priority in R2
      034270  104440                                                                  TRAP    C$GPRI
      034272  010002                                                                  MOV     R0,R2
 4463 034274  005037  035132                  CLR     CLKFLG          ; assume there is no clock with a CSR
 4464 034300                                  CLOCK   L,R1            ; get address of clock table
      034300  012700  000114                                                          MOV     #'L,R0
      034304  104462                                                                  TRAP    C$CLCK
      034306  010001                                                                  MOV     R0,R1
 4465 034310                                  SETVEC  #4,#NXM,#340    ; set up clock CSA trap
      034310  012746  000340                                                          MOV     #340,-(SP)
      034314  012746  032204                                                          MOV     #NXM,-(SP)
      034320  012746  000004                                                          MOV     #4,-(SP)
      034324  012746  000003                                                          MOV     #3,-(SP)
      034330  104437                                                                  TRAP    C$SVEC
      034332  062706  000010                                                          ADD     #10,SP
 4466 034336  005037  002770                  CLR     NXMFLG          ; clear NXM flag
 4467 034342  005771  000000                  TST     @(R1)           ; access the clock address
 4468 034346  005737  002770                  TST     NXMFLG          ; *don't delete, needed for falcon*
 4469 034352  005737  002770                  TST     NXMFLG          ; does the clock have a register ?
 4470 034356  001005                          BNE     LCLOCK          ; if not, branch
 4471 034360  005237  035132                  INC     CLKFLG          ; else flag there is a clock CSR
 4472 034364  012771  000100  000000         MOV     #100,@(R1)      ; and set it up to interrupt
 4473                                  ;
 4474                                  ; Use the L clock
 4475                                  ;
 4476 034372                          LCLOCK: CLRVEC  #4              ; set vector 4 to unused pool
      034372  012700  000004                                                          MOV     #4,R0
      034376  104436                                                                  TRAP    C$CVEC
 4477 034400  012703  000006                  MOV     #6,R3           ; if 50 hz, 100 ms = 5 interrupts
 4478 034404  026127  000006  000062         CMP     6(R1),#50.      ; 50 hz correct?
 4479 034412  001401                          BEQ     10$             ; if yes, branch
 4480 034414  005203                          INC     R3              ; else allow 6 interrupts
 4481
 4482 034416  010305                  10$:    MOV     R3,R5           ; save number of interrrupts
 4483 034420                                  SETVEC  #100,#KLINT,#340; set up the clock vector
      034420  012746  000340                                                          MOV     #340,-(SP)
      034424  012746  034506                                                          MOV     #KLINT,-(SP)
      034430  012746  000100                                                          MOV     #100,-(SP)
      034434  012746  000003                                                          MOV     #3,-(SP)
      034440  104437                                                                  TRAP    C$SVEC
      034442  062706  000010                                                          ADD     #10,SP
 4484                                                                 ; to wait for 1st interrupt
 4485 034446                                  SETPRI  #0              ; and drop the priority
      034446  012700  000000                                                          MOV     #0,R0
      034452  104441                                                                  TRAP    C$SPRI
 4486 034454  005000                          CLR     R0              ; clear R0 and the carry bit
 4487 034456  020305                  20$:    CMP     R3,R5           ; has count been dropped ?
 4488 034460  001004                          BNE     30$             ; if yes, start the counters
 4489 034462  005300                          DEC     R0              ; waited too long ?
 4490 034464  001374                          BNE     20$             ; if not, wait longer
 4491 034466  000137  034600                  JMP     USCLOK          ; if yes, assume no clock
```

SETCLK - Routine to set up delay counts

```
4492
4493 034472  005005                  30$:    CLR     R5              ; clear the high counter
4494 034474  005204                  40$:    INC     R4              ; count the delay for 5 or 6 interrupts
4495 034476  001376                          BNE     40$             ;
4496 034500  105205                          INCB    R5              ;
4497 034502  001374                          BNE     40$             ;
4498 034504  000435                          BR      USCLOK          ; if too long, assume no clock
4499
4500 034506  005303                  KLINT:  DEC     R3              ; 5 or 6 interrupts?
4501 034510  001401                          BEQ     40$             ; if yes, tidy up
4502 034512  000002                          RTI                     ; else keep counting
4503
4504 034514                          40$:    SETPRI  R2              ; restore the priority
     034514  010200                                                                  MOV     R2,R0
     034516  104441                                                                  TRAP    C$SPRI
4505 034520                                  SETVEC  #100,#CLINT,#340; ignore further interrupts to vector 100
     034520  012746  000340                                                          MOV     #340,-(SP)
     034524  012746  032214                                                          MOV     #CLINT,-(SP)
     034530  012746  000100                                                          MOV     #100,-(SP)
     034534  012746  000003                                                          MOV     #3,-(SP)
     034540  104437                                                                  TRAP    C$SVEC
     034542  062706  000010                                                          ADD     #10,SP
4506 034546  022626                          CMP     (SP)+,(SP)+     ; tidy up the stack`
4507 034550  005737  035132                  TST     CLKFLG          ; can we disable a clock ?
4508 034554  001402                          BEQ     50$             ; if not, branch
4509 03'556  005071  000000                  CLR     @(R1)           ; else, disable clock interrupts
4510
4511 034562  000241                  50$:    CLC                     ; divide the 100 millisecond counters
4512 034564  006005                          ROR     R5              ; by 4 to give 25 milliseconds
4513 034566  006004                          ROR     R4              ;
4514 034570  000241                          CLC                     ;
4515 034572  006005                          ROR     R5              ;
4516 034574  006004                          ROR     R4              ;
4517 034576  000524                          BR      SAVCNT          ; and save the count
4518                                 ;
4519                                 ; Come here if not enough clock interrupts occur before the counters overflow
4520                                 ;
4521 034600                          USCLOK: SETPRI  R2              ; restore the priority
     034600  010200                                                                  MOV     R2,R0
     034602  104441                                                                  TRAP    C$SPRI
4522 034604                                  SETVEC  #100,#CLINT,#340; ignore further interrupts to vector 100
     034604  012746  000340                                                          MOV     #340,-(SP)
     034610  012746  032214                                                          MOV     #CLINT,-(SP)
     034614  012746  000100                                                          MOV     #100,-(SP)
     034620  012746  000003                                                          MOV     #3,-(SP)
     034624  104437                                                                  TRAP    C$SVEC
     034626  062706  000010                                                          ADD     #10,SP
4523 034632  005737  035132                  TST     CLKFLG          ; can we disable a clock ?
4524 034636  001402                          BEQ     NOCLOK          ; if not, branch
4525 034640  005071  000000                  CLR     @(R1)           ; else disable clock interrupts
4526                                 ;
4527                                 ; Use the console for timing
4528                                 ;
4529         177560                          TKS=177560              ; keyboard status register
4530         177562                          TKB=177562              ; keyboard data buffer
4531         177564                          TPS=177564              ; printer status register
4532         177566                          TPB=177566              ; printer data buffer
```

SETCLK - Routine to set up delay counts

```
4533
4534 034644                          NOCLOK: SETVEC  #60,#TTINT,#340 ; set up interrupt vector
     034644  012746  000340                                                MOV     #340,-(SP)
     034650  012746  034776                                                MOV     #TTINT,-(SP)
     034654  012746  000060                                                MOV     #60,-(SP)
     034660  012746  000003                                                MOV     #3,-(SP)
     034664  104437                                                       TRAP    C$SVEC
     034666  062706  000010                                                ADD     #10,SP
4535 034672                                  PRINTF  #TIMMSG         ; 'TYPE 2 CHARACTERS 6 SECONDS APART'
     034672  012746  035134                                                MOV     #TIMMSG,-(SP)
     034676  012746  000001                                                MOV     #1,-(SP)
     034702  010600                                                       MOV     SP,R0
     034704  104417                                                       TRAP    C$PNTF
     034706  062706  000004                                                ADD     #4,SP
4536
4537 034712  105737  177560          10$:    TSTB    TKS             ; is first character ready?
4538 034716  100375                  10$     BPL     10$             ; if not, wait
4539 034720  013700  177562                  MOV     TKB,R0          ; else get the character
4540 034724  042700  177600                  BIC     #177600,R0      ; discard unwanted bits
4541 034730  020027  000003                  CMP     R0,#3           ; if ^C, return to supervisor
4542 034734  001001                          BNE     20$             ;
4543 034736                                  DOCLN                   ;
     034736  104444                                                       TRAP    C$DCLN
4544
4545 034740  013737  177562  177566  20$:    MOV     TKB,TPB         ; now echo the character
4546 034746                                  SETPRI  #0              ; drop the priority
     034746  012700  000000                                                MOV     #0,R0
     034752  104441                                                       TRAP    C$SPRI
4547 034754  012737  000100  177560          MOV     #100,TKS        ; allow interrupts
4548
4549 034762  012705  000360          30$:    MOV     #240.,R5        ; set up modulo 240 counter
4550 034766  005305                  40$:    DEC     R5              ; start counting
4551 034770  001376                          BNE     40$             ; r5 is modulo 240 counter
4552 034772  005204                          INC     R4              ; update the counter
4553 034774  000772                          BR      30$             ; 6 seconds/240 = 25 milliseconds
4554
4555 034776                          TTINT:  SETPRI  R2              ; restore the priority
     034776  010200                                                       MOV     R2,R0
     035000  104441                                                       TRAP    C$SPRI
4556 035002                                  CLRVEC  #60             ; and the keyboard vector
     035002  012700  000060                                                MOV     #60,R0
     035006  104436                                                       TRAP    C$CVEC
4557 035010  022626                          CMP     (SP)+,(SP)+     ; tidy up the stack
4558 035012  005037  177560                  CLR     TKS             ; disable interrupts
4559 035016  013700  177562                  MOV     TKB,R0          ; else get the character
4560 035022  042700  177600                  BIC     #177600,R0      ; discard unwanted bits
4561 035026  020027  000003                  CMP     R0,#3           ; if ^C, return to supervisor
4562 035032  001001                          BNE     10$             ;
4563 035034                                  DOCLN                   ;
     035034  104444                                                       TRAP    C$DCLN
4564 035036  013737  177562  177566  10$:    MOV     TKB,TPB         ; else, echo the character
4565 035044  004737  026134                  JSR     PC,CRLF         ; and print a line feed
4566                                 ;
4567                                 ; SAVE THE COUNTERS
4568                                 ;
4569 035050  010437  026126          SAVCNT: MOV     R4,CNT25M       ; save the 25 milliseconds counter
4570 035054  012700  000062                  MOV     #50.,R0         ; now divide by 50
```

SETCLK - Routine to set up delay counts

```
4571 035060  062704  000031            ADD     #25.,R4         ; to nearest 50
4572 035064  005001                    CLR     R1              ; initialise result
4573 035066  160004            10$:    SUB     R0,R4           ; remainder < 0 ?
4574 035070  002402                    BLT     20$             ; if yes, branch
4575 035072  005201                    INC     R1              ; else increment result
4576 035074  000774                    BR      10$             ; and try again
4577 035076  010137  026130    20$:    MOV     R1,CNT500       ; save the 500 microseconds counter
4578
4579 035102  012700  000024            MOV     #20.,R0         ; now divide by 20
4580 035106  062701  000012            ADD     #10.,R1         ; to nearest 20
4581 035112  005002                    CLR     R2              ; initialise result
4582 035114  160001            30$:    SUB     R0,R1           ; remainder < 0 ?
4583 035116  002402                    BLT     40$             ; if yes, branch
4584 035120  005202                    INC     R2              ; else increment result
4585 035122  000774                    BR      30$             ; and try again
4586 035124  010237  026132    40$:    MOV     R2,CNT25        ; save the 25 microseconds counter
4587
4588 035130  000207            SETEX:  RTS     PC              ; return
4589
4590 035132  000000            CLKFLG: .WORD   0               ; set if DRS finds a clock with a CSR
4591
4592                                   .NLIST  BEX
4593 035134     045     116     045   TIMMSG: .ASCIZ  /%N%ATYPE 2 CHARACTERS 6 SECONDS APART >/
4594                                   .LIST   BEX
4595                                   .EVEN
```

AUTODROP SECTION

```
                                .SBTTL  AUTODROP SECTION

                                ;++
                                ; THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
                                ; THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
                                ; SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
                                ; DROPPED FROM TESTING.
                                ;--

035204                                  BGNAUTO
035204                                                                  L$AUTO::

                                        SETVEC  #4,#NXM,#PRI07  ; set up non - existent memory trap vector.
035204  012746  000340                                                  MOV     #PRI07,-(SP)
035210  012746  032204                                                  MOV     #NXM,-(SP)
035214  012746  000004                                                  MOV     #4,-(SP)
035220  012746  000003                                                  MOV     #3,-(SP)
035224  104437                                                          TRAP    C$SVEC
035226  062706  000010                                                  ADD     #10,SP
035232  005037  002770                CLR     NXMFLG          ; clear non - existent memory flag
035236  005777  145364                TST     @DRXSCR         ; reference memory address for the DRX11-C
                                                                ; to see if it exists.

                                ; If the device doesn't exist, the resultant trap to vector 04 will
                                ; cause the flag NXMFLG to be set (see interrupt routine NXM).

035242  005737  002770                  TST     NXMFLG          ; was there a trap ?
035246  001403                         BEQ     10$             ; branch if not
035250                                  DODU    L$LUN           ; else drop the device
035250  013700  002074                                                  MOV     L$LUN,R0
035254  104451                                                          TRAP    C$DODU
035256                          10$:    CLRVEC  #4              ; return vector 4 to normal state
035256  012700  000004                                                  MOV     #4,R0
035262  104436                                                          TRAP    C$CVEC

035264                                  ENDAUTO
035264                                                                  L10026:
035264  104461                                                          TRAP    C$AUTO
```

CLEANUP CODING SECTION

```
                                .SBTTL  CLEANUP CODING SECTION

                                ;++
                                ; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
                                ; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
                                ;--

                                        BGNCLN
                                                                                L$CLEAN::
                                        SETPRI  #PRI07                  ; stop any more interrupts
                                                                                MOV     #PRI07,R0
                                                                                TRAP    C$SPRI
                                        CLRVEC  DRXVEC                  ; restore the DRS trap catcher
                                                                                MOV     DRXVEC,R0
                                                                                TRAP    C$CVEC
                                        CMP     #25.,L$TEST             ; are we coming from test 25 ?
                                        BNE     10$                     ; branch if not
                                        SETVEC  #4,#NXM,#PRI07          ; set up non - existent memory vector.
                                                                                MOV     #PRI07,-(SP)
                                                                                MOV     #NXM,-(SP)
                                                                                MOV     #4,-(SP)
                                                                                MOV     #3,-(SP)
                                                                                TRAP    C$SVEC
                                                                                ADD     #10,SP
                                        BIC     #100,@DRXCOR            ; finish up DMA
                                        CLRVEC  #4                      ; return vector 04 to normal state
                                                                                MOV     #4,R0
                                                                                TRAP    C$CVEC
                                10$:    EXIT    CLN
                                                                                TRAP    C$EXIT
                                                                                .WORD   L10027-.

                                        ENDCLN
                                                                        L10027:
                                                                                TRAP    C$CLEAN
```

DROP UNIT SECTION

```
                                .SBTTL  DROP UNIT SECTION

                                ;++
                                ; THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
                                ; TO NO LONGER BE TESTED.
                                ;--

035362                                  BGNDU
035362                                                                  L$DU::

035362  112760  000001  002750          MOVB    #1,DROPED(R0)           ; flag unit dropped in param table
035370                                  PRINTF  #DROPD,R0               ; 'UNIT DROPPED'
035370  010046                                                                  MOV     R0,-(SP)
035372  012746  035416                                                          MOV     #DROPD,-(SP)
035376  012746  000002                                                          MOV     #2,-(SP)
035402  010600                                                                  MOV     SP,R0
035404  104417                                                                  TRAP    C$PNTF
035406  062706  000006                                                          ADD     #6,SP

035412                                  EXIT    DU
035412  000167                                                                  .WORD   J$JMP
035414  000030                                                                  .WORD   L10030-2-.

                                .NLIST  BEX
035416      045     116     045 DROPD:  .ASCIZ  /%N%AUNIT %D2%A DROPPED/
                                .LIST   BEX
                                .EVEN

035446                                  ENDDU
035446                                                                  L10030:
035446  104453                                                                  TRAP    C$DU
```

ADD UNIT SECTION

```
                                .SBTTL  ADD UNIT SECTION

                                ;++
                                ; THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
                                ;  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
                                ;  TO THE TEST CYCLE.
                                ;--

035450                                  BGNAU
035450                                                                  L$AU::

035450  105060  002750                  CLRB    DROPED(R0)              ; flag unit not dropped in param table

035454                                  EXIT    AU
035454  000167                                                                  .WORD   J$JMP
035456  000000                                                                  .WORD   L10031-2-.

035460                                  ENDAU
035460                                                                  L10031:
035460  104452                                                                  TRAP    C$AU

035462                                  ENDMOD
```

ADD UNIT SECTION

```
   4688                                 .TITLE HARDWARE TESTS
   4699
   4726 035462                                  BGNMOD
```

TEST 1 - DRX11-C Register NXM Test.

```
                                .SBTTL  TEST 1 - DRX11-C Register NXM Test.

                                ;++
                                ; This test checks that accessing the DRX11-C SCR, COR, ADR and
                                ; DBR registers does not cause a NXM trap.
                                ;--

                                        BGNTST
                                                                                T1::
                                        CLR     STAFLG                  ; flag test has been run since start
                                        RFLAGS  R0                      ; read operator flags
                                                                                TRAP    C$RFLA
                                        BIT     #PNT,R0                 ; print test headers ?
                                        BEQ     10$                     ; if not, branch
                                        PRINTF  #TSHD1                  ; else, print test header
                                                                                MOV     #TSHD1,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP

                                10$:    CLR     ITRCNT                  ; clear iteration counter
                                        SETVEC  #4,#NXM,#PRI07          ; set up NXM trap service routine
                                                                                MOV     #PRI07,-(SP)
                                                                                MOV     #NXM,-(SP)
                                                                                MOV     #4,-(SP)
                                                                                MOV     #3,-(SP)
                                                                                TRAP    C$SVEC
                                                                                ADD     #10,SP

                                20$:    MOV     DRXSCR,R1               ; get first register address
                                        MOV     #4,R2                   ; test 4 registers
                                        CLR     R3                      ; clear the error flag

                                30$:    BGNSEG                          ;
                                                                                TRAP    C$BSEG
                                        CLR     NXMFLG                  ; clear the NXM flag
                                        TST     (R1)                    ; test register address
                                        TST     NXMFLG                  ; was there a trap ?
                                        CALL    INSERT                  ; skip branch if error insert selected
                                        BEQ     40$                     ; if no trap, branch
                                        INC     R3                      ; else flag the error
                                        ERRHRD  100,E100,ERDNR          ; print "ADDRESSING ERROR"
                                                                                TRAP    C$ERHRD
                                                                                .WORD   100
                                                                                .WORD   E100
                                                                                .WORD   ERDNR
                                40$:    ENDSEG                          ;
                                                                              10000$:
                                                                                TRAP    C$ESEG

                                        ADD     #2,R1                   ; point to next register
                                        DEC     R2                      ; all registers tested ?
                                        BNE     30$                     ; if not, branch

                                        TST     R3                      ; was there an error ?
                                        BEQ     50$                     ; if not, branch
```

TEST 1 - DRX11-C Register NXM Test.

```
4765 035634                                DODU    L$LUN                   ; else drop the unit under test
     035634  013700  002074                                                        MOV     L$LUN,R0
     035640  104451                                                             TRAP    C$DODU
4766 035642                                DOCLN                           ; and run the clean up routine
     035642  104444                                                             TRAP    C$DCLN
4767
4768 035644  005737  002234        50$:    TST     QVMODE                  ; is quick verify mode selected ?
4769 035650  001006                        BNE     60$                     ; if yes, exit test
4770 035652  005237  003032                INC     ITRCNT                  ; else, increment iteration counter
4771 035656  023727  003032  000010        CMP     ITRCNT,#10              ; iterations completed ?
4772 035664  001331                        BNE     20$                     ; if not, do another iteration
4773
4774 035666                        60$:    CLRVEC  #4                      ; restore DRS trap catcher
     035666  012700  000004                                                        MOV     #4,R0
     035672  104436                                                             TRAP    C$CVEC
4775 035674                                EXIT    TST                     ; and exit the test
     035674  104432                                                             TRAP    C$EXIT
     035676  000076                                                             .WORD   L10032-.
4776
4777                                       .NLIST  BEX
4778 035700    045     123     062  TSHD1:: .ASCIZ  /%S2%AREGISTER NXM TEST%N/
4779 035731    104     122     130  E100:   .ASCIZ  /DRX11-C REGISTER ADDRESSING ERROR/
4780                                       .LIST   BEX
4781                                       .EVEN
4782
4783 035774                                ENDTST
     035774                                                     L10032:
     035774  104401                                                       TRAP    C$ETST
```

```
                                .SBTTL  TEST 2 - Reset Test

                                ;++
                                ; This test checks that the AAF01-A ACS register is correctly set
                                ; or reset after a DRX11-C reset and after a data system reset.
                                ; Both are checked to complete within 1 millisecond.
                                ;
                                ; Bit 11 of the ACS register is read and a message is output to
                                ; show whether it indicates unipolar or bipolar output mode eg. :
                                ;
                                ; "MODULE IS SWITCHED TO UNIPOLAR MODE".
                                ;
                                ;--
035776                                  BGNTST
035776                                                                  T2::
035776  005037  003030                   CLR     STAFLG                  ; flag test has been run since start
036002                                  RFLAGS  R0                      ; read operator flags
036002  104421                                                                 TRAP    C$RFLA
036004  032700  001000                   BIT     #PNT,R0                 ; print test headers ?
036010  001410                           BEQ     10$                     ; if not, branch
036012                                  PRINTF  #TSHD2                  ; else, print test header
036012  012746  036362                                                          MOV     #TSHD2,-(SP)
036016  012746  000001                                                          MOV     #1,-(SP)
036022  010600                                                                  MOV     SP,R0
036024  104417                                                                  TRAP    C$PNTF
036026  062706  000004                                                          ADD     #4,SP
036032  005037  003032           10$:    CLR     ITRCNT                  ; clear iteration counter

036036  013700  002074                   MOV     L$LUN,R0                ; get unit number
036042  105760  002774                   TSTB    PFLAG1(R0)              ; has message already been printed ?
036046  001026                           BNE     30$                     ; if yes, branch
036050  105260  002774                   INCB    PFLAG1(R0)              ; else flag we are printing it now
036054  005737  003026                   TST     MODE                    ; module in unipolar mode ?
036060  001011                           BNE     20$                     ; if not, branch
036062                                  PRINTF  #UNIMES                 ; else, print "UNIPOLAR MODE"
036062  012746  036572                                                          MOV     #UNIMES,-(SP)
036066  012746  000001                                                          MOV     #1,-(SP)
036072  010600                                                                  MOV     SP,R0
036074  104417                                                                  TRAP    C$PNTF
036076  062706  000004                                                          ADD     #4,SP
036102  000410                           BR      30$                     ; and branch
036104                          20$:    PRINTF  #BIMES                  ; print "BIPOLAR  ODE"
036104  012746  036644                                                          MOV     #BIMES,-(SP)
036110  012746  000001                                                          MOV     #1,-(SP)
036114  010600                                                                  MOV     SP,R0
036116  104417                                                                  TRAP    C$PNTF
036120  062706  000004                                                          ADD     #4,SP

036124                          30$:    BGNSEG
036124  104404                                                                  TRAP    C$BSEG
036126  052777  000040  144472           BIS     #RES,@DRXSCR            ; do a device reset
036134  004737  026106                   JSR     PC,WT500                ; wait for 1 millisecond
036140  004737  026106                   JSR     PC,WT500                ; (2*500 microseconds)
036144  012737  010000  003020           MOV     #ACS,GOOD               ; save expected ACS contents
036152  012777  010000  144454           MOV     #ACS,@DRXDBR            ; set DBR to address ACS
036160  017737  144450  003022           MOV     @DRXDBR,BAD             ; read ACS register
036166  013700  003022                   MOV     BAD,R0                  ; get the contents into R0
```

TEST 2 - Reset Test

```
4824 036172  042700  171777          BIC     #↑C<UNI!CMP>,R0         ; and isolate uncertain bits
4825 036176  050037  003020          BIS     R0,GOOD                 ; set them in GOOD if they are set
4826 036202  023737  003022  003020  CMP     BAD,GOOD                ; contents correct ?
4827 036210  004737  026306          CALL    INSERT                  ; skip branch if error insert selected
4828 036214  001404                  BEQ     40$                     ; branch if ACS is OK
4829 036216                          ERRSOFT 200,E200,EGB            ; print "ACS REGISTER INCORRECT"
     036216  104457                                                          TRAP    C$ERSOFT
     036220  000310                                                          .WORD   200
     036222  036404                                                          .WORD   E200
     036224  023416                                                          .WORD   EGB
4830 036226                  40$:    ENDSEG                          ;           "AFTER DRX11-C RESET"
     036226                                                                   10000$:
     036226  104405                                                          TRAP    C$ESEG
4831
4832 036230                  50$:    BGNSEG                                  TRAP    C$BSEG
     036230  104404
4833 036232  012777  010020  144374  MOV     #ACS!DSR,@DRXDBR        ; set up for a data system reset
4834 036240  052777  000001  144360  BIS     #FNCT0,@DRXSCR          ; transfer to ACS
4835 036246  004737  026106          JSR     PC,WT500                ; wait for 1 millisecond
4836 036252  004737  026106          JSR     PC,WT500                ; (2*500 microseconds)
4837 036256  012737  010000  003020  MOV     #ACS,GOOD               ; save expected ACS contents
4838 036264  017737  144344  003022  MOV     @DRXDBR,BAD             ; read ACS register
4839 036272  013700  003022          MOV     BAD,R0                  ; get the contents into R0
4840 036276  042700  171777          BIC     #↑C<UNI!CMP>,R0         ; and isolate uncertain bits
4841 036302  050037  003020          BIS     R0,GOOD                 ; set them in GOOD if they are set
4842 036306  023737  003022  003020  CMP     BAD,GOOD                ; contents correct ?
4843 036314  004737  026306          CALL    INSERT                  ; skip branch if error insert selected
4844 036320  001404                  BEQ     60$                     ; branch if ACS is OK
4845 036322                          ERRSOFT 201,E201,EGB            ; print "ACS REGISTER INCORRECT"
     036322  104457                                                          TRAP    C$ERSOFT
     036324  000311                                                          .WORD   201
     036326  036475                                                          .WORD   E201
     036330  023416                                                          .WORD   EGB
4846 036332                  60$:    ENDSEG                          ;       "AFTER DATA SYSTEM RESET"
     036332                                                                   10001$:
     036332  104405                                                          TRAP    C$ESEG
4847
4848 036334  005737  002234          TST     QVMODE                  ; is quick verify mode selected ?
4849 036340  001006                  BNE     70$                     ; if yes, exit test
4850 036342  005237  003032          INC     ITRCNT                  ; else, increment iteration counter
4851 036346  023727  003032  000010  CMP     ITRCNT,#10              ; iterations completed ?
4852 036354  001263                  BNE     30$                     ; if not, do another iteration
4853
4854 036356                  70$:    EXIT    TST
     036356  104432                                                          TRAP    C$EXIT
     036360  000336                                                          .WORD   L10033-.
4855
4856                                 .NLIST  BEX
4857 036362      045     123     062 TSHD2:: .ASCIZ  /%S2%ARESET TEST%N/
4858 036404      101     103     123 E200:   .ASCIZ  /ACS REGISTER INCORRECT 1 MILLISECOND AFTER DRX11-C RESET/
4859 036475      101     103     123 E201:   .ASCIZ  /ACS REGISTER INCORRECT 1 MILLISECOND AFTER DATA SYSTEM RESET/
4860 036572      045     116     045 UNIMES: .ASCIZ  /%N%AMODULE IS SWITCHED TO UNIPOLAR MODE%N/
4861 036644      045     116     045 BIMES:  .ASCIZ  /%N%AMODULE IS SWITCHED TO BIPOLAR MODE%N/
4862                                 .LIST   BEX
4863                                 .EVEN
4864
4865 036716                          ENDTST
```

TEST 2 - Reset Test

SEQ 0125

```
        036716
        036716  104401                          L10033:
                                                        TRAP    C$ETST
```

TEST 3 - ACS Read/write test.

```
                                .SBTTL  TEST 3 - ACS Read/write test.

                                ;++
                                ; This test checks that the read/write bits of the ACS register
                                ; can all be set, all cleared and individually set.
                                ;--
036720                                  BGNTST
036720                                                                          T3::
036720  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
036724                                  RFLAGS  R0                      ; read operator flags
036724  104421                                                                  TRAP    C$RFLA
036726  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
036732  001410                          BEQ     10$                     ; if not, branch
036734                                  PRINTF  #TSHD3                  ; else, print test header
036734  012746  037344                                                          MOV     #TSHD3,-(SP)
036740  012746  000001                                                          MOV     #1,-(SP)
036744  010600                                                                  MOV     SP,R0
036746  104417                                                                  TRAP    C$PNTF
036750  062706  000004                                                          ADD     #4,SP
036754  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter

036760  052777  000040  143640  20$:    BIS     #RES,@DRXSCR            ; reset the DRX11-C
036766  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
036772  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)
                                ;
                                ; Check that all R/W bits can be set
                                ;
036776                                  BGNSEG
036776  104404                                                                  TRAP    C$BSEG
037000  012737  011417  003020          MOV     #ACS!MNT!COUT!SBE!ECE!MET!GO,GOOD ; expected value
037006  013777  003020  143620          MOV     GOOD,@DRXDBR            ; set up for ACS
037014  052777  000001  143604          BIS     #FNCTO,@DRXSCR          ; transfer to ACS
037022  017737  143606  003022          MOV     @DRXDBR,BAD             ; read from ACS
037030  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
037034  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
037040  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
037044  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
037052  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
037056  001404                          BEQ     30$                     ; branch if ACS is OK
037060                                  ERRSOFT 300,E300,EGB            ; print "BITS COULD NOT BE SET"
037060  104457                                                                  TRAP    C$ERSOFT
037062  000454                                                                  .WORD   300
037064  037377                                                                  .WORD   E300
037066  023416                                                                  .WORD   EGB
037070                          30$:    ENDSEG
037070                                                                  10000$:
037070  104405                                                                  TRAP    C$ESEG
                                ;
                                ; Check that all R/W bits can be cleared
                                ;
037072                                  BGNSEG
037072  104404                                                                  TRAP    C$BSEG
037074  012737  010000  003020          MOV     #ACS,GOOD               ; expected value
037102  013777  003020  143524          MOV     GOOD,@DRXDBR            ; set up for ACS
037110  052777  000001  143510          BIS     #FNCTO,@DRXSCR          ; transfer to ACS
037116  017737  143512  003022          MOV     @DRXDBR,BAD             ; read from ACS
037124  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
```

```
TEST 3 - ACS Read/write test.

 4909 037130  042700  171777                BIC     #↑C<UNI!CMP>,R0         ; and isolate uncertain bits
 4910 037134  050037  003020                BIS     R0,GOOD                 ; set them in GOOD if they are set
 4911 037140  023737  003022  003020        CMP     BAD,GOOD                ; contents correct ?
 4912 037146  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 4913 037152  001404                        BEQ     40$                     ; branch if ACS is OK
 4914 037154                                ERRSOFT 301,E301,EGB            ; print "BITS COULD NOT BE CLEARED"
      037154  104457                                                                TRAP    C$ERSOFT
      037156  000455                                                                .WORD   301
      037160  037444                                                                .WORD   E301
      037162  023416                                                                .WORD   EGB
 4915 037164                          40$:    ENDSEG
      037164                                                                  10001$:
      037164  104405                                                                TRAP    C$ESEG
 4916                                ;
 4917                                ; Check that each R/W bit can be set
 4918                                ;
 4919 037166  012701  001416                MOV     #MNT!COUT!SBE!ECE!MET,R1 ; save R/W bits in R1
 4920 037172  012702  000001                MOV     #1,R2                   ; first bit to test in R2
 4921 037176  030201                  50$:  BIT     R2,R1                   ; is it a R/W bit ?
 4922 037200  001003                        BNE     70$                     ; if yes, test it
 4923
 4924 037202  006302                  60$:  ASL     R2                      ; else find next R/W bit
 4925 037204  103442                        BCS     90$                     ; if all done, exit test
 4926 037206  000773                        BR      50$                     ; else check if next bit is R/W
 4927
 4928 037210                          70$:  BGNSEG
      037210  104404                                                                TRAP    C$BSEG
 4929 037212  012737  010000  003020        MOV     #ACS,GOOD               ; clear all R/W bits
 4930 037220  050237  003020                BIS     R2,GOOD                 ; except for one bit
 4931 037224  013777  003020  143402        MOV     GOOD,@DRXDBR            ; load into DBR
 4932 037232  052777  000001  143366        BIS     #FNCT0,@DRXSCR          ; transfer to ACS
 4933 037240  017737  143370  003022        MOV     @DRXDBR,BAD             ; read it back
 4934 037246  013700  003022                MOV     BAD,R0                  ; get the contents into R0
 4935 037252  042700  171777                BIC     #↑C<UNI!CMP>,R0         ; and isolate uncertain bits
 4936 037256  050037  003020                BIS     R0,GOOD                 ; set them in GOOD if they are set
 4937 037262  023737  003022  003020        CMP     BAD,GOOD                ; contents correct ?
 4938 037270  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 4939 037274  001404                        BEQ     80$                     ; branch if ACS is OK
 4940 037276                                ERRSOFT 302,E302,EGB            ; print "BITS COULD NOT BE"
      037276  104457                                                                TRAP    C$ERSOFT
      037300  000456                                                                .WORD   302
      037302  037515                                                                .WORD   E302
      037304  023416                                                                .WORD   EGB
 4941 037306                          80$:    ENDSEG                  ;       "INDIVIDUALLY SET"
      037306                                                                        10002$:
      037306  104405                                                                TRAP    C$ESEG
 4942
 4943 037310  000734                        BR      60$                     ; test next bit
 4944
 4945 037312  005737  002234          90$:  TST     QVMODE                  ; is quick verify mode selected ?
 4946 037316  001010                        BNE     100$                    ; if yes, exit test
 4947 037320  005237  003032                INC     ITRCNT                  ; else, increment iteration counter
 4948 037324  023727  003032  000010        CMP     ITRCNT,#10              ; iterations completed ?
 4949 037332  001402                        BEQ     100$                    ; if yes, branch
 4950 037334  000137  036760                JMP     20$                     ; else, do another iteration
 4951
 4952 037340                          100$: EXIT TST
```

TEST 3 - ACS Read/write test.

```
        037340  104432                                                                  TRAP    C$EXIT
        037342  000236                                                                  .WORD   L10034-.
   4953
   4954                                         .NLIST  BEX
   4955 037344     045     123     062  TSHD3:: .ASCIZ  \%S2%AACS READ/WRITE TEST%N\
   4956 037377     101     103     123  E300:   .ASCIZ  \ACS READ/WRITE BITS COULD NOT BE SET\
   4957 037444     101     103     123  E301:   .ASCIZ  \ACS READ/WRITE BITS COULD NOT BE CLEARED\
   4958 037515     101     103     123  E302:   .ASCIZ  \ACS READ/WRITE BITS COULD NOT BE INDIVIDUALLY SET\
   4959                                         .LIST   BEX
   4960                                         .EVEN
   4961
   4962 037600                                 ENDTST
        037600                                                          L10034:
        037600  104401                                                          TRAP    C$ETST
   4963
```

TEST 4 - CTA Read/write Test.

```
                                .SBTTL  TEST 4 - CTA Read/write Test.

                                ;++
                                ; This test checks that the read/write bits of the CTA register
                                ; can all be set, all cleared and individually set.
                                ;--

037602                                  BGNTST
037602                                                                  T4::
037602  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
037606                                  RFLAGS  R0                      ; read operator flags
037606  104421                                                                  TRAP    C$RFLA
037610  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
037614  001410                          BEQ     10$                     ; if not, branch
037616                                  PRINTF  #TSHD4                  ; else, print test header
037616  012746  040156                                                          MOV     #TSHD4,-(SP)
037622  012746  000001                                                          MOV     #1,-(SP)
037626  010600                                                                  MOV     SP,R0
037630  104417                                                                  TRAP    C$PNTF
037632  062706  000004                                                          ADD     #4,SP
037636  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter

037642  052777  000040  142756  20$:    BIS     #RES,@DRXSCR            ; reset the DRX11-C
037650  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
037654  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)
                                ;
                                ; Check that all R/W bits can be set
                                ;
037660                                  BGNSEG
037660  104404                                                                  TRAP    C$BSEG
037662  012737  021777  003020          MOV     #CTA!1777,GOOD          ; expected value
037670  013777  003020  142736          MOV     GOOD,@DRXDBR            ; set up for CTA
037676  052777  000001  142722          BIS     #FNCTO,@DRXSCR          ; transfer to CTA
037704  017737  142724  003022          MOV     @DRXDBR,BAD             ; read from CTA
037712  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
037720  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
037724  001404                          BEQ     30$                     ; branch if CTA is OK
037726                                  ERRSOFT 400,E400,EGB            ; print "BITS COULD NOT BE SET"
037726  104457                                                                  TRAP    C$ERSOFT
037730  000620                                                                  .WORD   400
037732  040211                                                                  .WORD   E400
037734  023416                                                                  .WORD   EGB
037736                          30$:    ENDSEG
037736                                                                10000$:
037736  104405                                                                  TRAP    C$ESEG
                                ;
                                ; Check that all R/W bits can be cleared
                                ;
037740                                  BGNSEG
037740  104404                                                                  TRAP    C$BSEG
037742  012737  020000  003020          MOV     #CTA,GOOD               ; expected value
037750  013777  003020  142656          MOV     GOOD,@DRXDBR            ; set up for CTA
037756  052777  000001  142642          BIS     #FNCTO,@DRXSCR          ; transfer to CTA
037764  017737  142644  003022          MOV     @DRXDBR,BAD             ; read from CTA
037772  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
040000  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
040004  001404                          BEQ     40$                     ; branch if CTA is OK
```

TEST 4 - CTA Read/write Test.

```
   5007 040006                                  ERRSOFT 401,E401,EGB            ; print "BITS COULD NOT BE CLEARED"
        040006  104457                                                                TRAP    C$ERSOFT
        040010  000621                                                                .WORD   401
        040012  040256                                                                .WORD   E401
        040014  023416                                                                .WORD   EGB
   5008 040016                          40$:    ENDSEG                                        10001$:
        040016                                                                        TRAP    C$ESEG
        040016  104405
   5009                                 ;
   5010                                 ; Check that each R/W bit can be set
   5011                                 ;
   5012 040020  012701  001777                  MOV     #1777,R1                ; save R/W bits in R1
   5013 040024  012702  000001                  MOV     #1,R2                   ; first bit to test in R2
   5014 040030  030201                  50$:    BIT     R2,R1                   ; is it a R/W bit ?
   5015 040032  001003                          BNE     70$                     ; if yes, test it
   5016
   5017 040034  006302                  60$:    ASL     R2                      ; else find next R/W bit
   5018 040036  103434                          BCS     90$                     ; if all done, exit test
   5019 040040  000773                          BR      50$                     ; else check if next bit is R/W
   5020
   5021 040042                          70$:    BGNSEG
        040042  104404                                                                TRAP    C$BSEG
   5022 040044  012737  020000  003020          MOV     #CTA,GOOD               ; clear all R/W bits
   5023 040052  050237  003020                  BIS     R2,GOOD                 ; except for one bit
   5024 040056  013777  003020  142550          MOV     GOOD,@DRXDBR            ; load into DBR
   5025 040064  052777  000001  142534          BIS     #FNCTO,@DRXSCR          ; transfer to CTA
   5026 040072  017737  142536  003022          MOV     @DRXDBR,BAD             ; read it back
   5027 040100  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
   5028 040106  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
   5029 040112  001404                          BEQ     80$                     ; branch if CTA is OK
   5030 040114                                  ERRSOFT 402,E402,EGB            ; print "BITS COULD NOT BE"
        040114  104457                                                                TRAP    C$ERSOFT
        040116  000622                                                                .WORD   402
        040120  040327                                                                .WORD   E402
        040122  023416                                                                .WORD   EGB
   5031 040124                          80$:    ENDSEG                          ;       "INDIVIDUALLY SET"
        040124                                                                                10002$:
        040124  104405                                                                TRAP    C$ESEG
   5032
   5033 040126  000742                          BR      60$                     ; test next bit
   5034
   5035 040130  005737  002234          90$:    TST     QVMODE                  ; is quick verify mode selected ?
   5036 040134  001006                          BNE     100$                    ; if yes, exit test
   5037 040136  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
   5038 040142  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
   5039 040150  001234                          BNE     20$                     ; if not, do another iteration
   5040
   5041 040152                          100$:   EXIT TST
        040152  104432                                                                TRAP    C$EXIT
        040154  000236                                                                 WORD   L10035-.
   5042
   5043                                         .NLIST  BEX
   5044 040156     045     123     062  TSHD4:: .ASCIZ   \%S2%ACTA READ/WRITE TEST%N\
   5045 040211     103     124     101  E400:   .ASCIZ   \CTA READ/WRITE BITS COULD NOT BE SET\
   5046 040256     103     124     101  E401:   .ASCIZ   \CTA READ/WRITE BITS COULD NOT BE CLEARED\
   5047 040327     103     124     101  E402:   .ASCIZ   \CTA READ/WRITE BITS COULD NOT BE INDIVIDUALLY SET\
   5048                                         .LIST   BEX
```

TEST 4 - CTA Read/write Test.

```
   5049                                         .EVEN
   5050
   5051 040412                                  ENDTST
        040412                                                  L10035:
        040412  104401                                                 TRAP    C$ETST
   5052
```

```
                                .SBTTL  TEST 5 - Control Table Data Test.

                                ;++
                                ; This test checks that bits 0 to 8 of each control word can all
                                ; be set, all cleared, and individually set.
                                ;--

040414                                  BGNTST
040414                                                                  T5::
040414  005037  003030                   CLR     STAFLG                  ; flag test has been run since start
040420                                  RFLAGS  R0                      ; read operator flags
040420  104421                                                                  TRAP    C$RFLA
040422  032700  001000                   BIT     #PNT,R0                 ; print test headers ?
040426  001410                           BEQ     10$                     ; if not, branch
040430                                  PRINTF  #TSHD5                  ; else, print test header
040430  012746  041016                                                          MOV     #TSHD5,-(SP)
040434  012746  000001                                                          MOV     #1,-(SP)
040440  010600                                                                  MOV     SP,R0
040442  104417                                                                  TRAP    C$PNTF
040444  062706  000004                                                          ADD     #4,SP

040450  005037  003032           10$:    CLR     ITRCNT                  ; clear iteration counter
040454  052777  000040  142144           BIS     #RES,@DRXSCR            ; reset the DRX11-C
040462  004737  026106                   JSR     PC,WT500                ; wait for 1 millisecond
040466  004737  026106                   JSR     PC,WT500                ; (2*500 microseconds)

040472  012702  020000                   MOV     #CTA,R2                 ; start with CWR 0 in R2

040476  010277  142132           20$:    MOV     R2,@DRXDBR              ; set up for transfer
040502  052777  000001  142116           BIS     #FNCTO,@DRXSCR          ; transfer to CTA
                                ;
                                ; Check that all R/W bits can be set
                                ;
040510                                  BGNSEG
040510  104404                                                                  TRAP    C$BSEG
040512  012737  030777  003020           MOV     #CWR!777,GOOD           ; expected value
040520  013777  003020  142106           MOV     GOOD,@DRXDBR            ; set up for CWR
040526  052777  000001  142072           BIS     #FNCTO,@DRXSCR          ; transfer to CWR
040534  017737  142074  003022           MOV     @DRXDBR,BAD             ; read from CWR
040542  023737  003022  003020           CMP     BAD,GOOD                ; contents correct ?
040550  004737  026306                   CALL    INSERT                  ; skip branch if error insert selected
040554  001404                           BEQ     30$                     ; branch if CWR is OK
040556                                  ERRSOFT 500,E500,ECGB           ; print "BITS COULD NOT BE SET"
040556  104457                                                                  TRAP    C$ERSOFT
040560  000764                                                                  .WORD   500
040562  041055                                                                  .WORD   E500
040564  023514                                                                  .WORD   ECGB
040566                          30$:    ENDSEG
040566                                                                10000$:
040566  104405                                                                  TRAP    C$ESEG
                                ;
                                ; Check that all R/W bits can be cleared
                                ;
040570                                  BGNSEG
040570  104404                                                                  TRAP    C$BSEG
040572  012737  030000  003020           MOV     #CWR,GOOD               ; expected value
040600  013777  003020  142026           MOV     GOOD,@DRXDBR            ; set up for CWR
```

TEST 5 - Control Table Data Test.

```
5096 040606  052777  000001  142012          BIS     #FNCTO,@DRXSCR          ; transfer to CWR
5097 040614  017737  142014  003022          MOV     @DRXDBR,BAD             ; read from CWR
5098 040622  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5099 040630  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5100 040634  001404                          BEQ     40$                     ; branch if CWR is OK
5101 040636                                  ERRSOFT 501,E501,ECGB           ; print "BITS COULD NOT BE CLEARED"
     040636  104457                                                                  TRAP    C$ERSOFT
     040640  000765                                                                  .WORD   501
     040642  041120                                                                  .WORD   E501
     040644  023514                                                                  .WORD   ECGB
5102 040646                          40$:    ENDSEG
     040646                                                                  10001$:
     040646  104405                                                                  TRAP    C$ESEG
5103                                 ;
5104                                 ; Check that each R/W bit can be set
5105                                 ;
5106 040650  012701  000777                  MOV     #777,R1                 ; save R/W bits in R1
5107 040654  012703  000001                  MOV     #1,R3                   ; first bit to test in R2
5108 040660  030301                  50$:    BIT     R3,R1                   ; is it a R/W bit ?
5109 040662  001003                          BNE     70$                     ; if yes, test it
5110
5111 040664  006303                  60$:    ASL     R3                      ; else find next R/W bit
5112 040666  103434                          BCS     90$                     ; if all done, exit test
5113 040670  000773                          BR      50$                     ; else check if next bit is R/W
5114
5115 040672                          70$:    BGNSEG
     040672  104404                                                                  TRAP    C$BSEG
5116 040674  012737  030000  003020          MOV     #CWR,GOOD               ; clear all R/W bits
5117 040702  050337  003020                  BIS     R3,GOOD                 ; except for one bit
5118 040706  013777  003020  141720          MOV     GOOD,@DRXDBR            ; load into DBR
5119 040714  052777  000001  141704          BIS     #FNCTO,@DRXSCR          ; transfer to CWR
5120 040722  017737  141706  003022          MOV     @DRXDBR,BAD             ; read it back
5121 040730  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5122 040736  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5123 040742  001404                          BEQ     80$                     ; branch if CWR is OK
5124 040744                                  ERRSOFT 502,E502,ECGB           ; print "BITS COULD NOT BE"
     040744  104457                                                                  TRAP    C$ERSOFT
     040746  000766                                                                  .WORD   502
     040750  041167                                                                  .WORD   E502
     040752  023514                                                                  .WORD   ECGB
5125 040754                          80$:    ENDSEG                          ;       "INDIVIDUALLY SET"
     040754                                                                  10002$:
     040754  104405                                                                  TRAP    C$ESEG
5126
5127 040756  000742                          BR      60$                     ; test next bit
5128
5129 040760  005202                  90$:    INC     R2                      ; increment the control word number
5130 040762  020227  022000                  CMP     R2,#CTA!2000            ; have we tested them all ?
5131 040766  103643                          BLO     20$                     ; if not, test the next one
5132
5133 040770  005737  002234                  TST     QVMODE                  ; is quick verify mode selected ?
5134 040774  001006                          BNE     100$                    ; if yes, exit test
5135 040776  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
5136 041002  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
5137 041010  001232                          BNE     20$                     ; if not, do another iteration
5138
5139 041012                          100$:   EXIT TST
```

TEST 5 - Control Table Data Test.

```
        041012  104432                                                          TRAP    C$EXIT
        041014  000234                                                          .WORD   L10036-.
   5140
   5141                                         .NLIST  BEX
   5142 041016     045     123     062  TSHD5:: .ASCIZ  /%S2%ACONTROL TABLE DATA TEST%N/
   5143 041055     103     117     116  E500:   .ASCIZ  /CONTROL WORD BITS COULD NOT BE SET/
   5144 041120     103     117     116  E501:   .ASCIZ  /CONTROL WORD BITS COULD NOT BE CLEARED/
   5145 041167     103     117     116  E502:   .ASCIZ  /CONTROL WORD BITS COULD NOT BE INDIVIDUALLY SET/
   5146                                         .LIST   BEX
   5147                                         .EVEN
   5148
   5149 041250                                 ENDTST
        041250                                                          L10036:
        041250  104401                                                         TRAP    C$ETST
   5150
```

```
                                .SBTTL  TEST 6 - Control Table Address Test.

                                ;++
                                ; This checks for interaction between different words of the
                                ; control table. All 1024 control words are loaded with a modulo
                                ; 511 count. The entire control table is then read and the
                                ; contents of each word compared with the data which was written
                                ; to it. To check bit 9 of the address logic the first half of the
                                ; control table is loaded with all zeros and the second half with
                                ; all ones. The table is again read to check that the data was
                                ; correctly written.
                                ;--
041252                                  BGNTST
041252                                                                  T6::
041252  005037  003030                 CLR     STAFLG                  ; flag test has been run since start
041256                                  RFLAGS  R0                      ; read operator flags
041256  104421                                                                  TRAP    C$RFLA
041260  032700  001000                 BIT     #PNT,R0                 ; print test headers ?
041264  001410                         BEQ     10$                     ; if not, branch
041266                                  PRINTF  #TSHD6                  ; else, print test header
041266  012746  041734                                                          MOV     #TSHD6,-(SP)
041272  012746  000001                                                          MOV     #1,-(SP)
041276  010600                                                                  MOV     SP,R0
041300  104417                                                                  TRAP    C$PNTF
041302  062706  000004                                                          ADD     #4,SP

041306  005037  003032         10$:    CLR     ITRCNT                  ; clear iteration counter
041312  052777  000040  141306         BIS     #RES,@DRXSCR            ; reset the DRX11-C
041320  004737  026106                 JSR     PC,WT500                ; wait for 1 millisecond
041324  004737  026106                 JSR     PC,WT500                ; (2*500 microseconds)
                                ;
                                ; Check address bits 0 to 8
                                ;
041330  012701  030000         20$:    MOV     #CWR,R1                 ; pattern for control word in R1
041334  012702  020000                 MOV     #CTA,R2                 ; control table address in R2

041340  010277  141270         30$:    MOV     R2,@DRXDBR              ; set up control table address
041344  052777  000001  141254         BIS     #FNCT0,@DRXSCR          ; transfer to CTA register
041352  010177  141256                 MOV     R1,@DRXDBR              ; set up pattern
041356  052777  000001  141242         BIS     #FNCT0,@DRXSCR          ; transfer to CWR
041364  005201                         INC     R1                      ; increment the pattern
041366  042701  001000                 BIC     #1000,R1                ; make modulo 511
041372  005202                         INC     R2                      ; increment the CTA
041374  020227  021777                 CMP     R2,#CTA!1777            ; all control words written ?
041400  101757                         BLOS    30$                     ; if not, write next one

041402  012737  030000  003020         MOV     #CWR,GOOD               ; expected pattern in GOOD
041410  012702  020000                 MOV     #CTA,R2                 ; control table address in R2
041414                          40$:    BGNSEG
041414  104404                                                                  TRAP    C$BSEG
041416  010277  141212                 MOV     R2,@DRXDBR              ; set up control table address
041422  052777  000001  141176         BIS     #FNCT0,@DRXSCR          ; transfer to CTA register
041430  012777  030000  141176         MOV     #CWR,@DRXDBR            ; set up to read the CWR
041436  017737  141172  003022         MOV     @DRXDBR,BAD             ; and read it
041444  023737  003022  003020         CMP     BAD,GOOD                ; contents correct ?
041452  004737  026306                 CALL    INSERT                  ; skip branch if error insert selected
041456  001404                         BEQ     50$                     ; branch if CWR is OK
```

TEST 6 - Control Table Address Test.

```
5201 041460                            ERRSOFT 600,E600,ECGB          ; print "CTA ERROR, BITS 0-8"
     041460  104457                                                                TRAP    C$ERSOFT
     041462  001130                                                                .WORD   600
     041464  041776                                                                .WORD   E600
     041466  023514                                                                .WORD   ECGB
5202 041470                      50$:  ENDSEG                                                10000$:
     041470                                                                        TRAP    C$ESEG
     041470  104405
5203 041472  005237  003020            INC     GOOD                   ; increment the pattern
5204 041476  042737  001000  003020    BIC     #1000,GOOD             ; make modulo 511
5205 041504  005202                    INC     R2                     ; increment the CTA
5206 041506  020227  021777            CMP     R2,#CTA!1777           ; all control words checked ?
5207 041512  101740                    BLOS    40$                    ; if not, check the next one
5208                             ;
5209                             ; Check address bit 9
5210                             ;
5211 041514  012701  030000            MOV     #CWR,R1                ; pattern 0 for control word in R1
5212 041520  012702  020000            MOV     #CTA,R2                ; control table address in R2
5213
5214 041524  010277  141104      60$:  MOV     R2,@DRXDBR             ; set up control table address
5215 041530  052777  000001  141070    BIS     #FNCT0,@DRXSCR         ; transfer to CTA register
5216 041536  010177  141072            MOV     R1,@DRXDBR             ; set up pattern
5217 041542  052777  000001  141056    BIS     #FNCT0,@DRXSCR         ; transfer to CWR
5218 041550  005202                    INC     R2                     ; increment the CTA
5219 041552  020227  021000            CMP     R2,#CTA!1000           ; half way through table ?
5220 041556  001002                    BNE     70$                    ; if not, branch
5221 041560  012701  030777            MOV     #CWR!777,R1            ; else set pattern to 777
5222 041564  020227  021777      70$:  CMP     R2,#CTA!1777           ; all control words written ?
5223 041570  101755                    BLOS    60$                    ; if not, write next one
5224
5225 041572  012737  030000  003020    MOV     #CWR,GOOD              ; expected pattern in GOOD
5226 041600  012702  020000            MOV     #CTA,R2                ; control table address in R2
5227
5228 041604                      80$:  BGNSEG
     041604  104404                                                                TRAP    C$BSEG
5229 041606  010277  141022            MOV     R2,@DRXDBR             ; set up control table address
5230 041612  052777  000001  141006    BIS     #FNCT0,@DRXSCR         ; transfer to CTA register
5231 041620  012777  030000  141006    MOV     #CWR,@DRXDBR           ; set up to read the CWR
5232 041626  017737  141002  003022    MOV     @DRXDBR,BAD            ; and read it
5233 041634  023737  003022  003020    CMP     BAD,GOOD               ; contents correct ?
5234 041642  004737  026306            CALL    INSERT                 ; skip branch if error insert selected
5235 041646  001404                    BEQ     90$                    ; branch if CWR is OK
5236 041650                            ERRSOFT 601,E601,ECGB          ; print "CTA ERROR, BIT 9"
     041650  104457                                                                TRAP    C$ERSOFT
     041652  001131                                                                .WORD   601
     041654  042047                                                                .WORD   E601
     041656  023514                                                                .WORD   ECGB
5237 041660                      90$:  ENDSEG                                                10001$:
     041660                                                                        TRAP    C$ESEG
     041660  104405
5238 041662  005202                    INC     R2                     ; increment the CTA
5239 041664  020227  021000            CMP     R2,#CTA!1000           ; half way through table ?
5240 041670  001003                    BNE     100$                   ; if not, branch
5241 041672  012737  030777  003020    MOV     #CWR!777,GOOD          ; else set expected pattern to 777
5242 041700  020227  021777     100$:  CMP     R2,#CTA!1777           ; all control words checked ?
5243 041704  101737                    BLOS    80$                    ; if not, check the next one
5244
```

TEST 6 - Control Table Address Test.

```
 5245 041706  005737  002234                    TST     QVMODE                  ; is quick verify mode selected ?
 5246 041712  001006                            BNE     110$                    ; if yes, exit test
 5247 041714  005237  003032                    INC     ITRCNT                  ; else, increment iteration counter
 5248 041720  023727  003032  000010            CMP     ITRCNT,#10              ; iterations completed ?
 5249 041726  001200                            BNE     20$                     ; if not, do another iteration
 5250
 5251 041730                         110$:      EXIT TST
      041730  104432                                                            TRAP    C$EXIT
      041732  000164                                                            .WORD   L10037-.
 5252
 5253                                           .NLIST  BEX
 5254 041734     045     123     062  TSHD6:: .ASCIZ  /%S2%ACONTROL TABLE ADDRESS TEST%N/
 5255 041776     103     117     116  E600:   .ASCIZ  /CONTROL TABLE ADDRESSING ERROR, BITS 0-8/
 5256 042047     103     117     116  E601:   .ASCIZ  /CONTROL TABLE ADDRESSING ERROR, BIT 9/
 5257                                           .LIST   BEX
 5258                                           .EVEN
 5259
 5260 042116                                    ENDTST
      042116                                                            L10037:
      042116  104401                                                            TRAP    C$ETST
 5261
```

TEST 7 - COUT and SEQ CONT L Interrupt Test.

```
                                .SBTTL  TEST 7 - COUT and SEQ CONT L Interrupt Test.

                                ;++
                                ; This tests that the command output (COUT) signal can be set and
                                ; that the SEQ CONT L input signal can be used to generate an
                                ; interrupt.
                                ;
                                ; The interrupt enable bit (6) of the DRX11-C SCR register is set
                                ; and in the AAF01-A's ACS register SBE is set. COUT is then set
                                ; and cleared. COUT is looped back to the SEQ CONT L input signal,
                                ; and the trailing edge sets SBR in the ACS register. This in turn
                                ; sets EOC and STAT0 in the DRX11-C SCR causing an interrupt.
                                ;--

042120                                  BGNTST
042120                                                                  T7::
042120  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
042124                                  RFLAGS  R0                      ; read operator flags
042124  104421                                                                 TRAP    C$RFLA
042126  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
042132  001410                          BEQ     10$                     ; if not, branch
042134                                  PRINTF  #TSHD7                  ; else, print test header
042134  012746  042534                                                         MOV     #TSHD7,-(SP)
042140  012746  000001                                                         MOV     #1,-(SP)
042144  010600                                                                 MOV     SP,R0
042146  104417                                                                 TRAP    C$PNTF
042150  062706  000004                                                         ADD     #4,SP

042154  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter
042160  052777  000040  140440  20$:    BIS     #RES,@DRXSCR            ; reset the DRX11-C
042166  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
042172  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)

042176                                  SETVEC  DRXVEC,#INT,#PRI07      ; set up DRX11-C vector
042176  012746  000340                                                         MOV     #PRI07,-(SP)
042202  012746  032216                                                         MOV     #INT,-(SP)
042206  013746  002636                                                         MOV     DRXVEC,-(SP)
042212  012746  000003                                                         MOV     #3,-(SP)
042216  104437                                                                 TRAP    C$SVEC
042220  062706  000010                                                         ADD     #10,SP

042224                          30$:    BGNSEG
042224  104404                                                                 TRAP    C$BSEG
042226  005037  002772                  CLR     INTFLG                  ; clear the interrupt flag
042232  005001                          CLR     R1                      ; clear a delay counter
042234                                  SETPRI  #PRI00                  ; drop the priority
042234  012700  000000                                                         MOV     #PRI00,R0
042240  104441                                                                 TRAP    C$SPRI
042242  012777  010010  140364          MOV     #ACS!SBE,@DRXDBR        ; set sequence break enable
042250  052777  000001  140350          BIS     #FNCT0,@DRXSCR          ; transfer to ACS register
042256  012777  000100  140342          MOV     #IE,@DRXSCR             ; allow interrupts from DRX11-C
042264  052777  000400  140342          BIS     #COUT,@DRXDBR           ; set COUT
042272  052777  000001  140326          BIS     #FNCT0,@DRXSCR          ; write to register
042300  042777  000400  140326          BIC     #COUT,@DRXDBR           ; and clear it again
042306  052777  000001  140312          BIS     #FNCT0,@DRXSCR          ; write to register

042314  005737  002772          40$:    TST     INTFLG                  ; have we had an interrupt ?
```

TEST 7 - COUT and SEQ CONT L Interrupt Test.

```
5304 042320  001002                          BNE     50$                     ; if yes, branch
5305 042322  005301                          DEC     R1                      ; have we waited long enough ?
5306 042324  001373                          BNE     40$                     ; if not, wait longer
5307
5308 042326                          50$:    SETPRI  #PRI07                  ; stop any more interrupts
     042326  012700  000340                                                          MOV     #PRI07,R0
     042332  104441                                                                  TRAP    C$SPRI
5309 042334  005737  002772                  TST     INTFLG                  ; did we get our interrupt ?
5310 042340  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5311 042344  001005                          BNE     60$                     ; branch if we got the interrupt
5312 042346                                  ERRSOFT 700,E700,ECHK           ; else print "INTERRUPT TIMEOUT"
     042346  104457                                                                  TRAP    C$ERSOFT
     042350  001274                                                                  .WORD   700
     042352  042606                                                                  .WORD   E700
     042354  023404                                                                  .WORD   ECHK
5313 042356                                  CKLOOP                          ; branch to BGNSEG if loop on error set
     042356  104406                                                                  TRAP    C$CLP1
5314
5315 042360  012737  010250  003020  60$:    MOV     #ACS!EOC!SBR!SBE,GOOD   ; get expected ACS contents
5316 042366  017737  140242  003022          MOV     @DRXDBR,BAD             ; and actual contents
5317 042374  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
5318 042400  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
5319 042404  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
5320 042410  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5321 042416  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5322 042422  001404                          BEQ     70$                     ; branch if ACS is OK
5323 042424                                  ERRSOFT 701,E701,EGB            ; else print "ACS INCORRECT"
     042424  104457                                                                  TRAP    C$ERSOFT
     042426  001275                                                                  .WORD   701
     042430  042654                                                                  .WORD   E701
     042432  023416                                                                  .WORD   EGB
5324
5325 042434  012737  100700  003020  70$:    MOV     #ATT!STAT0!RDY!IE,GOOD  ; get expected SCR contents
5326 042442  017737  140160  003022          MOV     @DRXSCR,BAD             ; and actual contents
5327 042450  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5328 042456  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5329 042462  001405                          BEQ     80$                     ; branch if SCR is OK
5330 042464                                  ERRSOFT 702,E702,EGB            ; else print "SCR INCORRECT"
     042464  104457                                                                  TRAP    C$ERSOFT
     042466  001276                                                                  .WORD   702
     042470  042727                                                                  .WORD   E702
     042472  023416                                                                  .WORD   EGB
5331 042474                                  CKLOOP                          ; branch to BGNSEG if loop on error set
     042474  104406                                                                  TRAP    C$CLP1
5332
5333 042476                          80$:    ENDSEG
     042476                                                                  10000$:
     042476  104405                                                                  TRAP    C$ESEG
5334
5335 042500  005737  002234                  TST     QVMODE                  ; is quick verify mode selected ?
5336 042504  001006                          BNE     90$                     ; if yes, exit test
5337 042506  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
5338 042512  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
5339 042520  001217                          BNE     20$                     ; if not, do another iteration
5340
5341 042522                          90$:    CLRVEC  DRXVEC                  ; restore the DRS trap catcher
     042522  013700  002636                                                          MOV     DRXVEC,R0
```

TEST 7 - COUT and SEQ CONT L Interrupt Test.

```
        042526  104436                                                          TRAP    C$CVEC
   5342 042530                          EXIT TST
        042530  104432                                                          TRAP    C$EXIT
        042532  000260                                                          .WORD   L10040-.
   5343
   5344                                 .NLIST  BEX
   5345 042534     045     123     062  TSHD7:: .ASCIZ  /%S2%ACOUT AND SEQ CONT L INTERRUPT TEST%N/
   5346 042606     111     116     124  E700:   .ASCIZ  /INTERRUPT TIMEOUT AFTER COUT LOOPBACK/
   5347 042654     101     103     123  E701:   .ASCIZ  /ACS REGISTER INCORRECT AFTER COUT LOOPBACK/
   5348 042727     104     122     130  E702:   .ASCIZ  /DRX11-C SCR REGISTER INCORRECT AFTER COUT LOOPBACK/
   5349                                 .LIST   BEX
   5350                                 .EVEN
   5351
   5352 043012                          ENDTST
        043012                                                          L10040:
        043012  104401                                                          TRAP    C$ETST
   5353
```

TEST 8 - DMA to Registers Test.

```
                                .SBTTL  TEST 8 - DMA to Registers Test.

                                ;++
                                ; This test performs a DMA write to the ACS and CTA registers. The
                                ; register contents are then read back under program control and
                                ; checked.
                                ;--

                                        BGNTST
                                                                                T8::
                                        CLR     STAFLG                  ; flag test has been run since start
                                        RFLAGS  R0                      ; read operator flags
                                                                                TRAP    C$RFLA
                                        BIT     #PNT,R0                 ; print test headers ?
                                        BEQ     10$                     ; if not, branch
                                        PRINTF  #TSHD8                  ; else, print test header
                                                                                MOV     #TSHD8,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP

                                10$:    CLR     ITRCNT                  ; clear iteration counter
                                        BIS     #RES,@DRXSCR            ; reset the DRX11-C
                                        JSR     PC,WT500                ; wait for 1 millisecond
                                        JSR     PC,WT500                ; (2*500 microseconds)

                                        MOV     #ACS!416,BUFOUT         ; set up 2 words for transfer
                                        MOV     #CTA!525,BUFOUT+2       ; to the ACS and CTA registers

                                        SETVEC  DRXVEC,#INT,#PRI07      ; set up DRX11-C vector
                                                                                MOV     #PRI07,-(SP)
                                                                                MOV     #INT,-(SP)
                                                                                MOV     DRXVEC,-(SP)
                                                                                MOV     #3,-(SP)
                                                                                TRAP    C$SVEC
                                                                                ADD     #10,SP

                                20$:    BGNSEG
                                                                                TRAP    C$BSEG
                                        SETPRI  #PRI00                  ; drop the priority
                                                                                MOV     #PRI00,R0
                                                                                TRAP    C$SPRI
                                        CLR     INTFLG                  ; clear the interrupt flag
                                        CLR     R1                      ; clear a delay counter
                                        MOV     #BAR1,@DRXCOR           ; address BAR1
                                        MOV     #BUFOUT,@DRXADR         ; put buffer address in BAR1
                                        MOV     #WCR1,@DRXCOR           ; set up WCR1
                                        MOV     #-2,@DRXADR             ; to transfer 2 words
                                        MOV     #RUN,@DRXCOR            ; start the DMA
                                        MOV     #IE,@DRXSCR             ; enable DRX11-C interrupts

                                30$:    TST     INTFLG                  ; have we had an interrupt ?
                                        BNE     40$                     ; if yes, branch
                                        DEC     R1                      ; have we waited long enough ?
                                        BNE     30$                     ; if not, wait longer
```

TEST 8 - DMA to Registers Test.

```
5396 043230                                40$:    SETPRI  #PRI07                  ; stop any more interrupts
     043230  012700  000340                                                                MOV     #PRI07,R0
     043234  104441                                                                        TRAP    C$SPRI
5397 043236  005737  002772                        TST     INTFLG                  ; did we get the interrupt ?
5398 043242  004737  026306                        CALL    INSERT                  ; skip branch if error insert selected
5399 043246  001005                                BNE     50$                     ; branch if we got the interrupt
5400 043250                                        ERRSOFT 800,E800,ECHK           ; else print "NO INTERRUPT AFTER DMA"
     043250  104457                                                                        TRAP    C$ERSOFT
     043252  001440                                                                        .WORD   800
     043254  043507                                                                        .WORD   E800
     043256  023404                                                                        .WORD   ECHK
5401 043260                                        CKLOOP                          ; branch to BGNSEG if loop on error set
     043260  104406                                                                        TRAP    C$CLP1
5402
5403 043262  013737  003040  003020  50$:    MOV     BUFOUT,GOOD                     ; get expected ACS contents
5404 043270  012777  010000  137336            MOV     #ACS,@DRXDBR                    ; address the ACS register
5405 043276  017737  137332  003022            MOV     @DRXDBR,BAD                     ; and get the actual contents
5406 043304  013700  003022                    MOV     BAD,R0                          ; get the contents into R0
5407 043310  042700  171777                    BIC     #^C<UNI!CMP>,R0                 ; and isolate uncertain bits
5408 043314  050037  003020                    BIS     R0,GOOD                         ; set them in GOOD if they are set
5409 043320  023737  003022  003020            CMP     BAD,GOOD                        ; contents correct ?
5410 043326  004737  026306                    CALL    INSERT                          ; skip branch if error insert selected
5411 043332  001405                            BEQ     60$                             ; branch if ACS is OK
5412 043334                                    ERRSOFT 801,E801,EGB                    ; else print "ACS INCORRECT"
     043334  104457                                                                        TRAP    C$ERSOFT
     043336  001441                                                                        .WORD   801
     043340  043553                                                                        .WORD   E801
     043342  023416                                                                        .WORD   EGB
5413 043344                                        CKLOOP                          ; branch to BGNSEG if loop on error set
     043344  104406                                                                        TRAP    C$CLP1
5414
5415 043346  013737  003042  003020  60$:    MOV     BUFOUT+2,GOOD                   ; get expected CTA contents
5416 043354  012777  020000  137252            MOV     #CTA,@DRXDBR                    ; address the CTA
5417 043362  017737  137246  003022            MOV     @DRXDBR,BAD                     ; and get the actual contents
5418 043370  023737  003022  003020            CMP     BAD,GOOD                        ; contents correct ?
5419 043376  004737  026306                    CALL    INSERT                          ; skip branch if error insert selected
5420 043402  001404                            BEQ     70$                             ; branch if CTA is OK
5421 043404                                    ERRSOFT 802,E802,EGB                    ; else print "CTA INCORRECT"
     043404  104457                                                                        TRAP    C$ERSOFT
     043406  001442                                                                        .WORD   802
     043410  043611                                                                        .WORD   E802
     043412  023416                                                                        .WORD   EGB
5422
5423 043414                                70$:    ENDSEG
     043414                                                                    10000$:
     043414  104405                                                                        TRAP    C$ESEG
5424
5425 043416  005737  002234                        TST     QVMODE                  ; is quick verify mode selected ?
5426 043422  001006                                BNE     80$                     ; if yes, exit test
5427 043424  005237  003032                        INC     ITRCNT                  ; else, increment iteration counter
5428 043430  023727  003032  000010                CMP     ITRCNT,#10              ; iterations completed ?
5429 043436  001236                                BNE     20$                     ; if not, do another iteration
5430
5431 043440                                80$:    CLRVEC  DRXVEC                  ; restore the DRS trap catcher
     043440  013700  002636                                                                MOV     DRXVEC,R0
     043444  104436                                                                        TRAP    C$CVEC
5432 043446                                        EXIT TST
```

TEST 8 - DMA to Registers Test.

```
        043446  104432                                                                  TRAP    C$EXIT
        043450  000200                                                                  .WORD   L10041-.
   5433
   5434                                                 .NLIST  BEX
   5435 043452      045     123     062  TSHD8::        .ASCIZ  /%S2%ADMA TO REGISTERS TEST%N/
   5436 043507      116     117     040  E800:          .ASCIZ  /NO INTERRUPT AFTER DMA TO REGISTERS/
   5437 043553      101     103     123  E801:          .ASCIZ  /ACS INCORRECT AFTER DMA WRITE/
   5438 043611      103     124     101  E802:          .ASCIZ  /CTA INCORRECT AFTER DMA WRITE/
   5439                                                 .LIST   BEX
   5440                                                 .EVEN
   5441
   5442 043650                                          ENDTST
        043650                                                              L10041:
        043650  104401                                                            TRAP    C$ETST
   5443
```

```
                                .SBTTL  TEST 9 - DMA to Control Table Test.

                                ;++
                                ; This test performs a single block DMA to all 1024 words of the
                                ; control table. The control table contents are then read back
                                ; under program control and checked.
                                ;--

043652                                  BGNTST
043652                                                                  T9::
043652  005037  003030                   CLR     STAFLG                 ; flag test has been run since start
043656                                  RFLAGS  R0                      ; read operator flags
043656  104421                                                                  TRAP    C$RFLA
043660  032700  001000                   BIT     #PNT,R0                ; print test headers ?
043664  001410                           BEQ     10$                    ; if not, branch
043666                                  PRINTF  #TSHD9                  ; else, print test header
043666  012746  044310                                                          MOV     #TSHD9,-(SP)
043672  012746  000001                                                          MOV     #1,-(SP)
043676  010600                                                                  MOV     SP,R0
043700  104417                                                                  TRAP    C$PNTF
043702  062706  000004                                                          ADD     #4,SP

043706  005037  003032          10$:     CLR     ITRCNT                 ; clear iteration counter
043712  052777  000040  136706           BIS     #RES,@DRXSCR           ; reset the DRX11-C
043720  004737  026106                   JSR     PC,WT500               ; wait for 1 millisecond
043724  004737  026106                   JSR     PC,WT500               ; (2*500 microseconds)

                                ;
                                ; Set up the DMA output buffer
                                ;
043730  012701  020000                   MOV     #CTA,R1                ; start with control word 0
043734  012702  030000                   MOV     #CWR,R2                ; initialize pattern for control word
043740  012703  003040                   MOV     #BUFOUT,R3             ; set up DMA buffer

043744  010123          20$:     MOV     R1,(R3)+               ; put a control word address in buffer
043746  010223                   MOV     R2,(R3)+               ; follow it with some data
043750  005202                   INC     R2                     ; increment the data pattern
043752  042702  001000           BIC     #1000,R2               ; prevent overflow to bit 9
043756  005201                   INC     R1                     ; go to next control word
043760  020127  021777           CMP     R1,#CTA!1777           ; all set up ?
043764  101767                   BLOS    20$                    ; if not, do some more

043766                                  SETVEC  DRXVEC,#INT,#PRI07      ; set up DRX11-C vector
043766  012746  000340                                                          MOV     #PRI07,-(SP)
043772  012746  032216                                                          MOV     #INT,-(SP)
043776  013746  002636                                                          MOV     DRXVEC,-(SP)
044002  012746  000003                                                          MOV     #3,-(SP)
044006  104437                                                                  TRAP    C$SVEC
044010  062706  000010                                                          ADD     #10,SP

044014          30$:     BGNSEG
044014  104404                                                                  TRAP    C$BSEG
044016                                  SETPRI  #PRI00                  ; drop the priority
044016  012700  000000                                                          MOV     #PRI00,R0
044022  104441                                                                  TRAP    C$SPRI
044024  005037  002772                   CLR     INTFLG                 ; clear the interrupt flag
044030  005001                           CLR     R1                     ; clear a delay counter
044032  012777  000000  136570           MOV     #BAR1,@DRXCOR          ; address BAR1
```

```
TEST 9 - DMA to Control Table Test.

 5486 044040  012777  003040  136564          MOV     #BUFOUT,@DRXADR         ; put buffer address in BAR1
 5487 044046  012777  000400  136554          MOV     #WCR1,@DRXCOR           ; set up WCR1
 5488 044054  012777  174000  136550          MOV     #-2048.,@DRXADR         ; to transfer 2048 words
 5489 044062  012777  020000  136540          MOV     #RUN,@DRXCOR            ; enable DRX11-C DMA's
 5490 044070  012777  000100  136530          MOV     #IE,@DRXSCR             ; enable interrupts
 5491
 5492 044076  005737  002772          40$:    TST     INTFLG                  ; have we had an interrupt ?
 5493 044102  001002                          BNE     50$                     ; if yes, branch
 5494 044104  005301                          DEC     R1                      ; have we waited long enough ?
 5495 044106  001373                          BNE     40$                     ; if not, wait longer
 5496                                 ;
 5497                                 ; Check for interrupt
 5498                                 ;
 5499 044110                          50$:    SETPRI  #PRI07                  ; stop any more interrupts
      044110  012700  000340                                                          MOV     #PRI07,R0
      044114  104441                                                                  TRAP    C$SPRI
 5500 044116  005737  002772                  TST     INTFLG                  ; did we get our interrupt ?
 5501 044122  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 5502 044126  001004                          BNE     60$                     ; branch if we got the interrupt
 5503 044130                                  ERRSOFT 900,E900,ECHK           ; else print "NO INTERRUPT AFTER DMA"
      044130  104457                                                                  TRAP    C$ERSOFT
      044132  001604                                                                  .WORD   900
      044134  044351                                                                  .WORD   E900
      044136  023404                                                                  .WORD   ECHK
 5504 044140                          60$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      044140  104406                                                                  TRAP    C$CLP1
 5505                                 ;
 5506                                 ; Check the control table
 5507                                 ;
 5508 044142  012737  030000  003020          MOV     #CWR,GOOD               ; expected pattern in GOOD
 5509 044150  012702  020000                  MOV     #CTA,R2                 ; control table address in R2
 5510 044154  010277  136454          70$:    MOV     R2,@DRXDBR              ; set up control table address
 5511 044160  052777  000001  136440          BIS     #FNCT0,@DRXSCR          ; transfer to CTA register
 5512 044166  012777  030000  136440          MOV     #CWR,@DRXDBR            ; set up to read the CWR
 5513 044174  017737  136434  003022          MOV     @DRXDBR,BAD             ; and read it
 5514 044202  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 5515 044210  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 5516 044214  001404                          BEQ     80$                     ; branch if CWR is OK
 5517 044216                                  ERRSOFT 901,E901,ECGB           ; print "CONTROL WORD INCORRECT"
      044216  104457                                                                  TRAP    C$ERSOFT
      044220  001605                                                                  .WORD   901
      044222  044421                                                                  .WORD   E901
      044224  023514                                                                  .WORD   ECGB
 5518 044226                          80$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      044226  104406                                                                  TRAP    C$CLP1
 5519
 5520 044230  005237  003020                  INC     GOOD                    ; increment the pattern
 5521 044234  042737  001000  003020          BIC     #1000,GOOD              ; make modulo 511
 5522 044242  005202                          INC     R2                      ; increment the CTA
 5523 044244  020227  021777                  CMP     R2,#CTA!1777            ; all control words checked ?
 5524 044250  101741                          BLOS    70$                     ; if not, check the next one
 5525 044252                                  ENDSEG
      044252                                                                          10000$:
      044252  104405                                                                  TRAP    C$ESEG
 5526
 5527 044254  005737  002234                  TST     QVMODE                  ; is quick verify mode selected ?
 5528 044260  001006                          BNE     90$                     ; if yes, exit test
```

TEST 9 - DMA to Control Table Test.

```
 5529 044262  005237  003032                 INC     ITRCNT                  ; else, increment iteration counter
 5530 044266  023727  003032  000010         CMP     ITRCNT,#10              ; iterations completed ?
 5531 044274  001247                         BNE     30$                     ; if not, do another iteration
 5532
 5533 044276                          90$:   CLRVEC  DRXVEC                  ; restore the DRS trap catcher
      044276  013700  002636                                                          MOV     DRXVEC,R0
      044302  104436                                                                  TRAP    C$CVEC
 5534 044304                                 EXIT TST
      044304  104432                                                                  TRAP    C$EXIT
      044306  000162                                                                  .WORD   L10042-.
 5535
 5536                                        .NLIST  BEX
 5537 044310     045     123     062  TSHD9:: .ASCIZ  /%S2%ADMA TO CONTROL TABLE TEST%N/
 5538 044351     116     117     040  E900:   .ASCIZ  /NO INTERRUPT AFTER DMA TO CONTROL TABLE/
 5539 044421     103     117     116  E901:   .ASCIZ  /CONTROL TABLE WORD INCORRECT AFTER DMA/
 5540                                        .LIST   BEX
 5541                                        .EVEN
 5542
 5543 044470                                 ENDTST
      044470                                                            L10042:
      044470  104401                                                                  TRAP    C$ETST
 5544
```

TEST 10 - CWR Mode 0 Test.

```
                                .SBTTL  TEST 10 - CWR Mode 0 Test.

                                ;++
                                ; This tests that after each conversion in mode 0, the CTA
                                ; register is incremented. The control table and values for output
                                ; are set up as follows :
                                ;
                                ; Control word    Mode    Channel    Output voltage
                                ;
                                ;       0          0         0           1.0
                                ;       1          0         1           2.0
                                ;       2          0         2           3.0
                                ;       3          0         3           0.0
                                ;
                                ; Conversions are then initiated with a DMA count of 3. After the
                                ; conversions are complete, the CTA register is checked to ensure
                                ; that it was correctly incremented.
                                ;--

                                        BGNTST
                                                                                T10::
                                        CLR     STAFLG                  ; flag test has been run since start
                                        RFLAGS  R0                      ; read operator flags
                                                                                TRAP    C$RFLA
                                        BIT     #PNT,R0                 ; print test headers ?
                                        BEQ     10$                     ; if not, branch
                                        PRINTF  #TSHD10                 ; else, print test header
                                                                                MOV     #TSHD10,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP

                                10$:    CLR     ITRCNT                  ; clear iteration counter
                                        BIS     #RES,@DRXSCR            ; reset the DRX11-C
                                        JSR     PC,WT500                ; wait for 1 millisecond
                                        JSR     PC,WT500                ; (2*500 microseconds)

                                        JSR     PC,SETTAB               ; set up control table and data
                                        4                               ; for 4 control words
                                        T10TAB                          ; using table at T10TAB

                                        MOV     CTIME,R1                ; get the selected conversion time
                                        BIS     #PCR,R1                 ; address the PCR
                                        MOV     R1,@DRXDBR              ; and enter the conversion time
                                        BIS     #FNCT0,@DRXSCR          ; transfer to PCR

                                20$:    BGNSEG
                                                                                TRAP    C$BSEG
                                        MOV     #CTA,@DRXDBR            ; reset the CTA register
                                        BIS     #FNCT0,@DRXSCR          ; back to control word 0
                                        MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
                                        MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
                                        MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
                                        MOV     #BUFOUT,@DRXADR         ; set up DMA address
                                        MOV     #WCR1,@DRXCOR           ; now select WCR1
                                        MOV     #-3,@DRXADR             ; to DMA 3 words
```

TEST 10 - CWR Mode 0 Test.

```
 5595 044664  012777  020000  135736          MOV     #RUN,@DRXCGR            ; start the DMA
 5596
 5597 044672  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 5598 044676  005002                          CLR     R2                      ; high byte is zero
 5599 044700  004737  027776                  JSR     PC,MUL                  ; multiply by
 5600 044704  000004                          4.                              ; the number of conversions
 5601 044706  004737  030072                  JSR     PC,DIV                  ; divide by
 5602 044712  000175                          125.                            ; 125
 5603
 5604 044714  005701                  30$:    TST     R1                      ; loop until R1/R2 is zero
 5605 044716  001404                          BEQ     50$                     ;
 5606 044720  005301                  40$:    DEC     R1                      ;
 5607 044722  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 5608 044726  000772                          BR      30$                     ;
 5609 044730  005702                  50$:    TST     R2                      ;
 5610 044732  001402                          BEQ     60$                     ; total timeout = number of conversions
 5611 044734  005302                          DEC     R2                      ;                 * conversion time * 2
 5612 044736  000770                          BR      40$                     ;
 5613
 5614 044740  012737  100600  003020  60$:    MOV     #ATT!STAT0!RDY,GOOD     ; get expected SCR contents
 5615 044746  017737  135654  003022          MOV     @DRXSCR,BAD             ; and actual contents
 5616 044754  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
 5617 044762  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 5618 044766  001404                          BEQ     70$                     ; branch if SCR is OK
 5619 044770                                  ERRSOFT 1000,E1000,EGB          ; else print "SCR INCORRECT"
      044770  104457                                                          TRAP    C$ERSOFT
      044772  001750                                                          .WORD   1000
      044774  045243                                                          .WORD   E1000
      044776  023416                                                          .WORD   EGB
 5620 045000                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      045000  104406                                                          TRAP    C$CLP1
 5621
 5622 045002  012737  010000  003020          MOV     #ACS,GOOD               ; get expected ACS contents
 5623 045010  012777  010000  135616          MOV     #ACS,@DRXDBR            ; address the ACS register
 5624 045016  017737  135612  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 5625 045024  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
 5626 045030  042700  171777                  BIC     #†C<UNI!CMP>,R0         ; and isolate uncertain bits
 5627 045034  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
 5628 045040  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 5629 045046  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 5630 045052  001405                          BEQ     80$                     ; branch if ACS is OK
 5631 045054                                  ERRSOFT 1001,E1001,EGB          ; else print "ACS INCORRECT"
      045054  104457                                                          TRAP    C$ERSOFT
      045056  001751                                                          .WORD   1001
      045060  045322                                                          .WORD   E1001
      045062  023416                                                          .WORD   EGB
 5632 045064                                  CKLOOP                          ; branch to BGNSEG if loop on error set
      045064  104406                                                          TRAP    C$CLP1
 5633
 5634 045066  012737  020003  003020  80$:    MOV     #CTA!3,GOOD             ; get expected CTA contents
 5635 045074  012777  020000  135532          MOV     #CTA,@DRXDBR            ; address the CTA
 5636 045102  017737  135526  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 5637 045110  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 5638 045116  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 5639 045122  001404                          BEQ     90$                     ; branch if CTA is OK
 5640 045124                                  ERRSOFT 1002,E1002,EGB          ; else print "CTA INCORRECT"
      045124  104457                                                          TRAP    C$ERSOFT
```

TEST 10 - CWR Mode 0 Test.

```
        045126  001752                                                          .WORD   1002
        045130  045371                                                          .WORD   E1002
        045132  023416                                                          .WORD   EGB
  5641
  5642 045134                           90$:    ENDSEG
       045134                                                           10000$:
       045134  104405                                                          TRAP    C$ESEG
  5643
  5644 045136  005737  002234                   TST     QVMODE                 ; is quick verify mode selected ?
  5645 045142  001006                           BNE     100$                   ; if yes, exit test
  5646 045144  005237  003032                   INC     ITRCNT                 ; else, increment iteration counter
  5647 045150  023727  003032  000010           CMP     ITRCNT,#10             ; iterations completed ?
  5648 045156  001211                           BNE     20$                    ; if not, do another iteration
  5649
  5650 045160                           100$:   EXIT TST
       045160  104432                                                          TRAP    C$EXIT
       045162  000256                                                          .WORD   L10043-.
  5651
  5652
  5653                                  ; Outputs :   Mode   Channel  Voltage
  5654                                  ;                              (mV)
  5655
  5656 045164  000000  000000  001750   T10TAB: .WORD   0,      0,      1000.  ; table for outputs
  5657 045172  000000  000001  003720           .WORD   0,      1,      2000.
  5658 045200  000000  000002  005670           .WORD   0,      2,      3000.
  5659 045206  000000  000003  007640           .WORD   0,      3,      4000.
  5660
  5661                                          .NLIST  BEX
  5662 045214      045     123     062   TSHD10::.ASCIZ  /%S2%ACWR MODE 0 TEST%N/
  5663 045243      104     122     130   E1000:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER MODE 0 CONVERSIONS/
  5664 045322      101     103     123   E1001:  .ASCIZ  /ACS INCORRECT AFTER MODE 0 CONVERSIONS/
  5665 045371      103     124     101   E1002:  .ASCIZ  /CTA INCORRECT AFTER MODE 0 CONVERSIONS/
  5666                                          .LIST   BEX
  5667                                          .EVEN
  5668
  5669 045440                                   ENDTST
       045440                                                           L10043:
       045440  104401                                                          TRAP    C$ETST
  5670
```

TEST 11 - CWR Mode 1 Test.

```
                                .SBTTL  TEST 11 - CWR Mode 1 Test.

                                ;++
                                ; This tests that after a conversion in mode 1, the CTA register
                                ; is reset to 0. The control table and values for output are set
                                ; up as follows :
                                ;
                                ; Control word    Mode    Channel    Output voltage
                                ;
                                ;       0          0         0           1.0
                                ;       1          0         1           2.0
                                ;       2          0         2           3.0
                                ;       3          0         3           4.0
                                ;       4          0         4           5.0
                                ;       5          1         5           6.0
                                ;
                                ; Conversions are then initiated with a DMA count of 6. After the
                                ; conversions are complete the CTA register is checked to ensure
                                ; that it was correctly reset.
                                ;--

045442                                  BGNTST
045442                                                                  T11::
045442  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
045446                                  RFLAGS  R0                      ; read operator flags
045446  104421                                                                  TRAP    C$RFLA
045450  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
045454  001410                          BEQ     10$                     ; if not, branch
045456                                  PRINTF  #TSHD11                 ; else, print test header
045456  012746  046200                                                          MOV     #TSHD11,-(SP)
045462  012746  000001                                                          MOV     #1,-(SP)
045466  010600                                                                  MOV     SP,R0
045470  104417                                                                  TRAP    C$PNTF
045472  062706  000004                                                          ADD     #4,SP

045476  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter
045502  052777  000040  135116          BIS     #RES,@DRXSCR            ; reset the DRX11-C
045510  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
045514  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)

045520  004737  030320                  JSR     PC,SETTAB               ; set up control table and data
045524  000006                          6                               ; for 6 control words
045526  046134                          T11TAB                          ; using table at T11TAB

045530  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
045534  052701  040000                  BIS     #PCR,R1                 ; address the PCR
045540  010177  135070                  MOV     R1,@DRXDBR              ; and enter the conversion time
045544  052777  000001  135054          BIS     #FNCT0,@DRXSCR          ; transfer to PCR

045552                          20$:    BGNSEG
045552  104404                                                                  TRAP    C$BSEG
045554  012777  020000  135052          MOV     #CTA,@DRXDBR            ; reset the CTA register
045562  052777  000001  135036          BIS     #FNCT0,@DRXSCR          ; back to control word 0
045570  012777  010001  135036          MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
045576  012777  000001  135022          MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
045604  012777  000000  135016          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
045612  012777  003040  135012          MOV     #BUFOUT,@DRXADR         ; set up DMA address
```

TEST 11 - CWR Mode 1 Test.

```
5721 045620  012777  000400  135002          MOV     #WCR1,@DRXCOR           ; now select WCR1
5722 045626  012777  177772  134776          MOV     #-6,@DRXADR             ; to DMA 6 words
5723 045634  012777  020000  134766          MOV     #RUN,@DRXCOR            ; start the DMA
5724
5725 045642  013701  002230                  MOV     CTIME,R1                ; get the conversion time
5726 045646  005002                          CLR     R2                      ; high byte is zero
5727 045650  004737  027776                  JSR     PC,MUL                  ; multiply by
5728 045654  000006                          6                               ; the number of conversions
5729 045656  004737  030072                  JSR     PC,DIV                  ; divide by
5730 045662  000175                          125.                            ; 125
5731
5732 045664  005701                  30$:    TST     R1                      ; loop until R1/R2 is zero
5733 045666  001404                          BEQ     50$                     ;
5734 045670  005301                  40$:    DEC     R1                      ;
5735 045672  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
5736 045676  000772                          BR      30$                     ;
5737 045700  005702                  50$:    TST     R2                      ;
5738 045702  001402                          BEQ     60$                     ; total timeout = number of conversions
5739 045704  005302                          DEC     R2                      ;                 * conversion time * 2
5740 045706  000770                          BR      40$                     ;
5741
5742 045710  012737  100600  003020  60$:    MOV     #ATT!STATO!RDY,GOOD     ; get expected SCR contents
5743 045716  017737  134704  003022          MOV     @DRXSCR,BAD             ; and actual contents
5744 045724  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
5745 045732  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5746 045736  001404                          BEQ     70$                     ; branch if SCR is OK
5747 045740                                  ERRSOFT 1100,E1100,EGB          ; else print "SCR INCORRECT"
     045740  104457                                                                  TRAP    C$ERSOFT
     045742  002114                                                                  .WORD   1100
     045744  046227                                                                  .WORD   E1100
     045746  023416                                                                  .WORD   EGB
5748 045750                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
     045750  104406                                                                  TRAP    C$CLP1
5749
5750 045752  012737  010000  003020          MOV     #ACS,GOOD               ; get expected ACS contents
5751 045760  012777  010000  134646          MOV     #ACS,@DRXDBR            ; address the ACS register
5752 045766  017737  134642  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
5753 045774  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
5754 046000  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
5755 046004  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
5756 046010  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5757 046016  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5758 046022  001405                          BEQ     80$                     ; branch if ACS is OK
5759 046024                                  ERRSOFT 1101,E1101,EGB          ; else print "ACS INCORRECT"
     046024  104457                                                                  TRAP    C$ERSOFT
     046026  002115                                                                  .WORD   1101
     046030  046305                                                                  .WORD   E1101
     046032  023416                                                                  .WORD   EGB
5760 046034                                  CKLOOP                          ; branch to BGNSEG if loop on error set
     046034  104406                                                                  TRAP    C$CLP1
5761
5762 046036  012737  020000  003020  80$:    MOV     #CTA,GOOD               ; get expected CTA contents
5763 046044  012777  020000  134562          MOV     #CTA,@DRXDBR            ; address the CTA
5764 046052  017737  134556  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
5765 046060  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5766 046066  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5767 046072  001404                          BEQ     90$                     ; branch if CTA is OK
```

TEST 11 - CWR Mode 1 Test.

```
5768 046074                                ERRSOFT 1102,E1102,EGB          ; else print "CTA INCORRECT"
     046074  104457                                                              TRAP    C$ERSOFT
     046076  002116                                                              .WORD   1102
     046100  046353                                                              .WORD   E1102
     046102  023416                                                              .WORD   EGB
5769
5770 046104                          90$:  ENDSEG
     046104                                                                 10000$:
     046104  104405                                                              TRAP    C$ESEG
5771
5772 046106  005737  002234                 TST     QVMODE                  ; is quick verify mode selected ?
5773 046112  001006                         BNE     100$                    ; if yes, exit test
5774 046114  005237  003032                 INC     ITRCNT                  ; else, increment iteration counter
5775 046120  023727  003032  000010         CMP     ITRCNT,#10              ; iterations completed ?
5776 046126  001211                         BNE     20$                     ; if not, do another iteration
5777
5778 046130                          100$: EXIT TST
     046130  104432                                                              TRAP    C$EXIT
     046132  000270                                                              .WORD   L10044-.
5779
5780
5781                                 ; Outputs :   Mode   Channel  Voltage
5782                                 ;                              (mV)
5783
5784 046134  000000  000000  001750 T11TAB: .WORD   0,      0,      1000.   ; table for outputs
5785 046142  000000  000001  003720         .WORD   0,      1,      2000.
5786 046150  000000  000002  005670         .WORD   0,      2,      3000.
5787 046156  000000  000003  007640         .WORD   0,      3,      4000.
5788 046164  000000  000004  011610         .WORD   0,      4,      5000.
5789 046172  000001  000005  013560         .WORD   1,      5,      6000.
5790
5791                                       .NLIST  BEX
5792 046200     045     123     062  TSHD11::.ASCIZ  /%S2%ACWR MODE 1 TEST%N/
5793 046227     104     122     130  E1100:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER MODE 1 CONVERSION/
5794 046305     101     103     123  E1101:  .ASCIZ  /ACS INCORRECT AFTER MODE 1 CONVERSION/
5795 046353     103     124     101  E1102:  .ASCIZ  /CTA INCORRECT AFTER MODE 1 CONVERSION/
5796                                       .LIST   BEX
5797                                       .EVEN
5798
5799 046422                          ENDTST
     046422                                                                 L10044:
     046422  104401                                                              TRAP    C$ETST
5800
```

TEST 12 - CWR Mode 2 Test.

```
                                .SBTTL  TEST 12 - CWR Mode 2 Test.

                                ;++
                                ; This tests that dummy control table loads can be made by using
                                ; mode 2. The control table and values for output are set up as
                                ; follows :
                                ;
                                ; Control word    Mode    Channel    Output voltage
                                ;
                                ;       0          0         0          1.0
                                ;       1          0         1          2.0
                                ;       2          2         1          2.0
                                ;       3          2         1          2.0
                                ;       4          2         1          2.0
                                ;       5          1         2          3.0
                                ;
                                ; The SCR interrupt enable bit is set and the DMA is started with
                                ; the DRX11-C WCR loaded for 3 DMA's. Conversions are then
                                ; initiated and the program waits for the conversions to complete.
                                ;--

046424                                  BGNTST
046424                                                                  T12::
046424  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
046430                                  RFLAGS  R0                      ; read operator flags
046430  104421                                                                 TRAP    C$RFLA
046432  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
046436  001410                          BEQ     10$                     ; if not, branch
046440                                  PRINTF  #TSHD12                 ; else, print test header
046440  012746  047162                                                         MOV     #TSHD12,-(SP)
046444  012746  000001                                                         MOV     #1,-(SP)
046450  010600                                                                 MOV     SP,R0
046452  104417                                                                 TRAP    C$PNTF
046454  062706  000004                                                         ADD     #4,SP

046460  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter
046464  052777  000040  134134          BIS     #RES,@DRXSCR            ; reset the DRX11-C
046472  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
046476  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)

046502  004737  030320                  JSR     PC,SETTAB               ; set up control table and data
046506  000006                          6                               ; for 6 control words
046510  047116                          T12TAB                          ; using table at T12TAB

046512  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
046516  052701  040000                  BIS     #PCR,R1                 ; address the PCR
046522  010177  134106                  MOV     R1,@DRXDBR              ; and enter the conversion time
046526  052777  000001  134072          BIS     #FNCT0,@DRXSCR          ; transfer to PCR

046534                          20$:    BGNSEG
046534  104404                                                                 TRAP    C$BSEG
046536  012777  020000  134070          MOV     #CTA,@DRXDBR            ; reset the CTA register
046544  052777  000001  134054          BIS     #FNCT0,@DRXSCR          ; back to control word 0
046552  012777  010001  134054          MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
046560  012777  000001  134040          MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
046566  012777  000000  134034          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
046574  012777  003040  134030          MOV     #BUFOUT,@DRXADR         ; set up DMA address
```

TEST 12 - CWR Mode 2 Test.

```
5851 046602  012777  000400  134020          MOV     #WCR1,@DRXCOR           ; now select WCR1
5852 046610  012777  177775  134014          MOV     #-3,@DRXADR             ; to DMA 3 words
5853 046616  012777  020000  134004          MOV     #RUN,@DRXCOR            ; start the DMA
5854
5855 046624  013701  002230                  MOV     CTIME,R1                ; get the conversion time
5856 046630  005002                          CLR     R2                      ; high byte is zero
5857 046632  004737  027776                  JSR     PC,MUL                  ; multiply by
5858 046636  000006                          6                               ; the number of conversions
5859 046640  004737  030072                  JSR     PC,DIV                  ; divide by
5860 046644  000175                          125.                            ; 125
5861
5862 046646  005701                  30$:    TST     R1                      ; loop until R1/R2 is zero
5863 046650  001404                          BEQ     50$                     ;
5864 046652  005301                  40$:    DEC     R1                      ;
5865 046654  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
5866 046660  000772                          BR      30$                     ;
5867 046662  005702                  50$:    TST     R2                      ;
5868 046664  001402                          BEQ     60$                     ; total timeout = number of conversions
5869 046666  005302                          DEC     R2                      ;                 * conversion time * 2
5870 046670  000770                          BR      40$                     ;
5871
5872 046672  012737  100600  003020  60$:    MOV     #ATT!STATO!RDY,GOOD     ; get expected SCR contents
5873 046700  017737  133722  003022          MOV     @DRXSCR,BAD             ; and actual contents
5874 046706  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
5875 046714  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5876 046720  001404                          BEQ     70$                     ; branch if SCR is OK
5877 046722                                  ERRSOFT 1200,E1200,EGB          ; else print "SCR INCORRECT"
     046722  104457                                                                  TRAP    C$ERSOFT
     046724  002260                                                                  .WORD   1200
     046726  047211                                                                  .WORD   E1200
     046730  023416                                                                  .WORD   EGB
5878 046732                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
     046732  104406                                                                  TRAP    C$CLP1
5879
5880 046734  012737  010000  003020          MOV     #ACS,GOOD               ; get expected ACS contents
5881 046742  012777  010000  133664          MOV     #ACS,@DRXDBR            ; address the ACS register
5882 046750  017737  133660  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
5883 046756  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
5884 046762  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
5885 046766  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
5886 046772  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5887 047000  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5888 047004  001405                          BEQ     80$                     ; branch if ACS is OK
5889 047006                                  ERRSOFT 1201,E1201,EGB          ; else print "ACS INCORRECT"
     047006  104457                                                                  TRAP    C$ERSOFT
     047010  002261                                                                  .WORD   1201
     047012  047270                                                                  .WORD   E1201
     047014  023416                                                                  .WORD   EGB
5890 047016                                  CKLOOP                          ; branch to BGNSEG if loop on error set
     047016  104406                                                                  TRAP    C$CLP1
5891
5892 047020  012737  020000  003020  80$:    MOV     #CTA,GOOD               ; get expected CTA contents
5893 047026  012777  020000  133600          MOV     #CTA,@DRXDBR            ; address the CTA
5894 047034  017737  133574  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
5895 047042  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
5896 047050  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
5897 047054  001404                          BEQ     90$                     ; branch if CTA is OK
```

TEST 12 - CWR Mode 2 Test.

```
5898 047056                            ERRSOFT 1202,E1202,EGB          ; else print "CTA INCORRECT"
     047056  104457                                                                  TRAP    C$ERSOFT
     047060  002262                                                                  .WORD   1202
     047062  047336                                                                  .WORD   E1202
     047064  023416                                                                  .WORD   EGB
5899
5900 047066                       90$:  ENDSEG
     047066                                                                  10000$:
     047066  104405                                                                  TRAP    C$ESEG
5901
5902 047070  005737  002234             TST     QVMODE                  ; is quick verify mode selected ?
5903 047074  001006                     BNE     100$                    ; if yes, exit test
5904 047076  005237  003032             INC     ITRCNT                  ; else, increment iteration counter
5905 047102  023727  003032  000010     CMP     ITRCNT,#10              ; iterations completed ?
5906 047110  001211                     BNE     20$                     ; if not, do another iteration
5907
5908 047112                       100$: EXIT TST
     047112  104432                                                                  TRAP    C$EXIT
     047114  000270                                                                  .WORD   L10045-.
5909
5910
5911                                   ; Outputs :   Mode   Channel  Voltage
5912                                   ;                              (mV)
5913
5914 047116  000000  000000  001750   T12TAB: .WORD   0,      0,      1000.   ; table for outputs
5915 047124  000000  000001  003720           .WORD   0,      1,      2000.
5916 047132  000002  000001  005670           .WORD   2,      1,      3000.
5917 047140  000002  000001  000000           .WORD   2,      1,      0       ; \
5918 047146  000002  000001  000000           .WORD   2,      1,      0       ;  > not used
5919 047154  000001  000002  000000           .WORD   1,      2,      0       ; /
5920
5921                                          .NLIST  BEX
5922 047162     045     123     062   TSHD12::.ASCIZ  /%S2%ACWR MODE 2 TEST%N/
5923 047211     104     122     130   E1200:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER MODE 2 CONVERSIONS/
5924 047270     101     103     123   E1201:  .ASCIZ  /ACS INCORRECT AFTER MODE 2 DMA OUTPUT/
5925 047336     103     124     101   E1202:  .ASCIZ  /CTA INCORRECT AFTER MODE 2 DMA OUTPUT/
5926                                          .LIST   BEX
5927                                          .EVEN
5928
5929 047404                            ENDTST
     047404                                                                  L10045:
     047404  104401                                                                  TRAP    C$ETST
5930
```

TEST 13 - CWR Mode 3, External Clock and SEQ CONT Test.

```
                                .SBTTL  TEST 13 - CWR Mode 3, External Clock and SEQ CONT Test.

                                ;++
                                ; This tests that the SEQ CONT signal can initiate and reinitiate
                                ; conversions in mode 3. The control table and values for output
                                ; are set up as follows :
                                ;
                                ; Control word    Mode    Channel    Output voltage
                                ;
                                ;       0          3         0           0.0
                                ;       1          0         0           1.0
                                ;       2          3         0           1.0
                                ;       3          1         1           2.0
                                ;
                                ; The SCR interrupt enable bit is set and the DMA is started with
                                ; the DRX11-C WCR loaded for 2 DMA's. The RATE CLK OUT signal is
                                ; connected to the SEQ CONT input via the test connector and so
                                ; triggers the conversions. The CTA register is checked to ensure
                                ; that this occurred.
                                ;--

047406                                  BGNTST
047406                                                                  T13::
047406  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
047412                                  RFLAGS  R0                      ; read operator flags
047412  104421                                                          TRAP    C$RFLA
047414  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
047420  001410                          BEQ     10$                     ; if not, branch
047422                                  PRINTF  #TSHD13                 ; else, print test header
047422  012746  050130                                                  MOV     #TSHD13,-(SP)
047426  012746  000001                                                  MOV     #1,-(SP)
047432  010600                                                          MOV     SP,R0
047434  104417                                                          TRAP    C$PNTF
047436  062706  000004                                                  ADD     #4,SP

047442  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter
047446  052777  000040  133152          BIS     #RES,@DRXSCR            ; reset the DRX11-C
047454  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
047460  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)

047464  004737  030320                  JSR     PC,SETTAB               ; set up control table and data
047470  000004                          4                               ; for 4 control words
047472  050100                          T13TAB                          ; using table at T13TAB

047474  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
047500  052701  040000                  BIS     #PCR,R1                 ; address the PCR
047504  010177  133124                  MOV     R1,@DRXDBR              ; and enter the conversion time
047510  052777  000001  133110          BIS     #FNCT0,@DRXSCR          ; transfer to PCR

047516                          20$:    BGNSEG
047516  104404                                                          TRAP    C$BSEG
047520  012777  020000  133106          MOV     #CTA,@DRXDBR            ; reset the CTA register
047526  052777  000001  133072          BIS     #FNCT0,@DRXSCR          ; back to control word 0
047534  012777  010001  133072          MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
047542  012777  000001  133056          MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
047550  012777  000000  133052          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
047556  012777  003040  133046          MOV     #BUFOUT,@DRXADR         ; set up DMA address
```

TEST 13 - CWR Mode 3, External Clock and SEQ CONT Test.

```
 5981 047564  012777  000400  133036          MOV     #WCR1,@DRXCOR           ; now select WCR1
 5982 047572  012777  177776  133032          MOV     #-2,@DRXADR             ; to DMA 2 words
 5983 047600  012777  020000  133022          MOV     #RUN,@DRXCOR            ; start the DMA
 5984
 5985 047606  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 5986 047612  005002                          CLR     R2                      ; high byte is zero
 5987 047614  004737  027776                  JSR     PC,MUL                  ; multiply by
 5988 047620  000004                          4                               ; the number of conversions
 5989 047622  004737  030072                  JSR     PC,DIV                  ; divide by
 5990 047626  000175                          125.                            ; 125
 5991
 5992 047630  005701                  30$:    TST     R1                      ; loop until R1/R2 is zero
 5993 047632  001404                          BEQ     50$                     ;
 5994 047634  005301                  40$:    DEC     R1                      ;
 5995 047636  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 5996 047642  000772                          BR      30$                     ;
 5997 047644  005702                  50$:    TST     R2                      ;
 5998 047646  001402                          BEQ     60$                     ; total timeout = number of conversions
 5999 047650  005302                          DEC     R2                      ;                 * conversion time * 2
 6000 047652  000770                          BR      40$                     ;
 6001
 6002 047654  012737  100600  003020  60$:    MOV     #ATT!STAT0!RDY,GOOD     ; get expected SCR contents
 6003 047662  017737  132740  003022          MOV     @DRXSCR,BAD             ; and actual contents
 6004 047670  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
 6005 047676  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6006 047702  001404                          BEQ     70$                     ; branch if SCR is OK
 6007 047704                                  ERRSOFT 1300,E1300,EGB          ; else print "SCR INCORRECT"
      047704  104457                                                          TRAP    C$ERSOFT
      047706  002424                                                          .WORD   1300
      047710  050214                                                          .WORD   E1300
      047712  023416                                                          .WORD   EGB
 6008 047714                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      047714  104406                                                          TRAP    C$CLP1
 6009
 6010 047716  012737  010000  003020          MOV     #ACS,GOOD               ; get expected ACS contents
 6011 047724  012777  010000  132702          MOV     #ACS,@DRXDBR            ; address the ACS register
 6012 047732  017737  132676  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6013 047740  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
 6014 047744  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
 6015 047750  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
 6016 047754  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 6017 047762  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6018 047766  001405                          BEQ     80$                     ; branch if ACS is OK
 6019 047770                                  ERRSOFT 1301,E1301,EGB          ; else print "ACS INCORRECT"
      047770  104457                                                          TRAP    C$ERSOFT
      047772  002425                                                          .WORD   1301
      047774  050275                                                          .WORD   E1301
      047776  023416                                                          .WORD   EGB
 6020 050000                                  CKLOOP                          ; branch to BGNSEG if loop on error set
      050000  104406                                                          TRAP    C$CLP1
 6021
 6022 050002  012737  020000  003020  80$:    MOV     #CTA,GOOD               ; get expected CTA contents
 6023 050010  012777  020000  132616          MOV     #CTA,@DRXDBR            ; address the CTA
 6024 050016  017737  132612  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6025 050024  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 6026 050032  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6027 050036  001404                          BEQ     90$                     ; branch if CTA is OK
```

TEST 13 - CWR Mode 3, External Clock and SEQ CONT Test.

```
                                        ERRSOFT 1302,E1302,EGB          ; else print "CTA INCORRECT"
050040  104457                                                                  TRAP    C$ERSOFT
050042  002426                                                                  .WORD   1302
050044  050346                                                                  .WORD   E1302
050046  023416                                                                  .WORD   EGB

                                90$:    ENDSEG
050050                                                                  10000$:
050050  104405                                                                  TRAP    C$ESEG

050052  005737  002234                TST     QVMODE                  ; is quick verify mode selected ?
050056  001006                        BNE     100$                    ; if yes, exit test
050060  005237  003032                INC     ITRCNT                  ; else, increment iteration counter
050064  023727  003032  000010        CMP     ITRCNT,#10              ; iterations completed ?
050072  001211                        BNE     20$                     ; if not, do another iteration

                                100$:   EXIT TST
050074  104432                                                                  TRAP    C$EXIT
050076  000322                                                                  .WORD   L10046-.

                                ; Outputs :   Mode    Channel  Voltage
                                ;                               (mV)

050100  000003  000000  001750  T13TAB: .WORD   3,      0,       1000.   ; table for outputs
050106  000000  000000  003720          .WORD   0,      0,       2000.
050114  000003  000000  000000          .WORD   3,      0,       0       ; not used
050122  000001  000001  000000          .WORD   1,      1,       0       ; ..   ..

                                        .NLIST  BEX
050130      045     123     062  TSHD13::.ASCIZ  /%S2%ACWR MODE 3, EXTERNAL CLOCK AND SEQ CONT TEST%N/
050214      104     122     130  E1300:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER SEQ CONT CONVERSIONS/
050275      101     103     123  E1301:  .ASCIZ  /ACS INCORRECT AFTER SEQ CONT CONVERSIONS/
050346      103     124     101  E1302:  .ASCIZ  /CTA INCORRECT AFTER SEQ CONT CONVERSIONS/
                                        .LIST   BEX
                                        .EVEN

                                        ENDTST
050420                                                                  L10046:
050420  104401                                                                  TRAP    C$ETST
```

TEST 14 - Programmable Clock Test.

```
                                .SBTTL  TEST 14 - Programmable Clock Test.

                                ;++
                                ; This tests the speed and count functionality of the programmable
                                ; clock. The PCR is loaded with the value 4000 to give a
                                ; conversion time of 400 microseconds. The control table is set up
                                ; to perform 2 conversions at this rate and a check is made that
                                ; the conversions require between 1.0 and 1.5 milliseconds to
                                ; complete.
                                ;
                                ; The 2 conversions are made in mode 0, outputing a value of 5.0
                                ; volts on channels 0 and 1.
                                ;--

050422                                  BGNTST
050422                                                                  T14::
050422  005037  003030                    CLR     STAFLG              ; flag test has been run since start
050426                                      RFLAGS  R0                  ; read operator flags
050426  104421                                                                 TRAP    C$RFLA
050430  032700  001000                    BIT     #PNT,R0             ; print test headers ?
050434  001410                            BEQ     10$                 ; if not, branch
050436                                      PRINTF  #TSHD14             ; else, print test header
050436  012746  051252                                                        MOV     #TSHD14,-(SP)
050442  012746  000001                                                        MOV     #1,-(SP)
050446  010600                                                                MOV     SP,R0
050450  104417                                                                TRAP    C$PNTF
050452  062706  000004                                                        ADD     #4,SP

050456  005037  003032  132136    10$:    CLR     ITRCNT              ; clear iteration counter
050462  052777  000040  132136            BIS     #RES,@DRXSCR        ; reset the DRX11-C
050470  004737  026106                    JSR     PC,WT500            ; wait for 1 millisecond
050474  004737  026106                    JSR     PC,WT500            ; (2*500 microseconds)

050500                                  SETVEC  DRXVEC,#INT,#PRI07  ; set up DRX11-C vector
050500  012746  000340                                                        MOV     #PRI07,-(SP)
050504  012746  032216                                                        MOV     #INT,-(SP)
050510  013746  002636                                                        MOV     DRXVEC,-(SP)
050514  012746  000003                                                        MOV     #3,-(SP)
050520  104437                                                                TRAP    C$SVEC
050522  062706  000010                                                        ADD     #10,SP

                                ;
                                ; Set up DMA output buffer
                                ;
050526  012701  011610                  MOV     #5000.,R1           ; set up for voltage of 5000 millivolts
050532  005002                          CLR     R2                  ; 0 microvolts in conversions
050534  004737  026510                  JSR     PC,ADCON            ; convert to 12 bit data in R1
050540  012703  003040                  MOV     #BUFOUT,R3          ; address of DMA buffer in R3
050544  012704  000002                  MOV     #2,R4               ; number of words to set up
050550  010123          20$:    MOV     R1,(R3)+            ; set up the data in the buffer
050552  005304                  DEC     R4                  ; 2 words set up ?
050554  001375                  BNE     20$                 ; if not, set up another

050556  012777  047640  132050  30$:    MOV     #PCR!4000.,@DRXDBR  ; conversion time of 400 microseconds
050564  052777  000001  132034          BIS     #FNCT0,@DRXSCR      ; transfer to PCR
                                ;
                                ; Set up control table
                                ;
```

TEST 14 - Programmable Clock Test.

```
 6104 050572  012701  000003                MOV     #3,R1                   ; set up 2 control words
 6105 050576  012702  020000                MOV     #CTA,R2                 ; set up for CWR0
 6106 050602  012703  030000                MOV     #CWR,R3                 ; mode 0, channel 0
 6107 050606  010277  132022          40$:  MOV     R2,@DRXDBR              ; address the CTA
 6108 050612  052777  000001  132006        BIS     #FNCT0,@DRXSCR          ; and transfer the control word address
 6109 050620  010377  132010                MOV     R3,@DRXDBR              ; address the control word register
 6110 050624  052777  000001  131774        BIS     #FNCT0,@DRXSCR          ; and transfer the control word
 6111 050632  005202                        INC     R2                      ; next control word
 6112 050634  005203                        INC     R3                      ; next channel
 6113 050636  005301                        DEC     R1                      ; 3 control words set up ?
 6114 050640  001362                        BNE     40$                     ; if not, do some more
 6115                               ;
 6116                               ; Set up DRX11-C and start the DMA output
 6117                               ;
 6118 050642                                BGNSEG
      050642  104404                                                        TRAP    C$BSEG
 6119 050644  012777  020000  131762        MOV     #CTA,@DRXDBR            ; reset the CTA register
 6120 050652  052777  000001  131746        BIS     #FNCT0,@DRXSCR          ; back to control word 0
 6121 050660                                SETPRI  #PRI00                  ; drop the priority
      050660  012700  000000                                                MOV     #PRI00,R0
      050664  104441                                                        TRAP    C$SPRI
 6122 050666  005037  002772                CLR     INTFLG                  ; clear the interrupt flag
 6123 050672  012777  010001  131734        MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
 6124 050700  012777  000001  131720        MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
 6125 050706  012777  000000  131714        MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
 6126 050714  012777  003040  131710        MOV     #BUFOUT,@DRXADR         ; set up DMA address
 6127 050722  012777  000400  131700        MOV     #WCR1,@DRXCOR           ; now select WCR1
 6128 050730  012777  177776  131674        MOV     #-2,@DRXADR             ; to DMA 2 words
 6129 050736  012777  020000  131664        MOV     #RUN,@DRXCOR            ; start the DMA
 6130 050744  052777  000100  131654        BIS     #IE,@DRXSCR             ; enable interrupts
 6131                               ;
 6132                               ; Wait for interrupts
 6133                               ;
 6134 050752  004737  026106                JSR     PC,WT500                ; wait for 1 millisecond
 6135 050756  004737  026106                JSR     PC,WT500                ; (2*500 microseconds)
 6136 050762  023727  002772  000002        CMP     INTFLG,#2               ; 2 interrupts ?
 6137 050770  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 6138 050774  001005                        BNE     60$                     ; if not 2 interrupts, branch
 6139 050776                                ERRSOFT 1400,E1400,ECHK         ; else print "CONVERSIONS TOO FAST"
      050776  104457                                                        TRAP    C$ERSOFT
      051000  002570                                                        .WORD   1400
      051002  051311                                                        .WORD   E1400
      051004  023404                                                        .WORD   ECHK
 6140 051006                                CKLOOP                          ; branch to BGNSEG if loop on error set
      051006  104406                                                        TRAP    C$CLP1
 6141
 6142 051010  004737  026106          60$:  JSR     PC,WT500                ; wait for another 500 microseconds
 6143 051014                                SETPRI  #PRI07                  ; stop any more interrupts
      051014  012700  000340                                                MOV     #PRI07,R0
      051020  104441                                                        TRAP    C$SPRI
 6144 051022  042777  000100  131576        BIC     #IE,@DRXSCR             ; from the DRX11-C
 6145 051030  023727  002772  000002        CMP     INTFLG,#2               ; did we get 2 interrupts ?
 6146 051036  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 6147 051042  001405                        BEQ     70$                     ; branch if we got 2 interrupts
 6148 051044                                ERRSOFT 1401,E1401,ECHK         ; else print "CONVERSIONS TOO SLOW"
      051044  104457                                                        TRAP    C$ERSOFT
      051046  002571                                                        .WORD   1401
```

TEST 14 - Programmable Clock Test.

```
                                                                                .WORD   E1401
                                                                                .WORD   ECHK
6149 051054                             CKLOOP                          ; branch to BGNSEG if loop on error set
     051054  104406                                                            TRAP    C$CLP1
6150                                    ;
6151                                    ; Check ACS and CTA registers
6152                                    ;
6153 051056  012737  010000  003020  70$:    MOV     #ACS,GOOD               ; get expected ACS contents
6154 051064  012777  010000  131542         MOV     #ACS,@DRXDBR            ; address the ACS register
6155 051072  017737  131536  003022         MOV     @DRXDBR,BAD             ; and get the actual contents
6156 051100  013700  003022                 MOV     BAD,R0                  ; get the contents into R0
6157 051104  042700  171777                 BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
6158 051110  050037  003020                 BIS     R0,GOOD                 ; set them in GOOD if they are set
6159 051114  023737  003022  003020         CMP     BAD,GOOD                ; contents correct ?
6160 051122  004737  026306                 CALL    INSERT                  ; skip branch if error insert selected
6161 051126  001404                         BEQ     80$                     ; branch if ACS is OK
6162 051130                                 ERRSOFT 1402,E1402,EGB          ; else print "ACS INCORRECT"
     051130  104457                                                            TRAP    C$ERSOFT
     051132  002572                                                            .WORD   1402
     051134  051417                                                            .WORD   E1402
     051136  023416                                                            .WORD   EGB
6163 051140                             80$:    CKLOOP                          ; branch to BGNSEG if loop on error set
     051140  104406                                                            TRAP    C$CLP1
6164
6165 051142  012737  020002  003020             MOV     #CTA!2,GOOD             ; get expected CTA contents
6166 051150  012777  020000  131456             MOV     #CTA,@DRXDBR            ; address the CTA
6167 051156  017737  131452  003022             MOV     @DRXDBR,BAD             ; and get the actual contents
6168 051164  023737  003022  003020             CMP     BAD,GOOD                ; contents correct ?
6169 051172  004737  026306                     CALL    INSERT                  ; skip branch if error insert selected
6170 051176  001404                             BEQ     90$                     ; branch if CTA is OK
6171 051200                                     ERRSOFT 1403,E1403,EGB          ; else print "CTA INCORRECT"
     051200  104457                                                                TRAP    C$ERSOFT
     051202  002573                                                                .WORD   1403
     051204  051453                                                                .WORD   E1403
     051206  023416                                                                .WORD   EGB
6172 051210                             90$:    ENDSEG
     051210                                                                 10000$:
     051210  104405                                                                TRAP    C$ESEG
6173
6174 051212  005737  002234                     TST     QVMODE                  ; is quick verify mode selected ?
6175 051216  001010                             BNE     110$                    ; if yes, exit test
6176 051220  005237  003032                     INC     ITRCNT                  ; else, increment iteration counter
6177 051224  023727  003032  000010             CMP     ITRCNT,#10              ; iterations completed ?
6178 051232  001402                             BEQ     110$                    ; if yes, exit
6179 051234  000137  050556                     JMP     30$                     ; else do another iteration
6180
6181 051240                             110$:   CLRVEC  DRXVEC                  ; restore the DRS trap catcher
     051240  013700  002636                                                        MOV     DRXVEC,R0
     051244  104436                                                            TRAP    C$CVEC
6182 051246                                     EXIT TST
     051246  104432                                                            TRAP    C$EXIT
     051250  000240                                                            .WORD   L10047-.
6183
6184                                            .NLIST  BEX
6185 051252     045     123     062  TSHD14::.ASCIZ  /%S2%APROGRAMMABLE CLOCK TEST%N/
6186 051311     103     117     116  E1400:  .ASCIZ  /CONVERSIONS TOO FAST IN CLOCK TEST/
6187 051354     103     117     116  E1401:  .ASCIZ  /CONVERSIONS TOO SLOW IN CLOCK TEST/
```

```
 6188 051417     101     103     123  E1402:  .ASCIZ  /ACS INCORRECT IN CLOCK TEST/
 6189 051453     103     124     101  E1403:  .ASCIZ  /CTA INCORRECT IN CLOCK TEST/
 6190                                         .LIST   BEX
 6191                                         .EVEN
 6192
 6193 051510                                 ENDTST
      051510                                                          L10047:
      051510  104401                                                          TRAP    C$ETST
 6194
```

TEST 15 - CWR Mode 4 Test.

```
                                .SBTTL  TEST 15 - CWR Mode 4 Test.

                                ;++
                                ; This tests that dummy control table loads can be made by using
                                ; mode 4. The control table and values for output are set up as
                                ; follows :
                                ;
                                ; Control word    Mode    Channel    Output voltage
                                ;
                                ;       0          0         0           1.0
                                ;       1          0         1           2.0
                                ;       2          4         1           2.0
                                ;       3          4         1           2.0
                                ;       4          0         2           3.0
                                ;       5          1         3           4.0
                                ;
                                ; The SCR interrupt enable bit is set and the DMA is started with
                                ; the DRX11-C WCR loaded for 4 DMA's. Conversions are then
                                ; initiated and the program waits for the       conversions to complete.
                                ;--

                                        BGNTST
                                                                                        T15::
                                        CLR     STAFLG                  ; flag test has been run since start
                                        RFLAGS  R0                      ; read operator flags
                                                                                TRAP    C$RFLA
                                        BIT     #PNT,R0                 ; print test headers ?
                                        BEQ     10$                     ; if not, branch
                                        PRINTF  #TSHD15                 ; else, print test header
                                                                                MOV     #TSHD15,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP

                                10$:    CLR     ITRCNT                  ; clear iteration counter
                                        BIS     #RES,@DRXSCR            ; reset the DRX11-C
                                        JSR     PC,WT500                ; wait for 1 millisecond
                                        JSR     PC,WT500                ; (2*500 microseconds)

                                        JSR     PC,SETTAB               ; set up control table and data
                                        6                               ; for 6 control words
                                        T15TAB                          ; using table at T15TAB

                                        MOV     CTIME,R1                ; get the selected conversion time
                                        BIS     #PCR,R1                 ; address the PCR
                                        MOV     R1,@DRXDBR              ; and enter the conversion time
                                        BIS     #FNCT0,@DRXSCR          ; transfer to PCR

                                20$:    BGNSEG
                                                                                TRAP    C$BSEG
                                        MOV     #CTA,@DRXDBR            ; reset the CTA register
                                        BIS     #FNCT0,@DRXSCR          ; back to control word 0
                                        MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
                                        MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
                                        MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
                                        MOV     #BUFOUT,@DRXADR         ; set up DMA address
```

TEST 15 - CWR Mode 4 Test.

```
 6245 051670  012777  000400  130732          MOV     #WCR1,@DRXCOR           ; now select WCR1
 6246 051676  012777  177774  130726          MOV     #-4,@DRXADR             ; to DMA 4 words
 6247 051704  012777  020000  130716          MOV     #RUN,@DRXCOR            ; start the DMA
 6248
 6249 051712  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 6250 051716  005002                          CLR     R2                      ; high byte is zero
 6251 051720  004737  027776                  JSR     PC,MUL                  ; multiply by
 6252 051724  000006                          6                               ; the number of conversions
 6253 051726  004737  030072                  JSR     PC,DIV                  ; divide by
 6254 051732  000175                          125.                            ; 125
 6255
 6256 051734  005701                  30$:    TST     R1                      ; loop until R1/R2 is zero
 6257 051736  001404                          BEQ     50$                     ;
 6258 051740  005301                  40$:    DEC     R1                      ;
 6259 051742  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 6260 051746  000772                          BR      30$                     ;
 6261 051750  005702                  50$:    TST     R2                      ;
 6262 051752  001402                          BEQ     60$                     ; total timeout = number of conversions
 6263 051754  005302                          DEC     R2                      ;                 * conversion time * 2
 6264 051756  000770                          BR      40$                     ;
 6265
 6266 051760  012737  100600  003020  60$:    MOV     #ATT!STAT0!RDY,GOOD     ; get expected SCR contents
 6267 051766  017737  130634  003022          MOV     @DRXSCR,BAD             ; and actual contents
 6268 051774  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
 6269 052002  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6270 052006  001404                          BEQ     70$                     ; branch if SCR is OK
 6271 052010                                  ERRSOFT 1500,E1500,EGB          ; else print "SCR INCORRECT"
      052010  104457                                                          TRAP    C$ERSOFT
      052012  002734                                                          .WORD   1500
      052014  052277                                                          .WORD   E1500
      052016  023416                                                          .WORD   EGB
 6272 052020                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      052020  104406                                                          TRAP    C$CLP1
 6273
 6274 052022  012737  010000  003020          MOV     #ACS,GOOD               ; get expected ACS contents
 6275 052030  012777  010000  130576          MOV     #ACS,@DRXDBR            ; address the ACS register
 6276 052036  017737  130572  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6277 052044  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
 6278 052050  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
 6279 052054  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
 6280 052060  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 6281 052066  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6282 052072  001405                          BEQ     80$                     ; branch if ACS is OK
 6283 052074                                  ERRSOFT 1501,E1501,EGB          ; else print "ACS INCORRECT"
      052074  104457                                                          TRAP    C$ERSOFT
      052076  002735                                                          .WORD   1501
      052100  052356                                                          .WORD   E1501
      052102  023416                                                          .WORD   EGB
 6284 052104                                  CKLOOP                          ; branch to BGNSEG if loop on error set
      052104  104406                                                          TRAP    C$CLP1
 6285
 6286 052106  012737  020000  003020  80$:    MOV     #CTA,GOOD               ; get expected CTA contents
 6287 052114  012777  020000  130512          MOV     #CTA,@DRXDBR            ; address the CTA
 6288 052122  017737  130506  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6289 052130  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 6290 052136  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6291 052142  001404                          BEQ     90$                     ; branch if CTA is OK
```

TEST 15 - CWR Mode 4 Test.

```
                                        ERRSOFT 1502,E1502,EGB          ; else print "CTA INCORRECT"
052144  104457                                                                  TRAP    C$ERSOFT
052146  002736                                                                  .WORD   1502
052150  052424                                                                  .WORD   E1502
052152  023416                                                                  .WORD   EGB

                                90$:    ENDSEG
052154                                                                  10000$:
052154  104405                                                                  TRAP    C$ESEG

052156  005737  002234                TST     QVMODE                  ; is quick verify mode selected ?
052162  001006                        BNE     100$                    ; if yes, exit test
052164  005237  003032                INC     ITRCNT                  ; else, increment iteration counter
052170  023727  003032  000010        CMP     ITRCNT,#10              ; iterations completed ?
052176  001211                        BNE     20$                     ; if not, do another iteration

                                100$:   EXIT TST
052200  104432                                                                  TRAP    C$EXIT
052202  000270                                                                  .WORD   L10050-.

                                ; Outputs :   Mode   Channel  Voltage
                                ;                              (mV)

052204  000000  000000  001750  T15TAB: .WORD   0,      0,      1000.   ; table for outputs
052212  000000  000001  003720          .WORD   0,      1,      2000.
052220  000004  000001  005670          .WORD   4,      1,      3000.
052226  000004  000001  007640          .WORD   4,      1,      4000.
052234  000000  000002  000000          .WORD   0,      2,      0       ; not used
052242  000001  000003  000000          .WORD   1,      3,      0       ; ... ....

                                        .NLIST  BEX
052250      045     123     062  TSHD15::.ASCIZ  /%S2%ACWR MODE 4 TEST%N/
052277      104     122     130  E1500:  .ASCIZ  /DRX11-C SCR INCORRECT AF[illegible]ODE 4 CONVERSIONS/
052356      101     103     123  E1501:  .ASCIZ  /ACS INCORRECT AFTER MOD[illegible] DMA OUTPUT/
052424      103     124     101  E1502:  .ASCIZ  /CTA INCORRECT AFTER MODE 4 DMA OUTPUT/
                                        .LIST   BEX
                                        .EVEN

                                        ENDTST
052472                                                          L10050:
052472                                                                  TRAP    C$ETST
052472  104401
```

TEST 16 - CWR Mode 5 Test.

```
                                .SBTTL  TEST 16 - CWR Mode 5 Test.

                                ;++
                                ; This tests that dummy control table loads can be made by using
                                ; mode 5. The control table and values for output are set up as
                                ; follows :
                                ;
                                ; Control word    Mode    Channel        Output voltage
                                ;                                    (first pass)  (second pass)
                                ;
                                ;       0         0         0           1.0           4.0
                                ;       1         0         1           2.0           5.0
                                ;       2         4         1           2.0           5.0
                                ;       3         4         1           2.0           5.0
                                ;       4         0         2           3.0           6.0
                                ;       5         5         2           3.0           6.0
                                ;
                                ; The SCR interrupt enable bit is set and the DMA is started with
                                ; the DRX11-C WCR loaded for 6 DMA's to use the control table
                                ; twice. Conversions are then initiated and the program waits for
                                ; the conversions to complete.
                                ;--

052474                                  BGNTST
052474                                                                          T16::
052474  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
052500                                  RFLAGS  R0                      ; read operator flags
052500  104421                                                                  TRAP    C$RFLA
052502  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
052506  001410                          BEQ     10$                     ; if not, branch
052510                                  PRINTF  #TSHD16                 ; else, print test header
052510  012746  053232                                                          MOV     #TSHD16,-(SP)
052514  012746  000001                                                          MOV     #1,-(SP)
052520  010600                                                                  MOV     SP,R0
052522  104417                                                                  TRAP    C$PNTF
052524  062706  000004                                                          ADD     #4,SP

052530  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter
052534  052777  000040  130064          BIS     #RES,@DRXSCR            ; reset the DRX11-C
052542  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
052546  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)

052552  004737  030320                  JSR     PC,SETTAB               ; set up control table and data
052556  000006                          6                               ; for 6 control words
052560  053166                          T16TAB                          ; using table at T16TAB

052562  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
052566  052701  040000                  BIS     #PCR,R1                 ; address the PCR
052572  010177  130036                  MOV     R1,@DRXDBR              ; and enter the conversion time
052576  052777  000001  130022          BIS     #FNCT0,@DRXSCR          ; transfer to PCR

052604                          20$:    BGNSEG
052604  104404                                                                  TRAP    C$BSEG
052606  012777  020000  130020          MOV     #CTA,@DRXDBR            ; reset the CTA register
052614  052777  000001  130004          BIS     #FNCT0,@DRXSCR          ; back to control word 0
052622  012777  010001  130004          MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
052630  012777  000001  127770          MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
```

TEST 16 - CWR Mode 5 Test.

```
 6375 052636  012777  000000  127764          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
 6376 052644  012777  003040  127760          MOV     #BUFOUT,@DRXADR         ; set up DMA address
 6377 052652  012777  000400  127750          MOV     #WCR1,@DRXCOR           ; now select WCR1
 6378 052660  012777  177772  127744          MOV     #-6,@DRXADR             ; to DMA 6 words
 6379 052666  012777  020000  127734          MOV     #RUN,@DRXCOR            ; start the DMA
 6380
 6381 052674  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 6382 052700  005002                          CLR     R2                      ; high byte is zero
 6383 052702  004737  027776                  JSR     PC,MUL                  ; multiply by
 6384 052706  000014                          12.                             ; the number of conversions
 6385 052710  004737  030072                  JSR     PC,DIV                  ; divide by
 6386 052714  000175                          125.                            ; 125
 6387
 6388 052716  005701                  30$:    TST     R1                      ; loop until R1/R2 is zero
 6389 052720  001404                          BEQ     50$                     ;
 6390 052722  005301                  40$:    DEC     R1                      ;
 6391 052724  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 6392 052730  000772                          BR      30$                     ;
 6393 052732  005702                  50$:    TST     R2                      ;
 6394 052734  001402                          BEQ     60$                     ; total timeout = number of conversions
 6395 052736  005302                          DEC     R2                      ;                 * conversion time * 2
 6396 052740  000770                          BR      40$                     ;
 6397
 6398 052742  012737  100600  003020  60$:    MOV     #ATT!STAT0!RDY,GOOD     ; get expected SCR contents
 6399 052750  017737  127652  003022          MOV     @DRXSCR,BAD             ; and actual contents
 6400 052756  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
 6401 052764  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6402 052770  001404                          BEQ     70$                     ; branch if SCR is OK
 6403 052772                                  ERRSOFT 1600,E1600,EGB          ; else print "SCR INCORRECT"
      052772  104457                                                          TRAP    C$ERSOFT
      052774  003100                                                          .WORD   1600
      052776  053261                                                          .WORD   E1600
      053000  023416                                                          .WORD   EGB
 6404 053002                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      053002  104406                                                          TRAP    C$CLP1
 6405
 6406 053004  012737  010000  003020          MOV     #ACS,GOOD               ; get expected ACS contents
 6407 053012  012777  010000  127614          MOV     #ACS,@DRXDBR            ; address the ACS register
 6408 053020  017737  127610  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6409 053026  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
 6410 053032  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
 6411 053036  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
 6412 053042  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 6413 053050  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6414 053054  001405                          BEQ     80$                     ; branch if ACS is OK
 6415 053056                                  ERRSOFT 1601,E1601,EGB          ; else print "ACS INCORRECT"
      053056  104457                                                          TRAP    C$ERSOFT
      053060  003101                                                          .WORD   1601
      053062  053340                                                          .WORD   E1601
      053064  023416                                                          .WORD   EGB
 6416 053066                                  CKLOOP                          ; branch to BGNSEG if loop on error set
      053066  104406                                                          TRAP    C$CLP1
 6417
 6418 053070  012737  020005  003020  80$:    MOV     #CTA!5,GOOD             ; get expected CTA contents
 6419 053076  012777  020000  127530          MOV     #CTA,@DRXDBR            ; address the CTA
 6420 053104  017737  127524  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6421 053112  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
```

TEST 16 - CWR Mode 5 Test.

```
 6422 053120  004737  026306                 CALL    INSERT                  ; skip branch if error insert selected
 6423 053124  001404                         BEQ     90$                     ; branch if CTA is OK
 6424 053126                                 ERRSOFT 1602,E1602,EGB          ; else print "CTA INCORRECT"
      053126  104457                                                                 TRAP    C$ERSOFT
      053130  003102                                                                 .WORD   1602
      053132  053406                                                                 .WORD   E1602
      053134  023416                                                                 .WORD   EGB
 6425
 6426 053136                         90$:    ENDSEG
      053136                                                                    10000$:
      053136  104405                                                                 TRAP    C$ESEG
 6427
 6428 053140  005737  002234                 TST     QVMODE                  ; is quick verify mode selected ?
 6429 053144  001006                         BNE     100$                    ; if yes, exit test
 6430 053146  005237  003032                 INC     ITRCNT                  ; else, increment iteration counter
 6431 053152  023727  003032  000010         CMP     ITRCNT,#10              ; iterations completed ?
 6432 053160  001211                         BNE     20$                     ; if not, do another iteration
 6433
 6434 053162                         100$:   EXIT TST
      053162  104432                                                                 TRAP    C$EXIT
      053164  000270                                                                 .WORD   L10051-.
 6435
 6436
 6437                                ; Outputs :   Mode   Channel  Voltage
 6438                                ;                              (mV)
 6439
 6440 053166  000000  000000  001750 T16TAB: .WORD   0,      0,      1000.   ; table for outputs
 6441 053174  000000  000001  003720         .WORD   0,      1,      2000.
 6442 053202  000004  000001  005670         .WORD   4,      1,      3000.
 6443 053210  000004  000001  007640         .WORD   4,      1,      4000.
 6444 053216  000000  000002  011610         .WORD   0,      2,      5000.
 6445 053224  000005  000002  013560         .WORD   5,      2,      6000.
 6446
 6447                                        .NLIST  BEX
 6448 053232     045     123     062 TSHD16::.ASCIZ  /%S2%ACWR MODE 5 TEST%N/
 6449 053261     104     122     130 E1600:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER MODE 5 CONVE SIONS/
 6450 053340     101     103     123 E1601:  .ASCIZ  /ACS INCORRECT AFTER MODE 5 DMA OUTPUT/
 6451 053406     103     124     101 E1602:  .ASCIZ  /CTA INCORRECT AFTER MODE 5 DMA OUTPUT/
 6452                                        .LIST   BEX
 6453                                        .EVEN
 6454
 6455 053454                                ENDTST
      053454                                                                    L10051:
      053454  104401                                                                 TRAP    C$ETST
 6456
```

```
                                .SBTTL  TEST 17 - CWR Mode 6 Test.

                                ;++
                                ; This tests that dummy control table loads can be made by using
                                ; mode 6. The PCR is loaded with the value 95 (complement of 4000)
                                ; to give a conversion time of 400 microseconds. The control table
                                ; is then set up to perform 5 conversions. All except the first
                                ; and last conversions are made in mode 6. The first and last are
                                ; made in mode 0 and mode 1, outputing a value of 1.0 volts and
                                ; 2.0 volts respectively on channels 0 and 1.
                                ;
                                ; The SCR interrupt enable bit is set and a DMA is started with
                                ; the DRX11-C WCR loaded for 2 DMA's. Conversions are then
                                ; initiated and the program checks that the conversions require
                                ; between 1 and 1.5 milliseconds to complete.
                                ;--

053456                                  BGNTST
053456                                                                  T17::
053456  005037  003030                 CLR     STAFLG                  ; flag test has been run since start
053462                                  RFLAGS  R0                      ; read operator flags
053462  104421                                                                 TRAP    C$RFLA
053464  032700  001000                 BIT     #PNT,R0                 ; print test headers ?
053470  001410                         BEQ     10$                     ; if not, branch
053472                                  PRINTF  #TSHD17                 ; else, print test header
053472  012746  054402                                                         MOV     #TSHD17,-(SP)
053476  012746  000001                                                         MOV     #1,-(SP)
053502  010600                                                                 MOV     SP,R0
053504  104417                                                                 TRAP    C$PNTF
053506  062706  000004                                                         ADD     #4,SP

053512  005037  003032         10$:    CLR     ITRCNT                  ; clear iteration counter
053516  052777  000040  127102         BIS     #RES,@DRXSCR            ; reset the DRX11-C
053524  004737  026106                 JSR     PC,WT500                ; wait for 1 millisecond
053530  004737  026106                 JSR     PC,WT500                ; (2*500 microseconds)

053534                                  SETVEC  DRXVEC,#INT,#PRI07      ; set up DRX11-C vector
053534  012746  000340                                                         MOV     #PRI07,-(SP)
053540  012746  032216                                                         MOV     #INT,-(SP)
053544  013746  002636                                                         MOV     DRXVEC,-(SP)
053550  012746  000003                                                         MOV     #3,-(SP)
053554  104437                                                                 TRAP    C$SVEC
053556  062706  000010                                                         ADD     #10,SP

                                ;
                                ; Set up the DMA output buffer
                                ;
053562  012701  001750                 MOV     #1000.,R1               ; set up for voltage of 1000 millivolts
053566  005002                         CLR     R2                      ; 0 microvolts
053570  004737  026510                 JSR     PC,ADCON                ; convert to 12 bit data in R1
053574  010137  003040                 MOV     R1,BUFOUT               ; and put it in the DMA buffer
053600  012701  003720                 MOV     #2000.,R1               ; set up for voltage of 2000 millivolts
053604  005002                         CLR     R2                      ; 0 microvolts
053606  004737  026510                 JSR     PC,ADCON                ; convert to 12 bit data in R1
053612  010137  003042                 MOV     R1,BUFOUT+2             ; and put it in the DMA buffer

053616  012777  040137  127010 20$:    MOV     #PCR!137,@DRXDBR        ; conversion time of 400 microseconds
053624  052777  000001  126774         BIS     #FNCT0,@DRXSCR          ; transfer to PCR
```

TEST 17 - CWR Mode 6 Test.

```
6502                                    ;
6503                                    ; Set up the control table
6504                                    ;
6505 053632  012777  020000  126774          MOV     #CTA,@DRXDBR            ; address control word 0
6506 053640  052777  000001  126760          BIS     #FNCT0,@DRXSCR          ; and transfer to the CTA register
6507 053646  012777  030000  126760          MOV     #CWR,@DRXDBR            ; set up for mode 0, channel 0
6508 053654  052777  000001  126744          BIS     #FNCT0,@DRXSCR          ; and transfer to CWR0
6509 053662  012701  000003                  MOV     #3,R1                   ; set up next 3 control words
6510 053666  012702  020001                  MOV     #CTA!1,R2               ; set up for CWR1
6511 053672  010277  126736          30$:    MOV     R2,@DRXDBR              ; address the CTA
6512 053676  052777  000001  126722          BIS     #FNCT0,@DRXSCR          ; and transfer the control word address
6513 053704  012777  030600  126722          MOV     #CWR!600,@DRXDBR        ; set up for mode 6
6514 053712  052777  000001  126706          BIS     #FNCT0,@DRXSCR          ; and transfer the control word
6515 053720  005202                          INC     R2                      ; next control word
6516 053722  005301                          DEC     R1                      ; 3 words set up in mode 6 ?
6517 053724  001362                          BNE     30$                     ; if not, do some more
6518 053726  012777  020004  126700          MOV     #CTA!4,@DRXDBR          ; address control word 4
6519 053734  052777  000001  126664          BIS     #FNCT0,@DRXSCR          ; and transfer to the CTA register
6520 053742  012777  030101  126664          MOV     #CWR!101,@DRXDBR        ; set up for mode 1, channel 1
6521 053750  052777  000001  126650          BIS     #FNCT0,@DRXSCR          ; and transfer to CWR 4
6522 053756  012777  020000  126650          MOV     #CTA,@DRXDBR            ; reinitiate the CTA
6523 053764  052777  000001  126634          BIS     #FNCT0,@DRXSCR          ; to CWR 0
6524                                    ;
6525                                    ; Set up the DRX11-C and start the DMA output
6526                                    ;
6527 053772                          40$:    BGNSEG
     053772  104404                                                          TRAP    C$BSEG
6528 053774  012777  020000  126632          MOV     #CTA,@DRXDBR            ; reset the CTA register
6529 054002  052777  000001  126616          BIS     #FNCT0,@DRXSCR          ; back to control word 0
6530 054010                                  SETPRI  #PRI00                  ; drop the priority
     054010  012700  000000                                                          MOV     #PRI00,R0
     054014  104441                                                                  TRAP    C$SPRI
6531 054016  005037  002772                  CLR     INTFLG                  ; clear the interrupt flag
6532 054022  012777  010001  126604          MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
6533 054030  012777  000001  126570          MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
6534 054036  012777  000000  126564          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
6535 054044  012777  003040  126560          MOV     #BUFOUT,@DRXADR         ; set up DMA address
6536 054052  012777  000400  126550          MOV     #WCR1,@DRXCOR           ; now select WCR1
6537 054060  012777  177776  126544          MOV     #-2,@DRXADR             ; to DMA 2 words
6538 054066  012777  020000  126534          MOV     #RUN,@DRXCOR            ; start the DMA
6539 054074  052777  000100  126524          BIS     #IE,@DRXSCR             ; enable interrupts
6540                                    ;
6541                                    ; Wait for interrupts
6542                                    ;
6543 054102  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
6544 054106  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)
6545 054112  023727  002772  000002          CMP     INTFLG,#2               ; 2 interrupts ?
6546 054120  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
6547 054124  001005                          BNE     50$                     ; if not 2 interrupts, branch
6548 054126                                  ERRSOFT 1700,E1700,ECHK         ; else print "CONVERSIONS TOO FAST"
     054126  104457                                                                  TRAP    C$ERSOFT
     054130  003244                                                                  .WORD   1700
     054132  054431                                                                  .WORD   E1700
     054134  023404                                                                  .WORD   ECHK
6549 054136                                  CKLOOP                          ; branch to BGNSEG if loop on error set
     054136  104406                                                                  TRAP    C$CLP1
6550
```

TEST 17 - CWR Mode 6 Test.

```
                                50$:    JSR     PC,WT500                ; wait another 500 microseconds
                                        SETPRI  #PRI07                  ; stop any more interrupts
054144  012700  000340                                                          MOV     #PRI07,R0
054150  104441                                                                  TRAP    C$SPRI
054152  042777  000100  126446          BIC     #IE,@DRXSCR             ; from the DRX11-C
054160  023727  002772  000002          CMP     INTFLG,#2               ; did we get 2 interrupts ?
054166  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
054172  001405                          BEQ     60$                     ; branch if we got 2 interrupts
054174                                  ERRSOFT 1701,E1701,ECHK         ; else print "CONVERSIONS TOO SLOW"
054174  104457                                                                  TRAP    C$ERSOFT
054176  003245                                                                  .WORD   1701
054200  054474                                                                  .WORD   E1701
054202  023404                                                                  .WORD   ECHK
054204                                  CKLOOP                          ; branch to BGNSEG if loop on error set
054204  104406                                                                  TRAP    C$CLP1
                                ;
                                ; Check the ACS and CTA registers
                                ;
054206  012737  010000  003020  60$:    MOV     #ACS,GOOD               ; get expected ACS contents
054214  012777  010000  126412          MOV     #ACS,@DRXDBR            ; address the ACS register
054222  017737  126406  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
054230  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
054234  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
054240  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
054244  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
054252  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
054256  001404                          BEQ     70$                     ; branch if ACS is OK
054260                                  ERRSOFT 1702,E1702,EGB          ; else print "ACS INCORRECT"
054260  104457                                                                  TRAP    C$ERSOFT
054262  003246                                                                  .WORD   1702
054264  054537                                                                  .WORD   E1702
054266  023416                                                                  .WORD   EGB
054270                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
054270  104406                                                                  TRAP    C$CLP1
054272  012737  020000  003020          MOV     #CTA,GOOD               ; get expected CTA contents
054300  012777  020000  126326          MOV     #CTA,@DRXDBR            ; address the CTA
054306  017737  126322  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
054314  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
054322  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
054326  001404                          BEQ     80$                     ; branch if CTA is OK
054330                                  ERRSOFT 1703,E1703,EGB          ; else print "CTA INCORRECT"
054330  104457                                                                  TRAP    C$ERSOFT
054332  003247                                                                  .WORD   1703
054334  054605                                                                  .WORD   E1703
054336  023416                                                                  .WORD   EGB
054340                          80$:    ENDSEG
054340                                                                  10000$:
054340  104405                                                                  TRAP    C$ESEG
054342  005737  002234                  TST     QVMODE                  ; is quick verify mode selected ?
054346  001010                          BNE     90$                     ; if yes, exit test
054350  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
054354  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
054362  001402                          BEQ     90$                     ; if yes, exit test
054364  000137  053772                  JMP     40$                     ; else, do another iteration
```

TEST 17 - CWR Mode 6 Test.

```
 6590 054370                            90$:    CLRVEC  DRXVEC                  ; restore the DRS trap catcher
      054370  013700  002636                                                            MOV     DRXVEC,R0
      054374  104436                                                                    TRAP    C$CVEC
 6591 054376                                    EXIT TST
      054376  104432                                                                    TRAP    C$EXIT
      054400  000254                                                                    .WORD   L10052-.
 6592
 6593                                           .NLIST  BEX
 6594 054402     045     123     062  TSHD17::.ASCIZ  /%S2%ACWR MODE 6 TEST%N/
 6595 054431     103     117     116  E1700:  .ASCIZ  /CONVERSIONS TOO FAST IN CWR MODE 6/
 6596 054474     103     117     116  E1701:  .ASCIZ  /CONVERSIONS TOO SLOW IN CWR MODE 6/
 6597 054537     101     103     123  E1702:  .ASCIZ  /ACS INCORRECT AFTER MODE 6 DMA OUTPUT/
 6598 054605     103     124     101  E1703:  .ASCIZ  /CTA INCORRECT AFTER MODE 6 DMA OUTPUT/
 6599                                           .LIST   BEX
 6600                                           .EVEN
 6601
 6602 054654                                    ENDTST
      054654                                                            L10052:
      054654  104401                                                                  TRAP    C$ETST
 6603
```

TEST 18 - CWR Mode 7 Test.

```
                                .SBTTL  TEST 18 - CWR Mode 7 Test.

                                ;++
                                ; This tests that dummy control table loads can be made by using
                                ; mode 7. The control table and values for output are set up as
                                ; follows :
                                ;
                                ; Control word    Mode    Channel    Output voltage
                                ;
                                ;       0          0         0           1.0
                                ;       1          7         0           1.0 *
                                ;       2          7         0           1.0
                                ;       3          0         1           2.0 -  should
                                ;       4          0         2           3.0  > not be
                                ;       5          0         3           4.0 -  output
                                ;
                                ; The SCR interrupt enable and ACS SBE bits are set end the DMA is
                                ; started with the DRX11-C WCR loaded for 3 DMA's. Conversions are
                                ; then initiated. As the second conversion completes, COUT goes
                                ; low causing the input SEQ CONT L to go low. The diagnostic
                                ; checks that an SBR interrupt then occurs and that conversions
                                ; stop with the CTA register pointing to the 4th control word.
                                ;--

                                        BGNTST
                                                                                        T18::
                                CLR     STAFLG                  ; flag test has been run since start
                                RFLAGS  R0                      ; read operator flags
                                                                        TRAP    C$RFLA
                                BIT     #PNT,R0                 ; print test headers ?
                                BEQ     10$                     ; if not, branch
                                PRINTF  #TSHD18                 ; else, print test header
                                                                        MOV     #TSHD18,-(SP)
                                                                        MOV     #1,-(SP)
                                                                        MOV     SP,R0
                                                                        TRAP    C$PNTF
                                                                        ADD     #4,SP

                        10$:    CLR     ITRCNT                  ; clear iteration counter
                                BIS     #RES,@DRXSCR            ; reset the DRX11-C
                                JSR     PC,WT500                ; wait for 1 millisecond
                                JSR     PC,WT500                ; (2*500 microseconds)

                                JSR     PC,SETTAB               ; set up control table and data
                                6                               ; for 6 control words
                                T18TAB                          ; using table at T18TAB

                                MOV     CTIME,R1                ; get the selected conversion time
                                BIS     #PCR,R1                 ; address the PCR
                                MOV     R1,@DRXDBR              ; and enter the conversion time
                                BIS     #FNCT0,@DRXSCR          ; transfer to PCR

                        20$:    BGNSEG
                                                                        TRAP    C$BSEG
                                MOV     #CTA,@DRXDBR            ; reset the CTA register
                                BIS     #FNCT0,@DRXSCR          ; back to control word 0
                                MOV     #ACS!SBE!GO,@DRXDBR     ; set up to start the conversions
```

TEST 18 - CWR Mode 7 Test.

```
 6654 055012  012777  000001  125606          MOV     #FNCTO,@DRXSCR          ; enable conversions and clear the SCR
 6655 055020  012777  000000  125602          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
 6656 055026  012777  003040  125576          MOV     #BUFOUT,@DRXADR         ; set up DMA address
 6657 055034  012777  000400  125566          MOV     #WCR1,@DRXCOR           ; now select WCR1
 6658 055042  012777  177775  125562          MOV     #-3,@DRXADR             ; to DMA 3 words
 6659 055050  012777  020000  125552          MOV     #RUN,@DRXCOR            ; start the DMA
 6660
 6661 055056  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 6662 055062  005002                          CLR     R2                      ; high byte is zero
 6663 055064  004737  027776                  JSR     PC,MUL                  ; multiply by
 6664 055070  000003                          3                               ; the number of conversions
 6665 055072  004737  030072                  JSR     PC,DIV                  ; divide by
 6666 055076  000175                          125.                            ; 125
 6667
 6668 055100  005701                  30$:    TST     R1                      ; loop until R1/R2 is zero
 6669 055102  001404                          BEQ     50$                     ;
 6670 055104  005301                  40$:    DEC     R1                      ;
 6671 055106  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 6672 055112  000772                          BR      30$                     ;
 6673 055114  005702                  50$:    TST     R2                      ;
 6674 055116  001402                          BEQ     60$                     ; total timeout = number of conversions
 6675 055120  005302                          DEC     R2                      ;                 * conversion time * 2
 6676 055122  000770                          BR      40$                     ;
 6677
 6678 055124  012737  100600  003020  60$:    MOV     #ATT!STATO!RDY,GOOD     ; get expected SCR contents
 6679 055132  017737  125470  003022          MOV     @DRXSCR,BAD             ; and actual contents
 6680 055140  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
 6681 055146  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6682 055152  001404                          BEQ     70$                     ; branch if SCR is OK
 6683 055154                                  ERRSOFT 1800,E1800,EGB          ; else print "SCR INCORRECT"
      055154  104457                                                                  TRAP    C$ERSOFT
      055156  003410                                                                  .WORD   1800
      055160  055443                                                                  .WORD   E1800
      055162  023416                                                                  .WORD   EGB
 6684 055164                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      055164  104406                                                                  TRAP    C$CLP1
 6685
 6686 055166  012737  010250  003020          MOV     #ACS!SBR!SBE!EOC,GOOD   ; get expected ACS contents
 6687 055174  012777  010000  125432          MOV     #ACS,@DRXDBR            ; address the ACS register
 6688 055202  017737  125426  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6689 055210  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
 6690 055214  042700  171777                  BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
 6691 055220  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
 6692 055224  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 6693 055232  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6694 055236  001405                          BEQ     80$                     ; branch if ACS is OK
 6695 055240                                  ERRSOFT 1801,E1801,EGB          ; else print "ACS INCORRECT"
      055240  104457                                                                  TRAP    C$ERSOFT
      055242  003411                                                                  .WORD   1801
      055244  055522                                                                  .WORD   E1801
      055246  023416                                                                  .WORD   EGB
 6696 055250                                  CKLOOP                          ; branch to BGNSEG if loop on error set
      055250  104406                                                                  TRAP    C$CLP1
 6697
 6698 055252  012737  020003  003020  80$:    MOV     #CTA!3,GOOD             ; get expected CTA contents
 6699 055260  012777  020000  125346          MOV     #CTA,@DRXDBR            ; address the CTA
 6700 055266  017737  125342  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
```

TEST 18 - CWR Mode 7 Test.

```
6701 055274  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
6702 055302  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
6703 055306  001404                          BEQ     90$                     ; branch if CTA is OK
6704 055310                                  ERRSOFT 1802,E1802,EGB          ; else print "CTA INCORRECT"
     055310  104457                                                              TRAP    C$ERSOFT
     055312  003412                                                              .WORD   1802
     055314  055570                                                              .WORD   E1802
     055316  023416                                                              .WORD   EGB
6705
6706 055320                          90$:    ENDSEG
     055320                                                                  10000$:
     055320  104405                                                              TRAP    C$ESEG
6707
6708 055322  005737  002234                  TST     QVMODE                  ; is quick verify mode selected ?
6709 055326  001006                          BNE     100$                    ; if yes, exit test
6710 055330  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
6711 055334  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
6712 055342  001211                          BNE     20$                     ; if not, do another iteration
6713
6714 055344                          100$:   EXIT TST
     055344  104432                                                              TRAP    C$EXIT
     055346  000270                                                              .WORD   L10053-.
6715
6716
6717                                 ; Outputs :    Mode    Channel  Voltage
6718                                 ;                                (mV)
6719
6720 055350  000000  000000  001750  T18TAB: .WORD   0,      0,       1000.   ; table for outputs
6721 055356  000007  000000  003720          .WORD   7,      0,       2000.
6722 055364  000007  000000  005670          .WORD   7,      0,       3000.
6723 055372  000000  000001  007640          .WORD   0,      1,       4000.
6724 055400  000000  000002  000000          .WORD   0,      2,       0       ; not used
6725 055406  000000  000003  000000          .WORD   0,      3,       0       ; ...  ..
6726
6727                                         .NLIST  BEX
6728 055414      045     123     062  TSHD18::.ASCIZ  /%S2%ACWR MODE 7 TEST%N/
6729 055443      104     122     130  E1800:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER MODE 7 CONVERSIONS/
6730 055522      101     103     123  E1801:  .ASCIZ  /ACS INCORRECT AFTER MODE 7 DMA OUTPUT/
6731 055570      103     124     101  E1802:  .ASCIZ  /CTA INCORRECT AFTER MODE 7 DMA OUTPUT/
6732                                         .LIST   BEX
6733                                         .EVEN
6734
6735 055636                                  ENDTST
     055636                                                                  L10053:
     055636  104401                                                              TRAP    C$ETST
6736
```

```
                                .SBTTL  TEST 19 - Data Buffer Empty Test.

                                ;++
                                ; This tests that an interrupt can be generated when the AAF01-A
                                ; data buffer becomes empty.
                                ;
                                ; The ACS external clock enable bit is set to prevent conversions.
                                ; The control table is then set up for 66 conversions in mode 0,
                                ; to output a value of 5.0 volts on sequential (modulo 15)
                                ; channels.
                                ;
                                ; DMA's are then initiated and the diagnostic waits for 1
                                ; millisecond for the FIFO to be fully loaded. The DRX11-C COR
                                ; maintenance bit is then set to prevent further DMA's. The ECE
                                ; bit is then cleared to allow conversions. A check is made that
                                ; the DBE bit causes an interrupt within a specific timeout
                                ; period and that the CTA register indicates that 64 conversions
                                ; took place.
                                ;--

055640                                  BGNTST
055640                                                                  T19::
055640  005037  003030                   CLR     STAFLG                 ; flag test has been run since start
055644                                   RFLAGS  R0                     ; read operator flags
055644  104421                                                                 TRAP    C$RFLA
055646  032700  001000                   BIT     #PNT,R0                ; print test headers ?
055652  001410                           BEQ     10$                    ; if not, branch
055654                                   PRINTF  #TSHD19                ; else print test header
055654  012746  056632                                                         MOV     #TSHD19,-(SP)
055660  012746  000001                                                         MOV     #1,-(SP)
055664  010600                                                                 MOV     SP,R0
055666  104417                                                                 TRAP    C$PNTF
055670  062706  000004                                                         ADD     #4,SP

055674  005037  003032           10$:    CLR     ITRCNT                 ; clear iteration counter
055700  052777  000040  124720   20$:    BIS     #RES,@DRXSCR           ; reset the DRX11-C
055706  004737  026106                   JSR     PC,WT500               ; wait for 1 millisecond
055712  004737  026106                   JSR     PC,WT500               ; (2*500 microseconds)
                                ;
                                ; Set up the DMA output buffer
                                ;
055716                                   SETVEC  DRXVEC,#INT,#PRI07     ; set up DRX11-C vector
055716  012746  000340                                                         MOV     #PRI07,-(SP)
055722  012746  032216                                                         MOV     #INT,-(SP)
055726  013746  002636                                                         MOV     DRXVEC,-(SP)
055732  012746  000003                                                         MOV     #3,-(SP)
055736  104437                                                                 TRAP    C$SVEC
055740  062706  000010                                                         ADD     #10,SP

055744  012701  011610                   MOV     #5000.,R1              ; set up for voltage of 5000 millivolts
055750  005002                           CLR     R2                     ; 0 microvolts in conversions
055752  004737  026510                   JSR     PC,ADCON               ; convert to 12 bit data in R1
055756  052701  070000                   BIS     #70000,R1              ; set RSA bits for output with MNT bit
055762  012703  003040                   MOV     #BUFOUT,R3             ; address of DMA buffer in R3
055766  012704  000100                   MOV     #64.,R4                ; number of words to set up
055772  010123                   30$:    MOV     R1,(R3)+               ; set up the data in the buffer
055774  005304                           DEC     R4                     ; 64 words set up ?
```

TEST 19 - Data Buffer Empty Test.

```
 6782 055776  001375                          BNE     30$                     ; if not, set up another
 6783 056000  012723  050001                  MOV     #50001,(R3)+            ; else set up the 65th word
 6784 056004  010113                          MOV     R1,(R3)                 ; and the 66th
 6785
 6786 056006  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
 6787 056012  052701  040000                  BIS     #PCR,R1                 ; address the PCR
 6788 056016  010177  124612                  MOV     R1,@DRXDBR              ; and enter the conversion time
 6789 056022  052777  000001  124576          BIS     #FNCT0,@DRXSCR          ; transfer to PCR
 6790                                ;
 6791                                ; Set up the control table
 6792                                ;
 6793 056030  012701  000101          40$:    MOV     #65.,R1                 ; set up 65 control words
 6794 056034  012702  020000                  MOV     #CTA,R2                 ; set up for CWR0
 6795 056040  012703  030000                  MOV     #CWR,R3                 ; mode 0, channel 0
 6796 056044  010277  124564          50$:    MOV     R2,@DRXDBR              ; address the CTA
 6797 056050  052777  000001  124550          BIS     #FNCT0,@DRXSCR          ; and transfer the control word address
 6798 056056  010377  124552                  MOV     R3,@DRXDBR              ; address the control word register
 6799 056062  052777  000001  124536          BIS     #FNCT0,@DRXSCR          ; and transfer the control word
 6800 056070  005202                          INC     R2                      ; next control word
 6801 056072  005203                          INC     R3                      ; next channel
 6802 056074  042703  000020                  BIC     #20,R3                  ; don't go higher than channel 15
 6803 056100  005301                          DEC     R1                      ; 65 control words set up ?
 6804 056102  001360                          BNE     50$                     ; if not, do some more
 6805 056104  010277  124524                  MOV     R2,@DRXDBR              ; set 66th control word
 6806 056110  052777  000001  124510          BIS     #FNCT0,@DRXSCR          ; to reset channel 0 to
 6807 056116  012777  030000  124510          MOV     #CWR,@DRXDBR            ; 5 volts after the DBR output
 6808 056124  052777  000001  124474          BIS     #FNCT0,@DRXSCR          ; (see below)
 6809                                ;
 6810                                ; Set up the DRX11-C and start the DMA
 6811                                ;
 6812 056132                                  BGNSEG
      056132  104404                                                          TRAP    C$BSEG
 6813 056134  012777  020000  124472          MOV     #CTA,@DRXDBR            ; reset the CTA register
 6814 056142  052777  000001  124456          BIS     #FNCT0,@DRXSCR          ; back to control word 0
 6815 056150                                  SETPRI  #PRI00                  ; drop the priority
      056150  012700  000000                                                  MOV     #PRI00,R0
      056154  104441                                                          TRAP    C$SPRI
 6816 056156  005037  002772                  CLR     INTFLG                  ; clear the interrupt flag
 6817 056162  012777  011005  124444          MOV     #ACS!ECE!GO!MNT,@DRXDBR ; set up to start the conversions
 6818 056170  012777  000001  124430          MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
 6819 056176  012777  000000  124424          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
 6820 056204  012777  003040  124420          MOV     #BUFOUT,@DRXADR         ; set up DMA address
 6821 056212  012777  000400  124410          MOV     #WCR1,@DRXCOR           ; now select WCR1
 6822 056220  012777  177634  124404          MOV     #-100.,@DRXADR          ; to DMA 100 words
 6823 056226  012777  020000  124374          MOV     #RUN,@DRXCOR            ; start the DMA
 6824 056234  052777  000100  124364          BIS     #IE,@DRXSCR             ; enable interrupts
 6825
 6826 056242  004737  026106                  JSR     PC,WT500                ; wait 1 millisecond for FIFO to load
 6827 056246  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)
 6828 056252  052777  010000  124350          BIS     #MAINT,@DRXCOR          ; prevent further DMA's
 6829 056260  012777  010001  124346          MOV     #ACS!GO,@DRXDBR         ; clear ext. clock enable - this value in
 6830                                                                         ;                           the DBR causes
 6831                                                                         ;                           channel 0 to go
 6832 056266  013701  002230                  MOV     CTIME,R1                ; get the conversion time   momentarily low
 6833 056272  005002                          CLR     R2                      ; high byte is zero         after the 64th
 6834 056274  004737  027776                  JSR     PC,MUL                  ; multiply by               conversion
 6835 056300  000100                          64.                             ; the number of conversions
```

TEST 19 - Data Buffer Empty Test.

```
                                        JSR     PC,DIV                  ; divide by
                                        125.                            ; 125
                                60$:    TST     R1                      ; loop until R1/R2 is zero
                                        BEQ     80$                     ;
                                70$:    DEC     R1                      ;
                                        JSR     PC,WT25                 ; wait 25 microseconds
                                        BR      60$                     ;
                                80$:    TST     R2                      ;
                                        BEQ     90$                     ; total timeout = number of conversions
                                        DEC     R2                      ;                 * conversion time * 2
                                        BR      70$                     ;

                                90$:    SETPRI  #PRI07                  ; stop any interrupts
056334  012700  000340                                                          MOV     #PRI07,R0
056340  104441                                                                 TRAP    C$SPRI
                                        BIC     #IE,@DRXSCR             ; from the DRX11-C
                                        TST     INTFLG                  ; did we get an interrupt ?
                                        CALL    INSERT                  ; skip branch if error insert selected
                                        BNE     100$                    ; branch if we got an interrupt
                                        ERRSOFT 1900,E1900,ECHK         ; else print "NO INTERRUPT"
056362  104457                                                                 TRAP    C$ERSOFT
056364  003554                                                                 .WORD   1900
056366  056670                                                                 .WORD   E1900
056370  023404                                                                 .WORD   ECHK
                                100$:   CKLOOP                          ; branch to BGNSEG if loop on error set
056372  104406                                                                 TRAP    C$CLP1

056374  012737  110600  003020        MOV     #ATT!WAIT!STAT0!RDY,GOOD ; get expected SCR contents
056402  017737  124220  003022        MOV     @DRXSCR,BAD             ; and get the actual contents
056410  023737  003022  003020        CMP     BAD,GOOD                ; contents correct ?
056416  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
056422  001404                        BEQ     110$                    ; branch if SCR is OK
056424                                ERRSOFT 1901,E1901,EGB          ; else print "SCR INCORRECT"
056424  104457                                                                 TRAP    C$ERSOFT
056426  003555                                                                 .WORD   1901
056430  056727                                                                 .WORD   E1901
056432  023416                                                                 .WORD   EGB
                                110$:   CKLOOP                          ; branch to BGNSEG if loop on error set
056434  104406                                                                 TRAP    C$CLP1

056436  012737  010300  003020        MOV     #ACS!DBE!EOC,GOOD       ; get expected ACS contents
056444  012777  010000  124162        MOV     #ACS,@DRXDBR            ; address the ACS register
056452  017737  124156  003022        MOV     @DRXDBR,BAD             ; and get the actual contents
056460  013700  003022                MOV     BAD,R0                  ; get the contents into R0
056464  042700  171777                BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
056470  050037  003020                BIS     R0,GOOD                 ; set them in GOOD if they are set
056474  023737  003022  003020        CMP     BAD,GOOD                ; contents correct ?
056502  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
056506  001404                        BEQ     120$                    ; branch if ACS is OK
056510                                ERRSOFT 1902,E1902,EGB          ; else print "ACS INCORRECT"
056510  104457                                                                 TRAP    C$ERSOFT
056512  003556                                                                 .WORD   1902
056514  057012                                                                 .WORD   E1902
056516  023416                                                                 .WORD   EGB
                                120$:   CKLOOP                          ; branch to BGNSEG if loop on error set
056520  104406                                                                 TRAP    C$CLP1
```

TEST 19 - Data Buffer Empty Test.

```
 6876
 6877 056522  012737  020102  003020          MOV     #CTA!66.,GOOD           ; get expected CTA contents
 6878 056530  012777  020000  124076          MOV     #CTA,@DRXDBR            ; address the CTA
 6879 056536  017737  124072  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 6880 056544  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 6881 056552  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 6882 056556  001404                          BEQ     130$                    ; branch if CTA is OK
 6883 056560                                  ERRSOFT 1903,E1903,EGB          ; else print "CTA INCORRECT"
      056560  104457                                                                  TRAP    C$ERSOFT
      056562  003557                                                                  .WORD   1903
      056564  057076                                                                  .WORD   E1903
      056566  023416                                                                  .WORD   EGB
 6884 056570                          130$:   ENDSEG
      056570                                                                  10000$:
      056570  104405                                                                  TRAP    C$ESEG
 6885
 6886 056572  005737  002234                  TST     QVMODE                  ; is quick verify mode selected ?
 6887 056576  001010                          BNE     140$                    ; if yes, exit test
 6888 056600  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
 6889 056604  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
 6890 056612  001402                          BEQ     140$                    ; if yes, exit test
 6891 056614  000137  055700                  JMP     20$                     ; else, do another iteration
 6892
 6893 056620                          140$:   CLRVEC  DRXVEC                  ; restore the DRS trap catcher
      056620  013700  002636                                                          MOV     DRXVEC,R0
      056624  104436                                                                  TRAP    C$CVEC
 6894 056626                          150$:   EXIT TST
      056626  104432                                                                  TRAP    C$EXIT
      056630  000332                                                                  .WORD   L10054-.
 6895
 6896                                         .NLIST  BEX
 6897 056632     045     123     062  TSHD19::.ASCIZ   /%S2%ADATA BUFFER EMPTY TEST%N/
 6898 056670     116     117     040  E1900:  .ASCIZ   /NO DATA BUFFER EMPTY INTERRUPT/
 6899 056727     104     122     130  E1901:  .ASCIZ   /DRX11-C SCR INCORRECT AFTER DATA BUFFER EMPTY TEST/
 6900 057012     101     103     123  E1902:  .ASCIZ   /ACS REGISTER INCORRECT AFTER DATA BUFFER EMPTY TEST/
 6901 057076     103     124     101  E1903:  .ASCIZ   /CTA REGISTER INCORRECT AFTER DATA BUFFER EMPTY TEST/
 6902                                         .LIST   BEX
 6903                                         .EVEN
 6904
 6905 057162                                  ENDTST
      057162                                                                  L10054:
      057162  104401                                                                  TRAP    C$ETST
 6906
```

```
 6908                                   .SBTTL  TEST 20 - External Clock Test.
 6909
 6910                                   ;++
 6911                                   ; This checks that the RATE CLOCK IN/OUT line can be used to
 6912                                   ; trigger conversions. The control table and values for output are
 6913                                   ; set up as follows :
 6914                                   ;
 6915                                   ; Control word    Mode    Channel    Output voltage
 6916                                   ;
 6917                                   ;       0          0        0           1.0
 6918                                   ;       1          0        1           2.0
 6919                                   ;       2          1        2           3.0
 6920                                   ;
 6921                                   ; The external clock enable and maintenance bits are set and a 3
 6922                                   ; word DMA is enabled. The diagnostic then waits for 1 millisecond
 6923                                   ; for the data to be loaded into the FIFO, after which COUT is set
 6924                                   ; and cleared 3 times. This signal is looped back to the RATE
 6925                                   ; CLOCK IN/OUT line via the test connector. After the third pulse
 6926                                   ; a check is made that an End of conversion interrupt occurred and
 6927                                   ; that the SCR and ACS registers are correct.
 6928                                   ;--
 6929
 6930 057164                                   BGNTST
      057164                                                                   T20::
 6931 057164  005037  003030                   CLR     STAFLG                 ; flag test has been run since start
 6932 057170                                   RFLAGS  R0                     ; read operator flags
      057170  104421                                                           TRAP    C$RFLA
 6933 057172  032700  001000                   BIT     #PNT,R0                ; print test headers ?
 6934 057176  001410                           BEQ     10$                    ; if not, branch
 6935 057200                                   PRINTF  #TSHD20                ; else, print test header
      057200  012746  057756                                                   MOV     #TSHD20,-(SP)
      057204  012746  000001                                                   MOV     #1,-(SP)
      057210  010600                                                           MOV     SP,R0
      057212  104417                                                           TRAP    C$PNTF
      057214  062706  000004                                                   ADD     #4,SP
 6936
 6937 057220  005037  003032          10$:     CLR     ITRCNT                 ; clear iteration counter
 6938 057224  052777  000040  123374           BIS     #RES,@DRXSCR           ; reset the DRX11-C
 6939 057232  004737  026106                   JSR     PC,WT500               ; wait for 1 millisecond
 6940 057236  004737  026106                   JSR     PC,WT500               ; (2*500 microseconds)
 6941
 6942 057242                                   SETVEC  DRXVEC,#INT,#PRI07     ; set up DRX11-C vector
      057242  012746  000340                                                   MOV     #PRI07,-(SP)
      057246  012746  032216                                                   MOV     #INT,-(SP)
      057252  013746  002636                                                   MOV     DRXVEC,-(SP)
      057256  012746  000003                                                   MOV     #3,-(SP)
      057262  104437                                                           TRAP    C$SVEC
      057264  062706  000010                                                   ADD     #10,SP
 6943
 6944 057270  004737  030320                   JSR     PC,SETTAB              ; set up control table and data
 6945 057274  000003                           3                              ; for 3 control words
 6946 057276  057734                           T20TAB                         ; using table at T20TAB
 6947
 6948 057300  013701  002230                   MOV     CTIME,R1               ; get the selected conversion time
 6949 057304  052701  040000                   BIS     #PCR,R1                ; address the PCR
 6950 057310  010177  123320                   MOV     R1,@DRXDBR             ; and enter the conversion time
 6951 057314  052777  000001  123304           BIS     #FNCT0,@DRXSCR         ; transfer to PCR
```

TEST 20 - External Clock Test.

```
 6952
 6953 057322                          20$:    BGNSEG
      057322  104404                                                          TRAP    C$BSEG
 6954 057324  012777  020000  123302          MOV     #CTA,@DRXDBR            ; reset the CTA register
 6955 057332  052777  000001  123266          BIS     #FNCTO,@DRXSCR          ; back to control word 0
 6956 057340                                  SETPRI  #PRIOO                  ; drop the priority
      057340  012700  000000                                                  MOV     #PRIOO,RO
      057344  104441                                                          TRAP    C$SPRI
 6957 057346  005037  002772                  CLR     INTFLG                  ; clear the interrupt flag
 6958 057352  012777  011005  123254          MOV     #ACS!ECE!GO!MNT,@DRXDBR ; set up to start the conversions
 6959 057360  012777  000001  123240          MOV     #FNCTO,@DRXSCR          ; enable conversions and clear the SCR
 6960 057366  012777  000000  123234          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
 6961 057374  012777  003040  123230          MOV     #BUFOUT,@DRXADR         ; set up DMA address
 6962 057402  012777  000400  123220          MOV     #WCR1,@DRXCOR           ; now select WCR1
 6963 057410  012777  177775  123214          MOV     #-3,@DRXADR             ; to DMA 3 words
 6964 057416  012777  020000  123204          MOV     #RUN,@DRXCOR            ; start the DMA
 6965 057424  052777  000100  123174          BIS     #IE,@DRXSCR             ; enable interrupts
 6966
 6967 057432  004737  026106                  JSR     PC,WT500                : wait 1 millisecond for FIFO to load
 6968 057436  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)
 6969
 6970 057442  012701  000003                  MOV     #3,R1                   ; set and clear COUT 3 times
 6971 057446  052777  010400  123160  30$:    BIS     #ACS!COUT,@DRXDBR       ; set COUT
 6972 057454  052777  000001  123144          BIS     #FNCTO,@DRXSCR          ; in ACS register
 6973 057462  042777  000400  123144          BIC     #COUT,@DRXDBR           ; clear COUT
 6974 057470  052777  000001  123130          BIS     #FNCTO,@DRXSCR          ; in ACS register
 6975 057476  005301                          DEC     R1                      ; done 3 times ?
 6976 057500  001362                          BNE     30$                     ; if not, go back
 6977
 6978 057502  012777  010405  123124          MOV     #ACS!COUT!ECE!GO,@DRXDBR; clear MNT and set COUT
 6979 057510  052777  000001  123110          BIS     #FNCTO,@DRXSCR          ; to complete the conversion
 6980
 6981 057516  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 6982 057522  005002                          CLR     R2                      ; high byte is zero
 6983 057524  004737  030072                  JSR     PC,DIV                  ; divide by
 6984 057530  000175                          125.                            ; 125
 6985
 6986 057532  005701                  40$:    TST     R1                      ; loop until R1/R2 is zero
 6987 057534  001404                          BEQ     60$                     ;
 6988 057536  005301                  50$:    DEC     R1                      ;
 6989 057540  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 6990 057544  000772                          BR      40$                     ;
 6991 057546  005702                  60$:    TST     R2                      ;
 6992 057550  001402                          BEQ     70$                     ; total timeout = number of conversions
 6993 057552  005302                          DEC     R2                      ;                 * conversion time * 2
 6994 057554  000770                          BR      50$                     ;
 6995
 6996 057556                          70$:    SETPRI  #PRIO7                  ; stop any interrupts
      057556  012700  000340                                                  MOV     #PRIO7,RO
      057562  104441                                                          TRAP    C$SPRI
 6997 057564  042777  000100  123034          BIC     #IE,@DRXSCR             ; from the DRX11-C
 6998 057572  005737  002772                  TST     INTFLG                  ; did we get an interrupt ?
 6999 057576  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 7000 057602  001004                          BNE     80$                     ; branch if we got an interrupt
 7001 057604                                  ERRSOFT 2000,E2000,ECHK         ; else print "NO INTERRUPT"
      057604  104457                                                          TRAP    C$ERSOFT
      057606  003720                                                          .WORD   2000
```

TEST 20 - External Clock Test.

```
        057610  060011                                                          .WORD   E2000
        057612  023404                                                          .WORD   FCHK
  7002  057614                          80$:    CKLOOP                        ; branch to BGNSEG if loop on error set
        057614  104406                                                          TRAP    C$CLP1
  7003
  7004  057616  012737  010404  003020          MOV     #ACS!COUT!ECE,GOOD    ; get expected ACS contents
  7005  057624  012777  010000  123002          MOV     #ACS,@DRXDBR          ; address the ACS register
  7006  057632  017737  122776  003022          MOV     @DRXDBR,BAD           ; and get the actual contents
  7007  057640  013700  003022                  MOV     BAD,R0                ; get the contents into R0
  7008  057644  042700  171777                  BIC     #^C<UNI!CMP>,R0       ; and isolate uncertain bits
  7009  057650  050037  003020                  BIS     R0,GOOD               ; set them in GOOD if they are set
  7010  057654  023737  003022  003020          CMP     BAD,GOOD              ; contents correct ?
  7011  057662  004737  026306                  CALL    INSERT                ; skip branch if error insert selected
  7012  057666  001404                          BEQ     90$                   ; branch if ACS is OK
  7013  057670                                  ERRSOFT 2001,E2001,EGB        ; else print "ACS INCORRECT"
        057670  104457                                                          TRAP    C$ERSOFT
        057672  003721                                                          .WORD   2001
        057674  060060                                                          .WORD   E2001
        057676  023416                                                          .WORD   EGB
  7014  057700                          90$:    ENDSEG
        057700                                                          10000$:
        057700  104405                                                          TRAP    C$ESEG
  7015
  7016  057702  005737  002234                  TST     QVMODE                ; is quick verify mode selected ?
  7017  057706  001010                          BNE     100$                  ; if yes, exit test
  7018  057710  005237  003032                  INC     ITRCNT                ; else, increment iteration counter
  7019  057714  023727  003032  000010          CMP     ITRCNT,#10            ; iterations completed ?
  7020  057722  001402                          BEQ     100$                  ; if yes, branch
  7021  057724  000137  057322                  JMP     20$                   ; else do another iteration
  7022
  7023  057730                          100$:   EXIT TST
        057730  104432                                                          TRAP    C$EXIT
        057732  000210                                                          .WORD   L10055-.
  7024
  7025                                  ; Outputs :   Mode    Channel  Voltage
  7026                                  ;                                (mV)
  7027
  7028  057734  000000  000000  001750  T20TAB: .WORD   0,      0,       1000.  ; table for outputs
  7029  057742  000000  000001  003720          .WORD   0,      1,       2000.
  7030  057750  000001  000002  005670          .WORD   1,      2,       3000.
  7031
  7032                                          .NLIST  BEX
  7033  057756     045     123     062  TSHD20::.ASCIZ  /%S2%AEXTERNAL CLOCK TEST%N/
  7034  060011     116     117     040  E2000:  .ASCIZ  /NO INTERRUPT AFTER EXTERNAL CLOCK TEST/
  7035  060060     101     103     123  E2001:  .ASCIZ  /ACS REGISTER INCORRECT AFTER EXTERNAL CLOCK TEST/
  7036                                          .LIST   BEX
  7037                                          .EVEN
  7038
  7039  060142                                  ENDTST
        060142                                                          L10055:
        060142  104401                                                          TRAP    C$ETST
  7040
```

```
                                .SBTTL  TEST 21 - Linearity Test.

                                ;**
                                ; This checks the linearity of the AAF01-A. Each possible value is
                                ; output to channel 15. For each of these outputs, two outputs are
                                ; made on channel 0, the first using a value N bits greater than
                                ; that on channel 15, and the second, N bits lower. N is equal
                                ; to the value entered as "TOLERANCE" in the startup software
                                ; questions. A check is made that the first output on channel 0
                                ; results in a logical 0 from the comparator logic, and that the
                                ; second results in a 1. All outputs are made by program using the
                                ; DBF register. At the range boundaries, only one comparison with
                                ; channel 0 is made.
                                ;--

060144                                  BGNTST
060144                                                                  T21::
060144  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
060150                                  RFLAGS  R0                      ; read operator flags
060150  104421                                                                  TRAP    C$RFLA
060152  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
060156  001410                          BEQ     10$                     ; if not, branch
060160                                  PRINTF  #TSHD21                 ; else, print test header
060160  012746  060722                                                          MOV     #TSHD21,-(SP)
060164  012746  000001                                                          MOV     #1,-(SP)
060170  010600                                                                  MOV     SP,R0
060172  104417                                                                  TRAP    C$PNTF
060174  062706  000004                                                          ADD     #4,SP

060200  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter
060204  052777  000040  122414          BIS     #RES,@DRXSCR            ; reset the DRX11-C
060212  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
060216  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)

060222  012701  007777          20$:    MOV     #DBF!7777,R1            ; initialise output pattern
060226  012777  020000  122400          MOV     #CTA,@DRXDBR            ; address control word 0
060234  052777  000001  122364          BIS     #FNCT0,@DRXSCR          ; in CTA register
060242  012777  030017  122364          MOV     #CWR!15.,@DRXDBR        ; set up to output on channel 15
060250  052777  000001  122350          BIS     #FNCT0,@DRXSCR          ; transfer to CWR
060256  010177  122352                  MOV     R1,@DRXDBR              ; output the data
060262  052777  000001  122336          BIS     #FNCT0,@DRXSCR          ; to the DBF register
060270  012777  030000  122336          MOV     #CWR,@DRXDBR            ; set up to output to channel 0
060276  052777  000001  122322          BIS     #FNCT0,@DRXSCR          ; output slightly diferent data
060304  012777  007770  122322          MOV     #7770,@DRXDBR           ; to stop the comparator oscillating
060312  052777  000001  122306          BIS     #FNCT0,@DRXSCR          ; too much

060320  012702  000012                  MOV     #12,R2                  ; wait 5 milliseconds
060324  004737  026106          30$:    JSR     PC,WT500                ; for the comparator to stop
060330  005302                          DEC     R2                      ; oscillating
060332  001374                          BNE     30$                     ;

060334                          40$:    BGNSEG
060334  104404                                                                  TRAP    C$BSEG
060336  012777  020000  122270          MOV     #CTA,@DRXDBR            ; address control word 0
060344  052777  000001  122254          BIS     #FNCT0,@DRXSCR          ; in CTA register
060352  012777  030017  122254          MOV     #CWR!15.,@DRXDBR        ; set up to output on channel 15
060360  052777  000001  122240          BIS     #FNCT0,@DRXSCR          ; transfer to CWR
```

TEST 21 - Linearity Test.

```
 7091 060366  010177  122242                MOV     R1,@DRXDBR              ; output the data
 7092 060372  052777  000001  122226        BIS     #FNCT0,@DRXSCR          ; to the DBF register
 7093
 7094 060400  010102                        MOV     R1,R2                   ; get the channel 15 data
 7095 060402  063702  002232                ADD     TOL,R2                  ; add tolerance
 7096 060406  032702  010000                BIT     #10000,R2               ; too high for testing ?
 7097 060412  001046                        BNE     50$                     ; if yes, skip to next section
 7098 060414  012777  030000  122212        MOV     #CWR,@DRXDBR            ; else, set up to output on channel 0
 7099 060422  052777  000001  122176        BIS     #FNCT0,@DRXSCR          ; transfer to CWR
 7100 060430  010277  122200                MOV     R2,@DRXDBR              ; output the data
 7101 060434  052777  000001  122164        BIS     #FNCT0,@DRXSCR          ; to the DBF register
 7102 060442  012737  010000  003020        MOV     #ACS,GOOD               ; get expected ACS contents
 7103 060450  004737  026106                JSR     PC,WT500                ; wait 500 microseconds
 7104 060454  012777  010000  122152        MOV     #ACS,@DRXDBR            ; address the ACS register
 7105 060462  017737  122146  003022        MOV     @DRXDBR,BAD             ; and get the actual contents
 7106 060470  013700  003022                MOV     BAD,R0                  ; get the contents into R0
 7107 060474  042700  173777                BIC     #^CUNI,R0               ; and isolate uncertain bits
 7108 060500  050037  003020                BIS     R0,GOOD                 ; set them in GOOD if they are set
 7109 060504  023737  003022  003020        CMP     BAD,GOOD                ; contents correct ?
 7110 060512  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 7111 060516  001404                        BEQ     50$                     ; branch if ACS is OK
 7112 060520                                ERRSOFT 2100,E2100,EGBCC        ; else print "ACS INCORRECT"
      060520  104457                                                            TRAP    C$ERSOFT
      060522  004064                                                            .WORD   2100
      060524  060750                                                            .WORD   E2100
      060526  023622                                                            .WORD   EGBCC
 7113 060530                        50$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      060530  104406                                                            TRAP    C$CLP1
 7114
 7115 060532  010102                        MOV     R1,R2                   ; get the channel 15 data
 7116 060534  163702  002232                SUB     TOL,R2                  ; subtract tolerance
 7117 060540  103446                        BCS     60$                     ; branch if too low
 7118 060542  012777  030000  122064        MOV     #CWR,@DRXDBR            ; else, set up to output on channel 0
 7119 060550  052777  000001  122050        BIS     #FNCT0,@DRXSCR          ; transfer to CWR
 7120 060556  010277  122052                MOV     R2,@DRXDBR              ; output the data
 7121 060562  052777  000001  122036        BIS     #FNCT0,@DRXSCR          ; to the DBF register
 7122 060570  012737  012000  003020        MOV     #ACS!CMP,GOOD           ; get expected ACS contents
 7123 060576  004737  026106                JSR     PC,WT500                ; wait 500 microseconds
 7124 060602  012777  010000  122024        MOV     #ACS,@DRXDBR            ; address the ACS register
 7125 060610  017737  122020  003022        MOV     @DRXDBR,BAD             ; and get the actual contents
 7126 060616  013700  003022                MOV     BAD,R0                  ; get the contents into R0
 7127 060622  042700  173777                BIC     #^CUNI,R0               ; and isolate uncertain bits
 7128 060626  050037  003020                BIS     R0,GOOD                 ; set them in GOOD if they are set
 7129 060632  023737  003022  003020        CMP     BAD,GOOD                ; contents correct ?
 7130 060640  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 7131 060644  001404                        BEQ     60$                     ; branch if ACS is OK
 7132 060646                                ERRSOFT 2100,E2100,EGBCC        ; else print "ACS INCORRECT"
      060646  104457                                                            TRAP    C$ERSOFT
      060650  004064                                                            .WORD   2100
      060652  060750                                                            .WORD   E2100
      060654  023622                                                            .WORD   EGBCC
 7133 060656                        60$:    ENDSEG
      060656                                                                10000$:
      060656  104405                                                            TRAP    C$ESEG
 7134
 7135 060660  005301                        DEC     R1                      ; next output value
 7136 060662  020127  000000                CMP     R1,#DBF                 ; all values done ?
```

TEST 21 - Linearity Test.

```
 7137 060666  002222                          BGE     40$                     ; if not, do the next
 7138
 7139 060670  005737  002234                  TST     QVMODE                  ; is quick verify mode selected ?
 7140 060674  001010                          BNE     70$                     ; if yes, exit test
 7141 060676  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
 7142 060702  023727  003032  000004          CMP     ITRCNT,#4               ; iterations completed ?
 7143 060710  001402                          BEQ     70$                     ; if yes, branch
 7144 060712  000137  060222                  JMP     20$                     ; else do another iteration
 7145
 7146 060716                          70$:    EXIT TST
      060716  104432                                                          TRAP    C$EXIT
      060720  000102                                                          .WORD   L10056-.
 7147
 7148                                         .NLIST  BEX
 7149 060722     045     123     062  TSHD21::.ASCIZ  /%S2%ALINEARITY TEST%N/
 7150 060750     101     103     123  E2100:  .ASCIZ  /ACS INCORRECT AFTER DBF OUTPUT COMPARISON/
 7151                                         .LIST   BEX
 7152                                         .EVEN
 7153
 7154 061022                                  ENDTST
      061022                                                      L10056:
      061022  104401                                                        TRAP    C$ETST
 7155
```

TEST 22 - DMA Output Comparison Test.

```
                                .SBTTL  TEST 22 - DMA Output Comparison Test.

                                ;++
                                ; This checks that conversions can be made under DMA control. The
                                ; control table is set up to perform 2 outputs, on channels 0 and
                                ; 15. An internal buffer is set up to contain the data 1777 and
                                ; 3777. These are selected so that the ACS CMP bit wil be set.
                                ; The DRX11-C is then set up for a 2 word DMA and conversions are
                                ; initiated by setting the GO bit of the ACS register. After the
                                ; DMA has completed, the ACS register CMP bit is checked to be
                                ; set. The test is then repeated but with the data words
                                ; interchanged so that the ACS CMP bit is not set.
                                ;--

061024                                  BGNTST
061024                                                                  T22::
061024  005037  003030                  CLR     STAFLG                  ; flag test has been run since start
061030                                  RFLAGS  R0                      ; read operator flags
061030  104421                                                                 TRAP    C$RFLA
061032  032700  001000                  BIT     #PNT,R0                 ; print test headers ?
061036  001410                          BEQ     10$                     ; if not, branch
061040                                  PRINTF  #TSHD22                 ; else, print test header
061040  012746  061602                                                         MOV     #TSHD22,-(SP)
061044  012746  000001                                                         MOV     #1,-(SP)
061050  010600                                                                 MOV     SP,R0
061052  104417                                                                 TRAP    C$PNTF
061054  062706  000004                                                         ADD     #4,SP

061060  005037  003032          10$:    CLR     ITRCNT                  ; clear iteration counter
061064  052777  000040  121534          BIS     #RES,@DRXSCR            ; reset the DRX11-C
061072  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
061076  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)

061102  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
061106  052701  040000                  BIS     #PCR,R1                 ; address the PCR
061112  010177  121516                  MOV     R1,@DRXDBR              ; and enter the conversion time
061116  052777  000001  121502          BIS     #FNCT0,@DRXSCR          ; transfer to PCR

061124  012737  001777  003040  20$:    MOV     #1777,BUFOUT            ; set up data for DMA
061132  012737  003777  003042          MOV     #3777,BUFOUT+2          ; 1777 and 3777
061140  012703  012000                  MOV     #ACS!CMP,R3             ; expected ACS contents

061144                          30$:    BGNSEG
061144  104404                                                                 TRAP    C$BSEG
061146  012777  020000  121460          MOV     #CTA,@DRXDBR            ; address the CTA to CWR 0
061154  052777  000001  121444          BIS     #FNCT0,@DRXSCR          ; and transfer to the CTA
061162  012777  030000  121444          MOV     #CWR,@DRXDBR            ; set up for channel 0
061170  052777  000001  121430          BIS     #FNCT0,@DRXSCR          ; and transfer to CWR 0
061176  012777  020001  121430          MOV     #CTA!1,@DRXDBR          ; address the CTA to CWR 1
061204  052777  000001  121414          BIS     #FNCT0,@DRXSCR          ; and transfer to the CTA
061212  012777  030017  121414          MOV     #CWR!15.,@DRXDBR        ; set up for channel 15
061220  052777  000001  121400          BIS     #FNCT0,@DRXSCR          ; and transfer to CWR 1
061226  012777  020000  121400          MOV     #CTA,@DRXDBR            ; address the CTA to CWR 0
061234  052777  000001  121364          BIS     #FNCT0,@DRXSCR          ; and transfer to the CTA

061242  012777  010001  121364          MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
061250  012777  000001  121350          MOV     #FNCT0,@DRXSCR          ; enable conversions and clear the SCR
```

TEST 22 - DMA Output Comparison Test.

```
 7206 061256  012777  000000  121344          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
 7207 061264  012777  003040  121340          MOV     #BUFOUT,@DRXADR         ; set up DMA address
 7208 061272  012777  000400  121330          MOV     #WCR1,@DRXCOR           ; now select WCR1
 7209 061300  012777  177776  121324          MOV     #-2,@DRXADR             ; to DMA 2 words
 7210 061306  012777  020000  121314          MOV     #RUN,@DRXCOR            ; start the DMA
 7211
 7212 061314  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 7213 061320  005002                          CLR     R2                      ; high byte is zero
 7214 061322  004737  027776                  JSR     PC,MUL                  ; multiply by
 7215 061326  000002                          2                               ; the number of conversions
 7216 061330  004737  030072                  JSR     PC,DIV                  ; divide by
 7217 061334  000175                          125.                            ; 125
 7218
 7219 061336  005701                  40$:    TST     R1                      ; loop until R1/R2 is zero
 7220 061340  001404                          BEQ     60$                     ;
 7221 061342  005301                  50$:    DEC     R1                      ;
 7222 061344  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 7223 061350  000772                          BR      40$                     ;
 7224 061352  005702                  60$:    TST     R2                      ;
 7225 061354  001402                          BEQ     70$                     ; total timeout = number of conversions
 7226 061356  005302                          DEC     R2                      ;                 * conversion time * 2
 7227 061360  000770                          BR      50$                     ;
 7228
 7229 061362  012737  100600  003020  70$:    MOV     #ATT!STATO!RDY,GOOD     ; get expected SCR contents
 7230 061370  017737  121232  003022          MOV     @DRXSCR,BAD             ; and actual contents
 7231 061376  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
 7232 061404  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 7233 061410  001404                          BEQ     80$                     ; branch if SCR is OK
 7234 061412                                  ERRSOFT 2200,E2200,EGB          ; else print "SCR INCORRECT"
      061412  104457                                                                  TRAP    C$ERSOFT
      061414  004230                                                                  .WORD   2200
      061416  061644                                                                  .WORD   E2200
      061420  023416                                                                  .WORD   EGB
 7235 061422                          80$:    CKLOOP                          ; branch to BGNSEG if loop on error set
      061422  104406                                                                  TRAP    C$CLP1
 7236
 7237 061424  010337  003020                  MOV     R3,GOOD                 ; get expected ACS contents
 7238 061430  012777  010000  121176          MOV     #ACS,@DRXDBR            ; address the ACS register
 7239 061436  017737  121172  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
 7240 061444  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
 7241 061450  042700  173777                  BIC     #^CUNI,R0               ; and isolate uncertain bits
 7242 061454  050037  003020                  BIS     R0,GOOD                 ; set them in GOOD if they are set
 7243 061460  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
 7244 061466  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 7245 061472  001410                          BEQ     90$                     ; branch if ACS is OK
 7246 061474  013701  003040                  MOV     BUFOUT,R1               ; channel 0 data
 7247 061500  013702  003042                  MOV     BUFOUT+2,R2             ; channel 15 data
 7248 061504                                  ERRSOFT 2201,E2201,EGBCC        ; else print "ACS INCORRECT"
      061504  104457                                                                  TRAP    C$ERSOFT
      061506  004231                                                                  .WORD   2201
      061510  061726                                                                  .WORD   E2201
      061512  023622                                                                  .WORD   EGBCC
 7249 061514                          90$:    ENDSEG
      061514                                                                  10000$:
      061514  104405                                                                  TRAP    C$ESEG
 7250
 7251 061516  020327  010000                  CMP     R3,#ACS                 ; CMP set test done ?
```

TEST 22 - DMA Output Comparison Test.

```
   7252 061522  001412                          BEQ     100$                    ; if yes, branch
   7253 061524  012737  003777  003040          MOV     #3777,BUFOUT            ; else, set up data for DMA
   7254 061532  012737  001777  003042          MOV     #1777,BUFOUT+2          ; 3777 and 1777
   7255 061540  012703  010000                  MOV     #ACS,R3                 ; flag that CMP should be clear
   7256 061544  000137  061144                  JMP     30$                     ; and do CMP clear test
   7257
   7258 061550  005737  002234          100$:   TST     QVMODE                  ; is quick verify mode selected ?
   7259 061554  001010                          BNE     110$                    ; if yes, exit test
   7260 061556  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
   7261 061562  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
   7262 061570  001402                          BEQ     110$                    ; if yes, exit test
   7263 061572  000137  061124                  JMP     20$                     ; else, do another iteration
   7264
   7265 061576                          110$:   EXIT TST
        061576  104432                                                          TRAP    C$EXIT
        061600  000200                                                          .WORD   L10057-.
   7266
   7267                                         .NLIST  BEX
   7268 061602     045     123     062  TSHD22::.ASCIZ  /%S2%ADMA OUTPUT COMPARISON TEST%N/
   7269 061644     104     122     130  E2200:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER DMA OUTPUT COMPARISON/
   7270 061726     101     103     123  E2201:  .ASCIZ  /ACS INCORRECT AFTER DMA OUTPUT COMPARISON/
   7271                                         .LIST   BEX
   7272                                         .EVEN
   7273
   7274 062000                                  ENDTST
        062000                                                          L10057:
        062000  104401                                                          TRAP    C$ETST
   7275
```

TEST 23 - Memory Transfer Conversion test.

```
                                .SBTTL  TEST 23 - Memory Transfer Conversion test.

                                ;++
                                ; This tests DMA conversions in memory transfer mode. An internal
                                ; buffer is set up to contain the following control and data
                                ; words :
                                ;
                                ; Word    Contents    Description
                                ;
                                ;   0       0000      Control word, mode 0, channel 0
                                ;   1       0000      Data, 9.9976V unipolar or 9.9951V bipolar
                                ;   2       0001      Control word, mode 0, channel 1
                                ;   3       1777      Data,  7.5000V unipolar or 5.0000V bipolar
                                ;   4       0201      Control word, mode 2, channel 1
                                ;   5       0201      Control word, mode 2, channel 1
                                ;   6       0002      Control word, mode 0, channel 2
                                ;   7       3777      Data,  5.0000V unipolar or 0.0000V bipolar
                                ;   8       0003      Control word, mode 0, channel 3
                                ;   9       5777      Data,  2.5000V unipolar or -5.0000V bipolar
                                ;  10       0303      Control word, mode 3, channel 3
                                ;  11       0004      Control word, mode 0, channel 4
                                ;  12       7776      Data,  0.0024V unipolar or -9.9951V bipolar
                                ;  13       0404      Control word, mode 4, channel 4
                                ;  14       0005      Control word, mode 0, channel 5
                                ;  15       7777      Data,  0.0000V unipolar or -10.0000V bipolar
                                ;  16       0605      Control word, mode 6, channel 5
                                ;  17       0017      Control word, mode 0, channel 15
                                ;  18       3777      Data,  5.0000V unipolar or 0.0000V bipolar
                                ;  19       0717      Control word, mode 7, channel 15
                                ;
                                ; These are selected to test combinations of CWR modes and
                                ; channels so that at the end of the conversion the ACS CMP bit
                                ; will be set.
                                ;
                                ; The DRX11-C is set up to start a 20 word DMA. The GO and MET
                                ; bits of the ACS register are then set. After the DMA has
                                ; completed, the ACS register MET and CMP bits are checked to be
                                ; set.
                                ;
                                ; The test is then repeated but with the data words for channels 0
                                ; and 15 complemented so that after the conversions the ACS CMP
                                ; bit is not set.
                                ;--
```

TEST 23 - Memory Transfer Conversion test.

```
                                        BGNTST
                                                                        T23::
                        CLR     STAFLG          ; flag test has been run since start
                        RFLAGS  R0              ; read operator flags
                                                        TRAP    C$RFLA
                        BIT     #PNT,R0         ; print test headers ?
                        BEQ     10$             ; if not, branch
                        PRINTF  #TSHD23         ; else, print test header
                                                        MOV     #TSHD23,-(SP)
                                                        MOV     #1,-(SP)
                                                        MOV     SP,R0
                                                        TRAP    C$PNTF
                                                        ADD     #4,SP

                10$:    CLR     ITRCNT          ; clear iteration counter
                        BIS     #RES,@DRXSCR    ; reset the DRX11-C
                        JSR     PC,WT500        ; wait for 1 millisecond
                        JSR     PC,WT500        ; (2*500 microseconds)

                        MOV     CTIME,R1        ; get the selected conversion time
                        BIS     #PCR,R1         ; address the PCR
                        MOV     R1,@DRXDBR      ; and enter the conversion time
                        BIS     #FNCT0,@DRXSCR  ; transfer to PCR

                        MOV     #T23TA1,R3      ; set up address of first data buffer
                        MOV     #ACS!CMP!MET,R4 ; and expected ACS contents

                20$:    BGNSEG
                                                        TRAP    C$BSEG
                        MOV     #ACS!MET!GO,@DRXDBR     ; set up to start the conversions
                        MOV     #FNCT0,@DRXSCR  ; enable conversions and clear the SCR
                        MOV     #BAR1,@DRXCOR   ; multiplex to BAR1
                        MOV     R3,@DRXADR      ; set up DMA address
                        MOV     #WCR1,@DRXCOR   ; now select WCR1
                        MOV     #-20.,@DRXADR   ; to DMA 20 words
                        MOV     #RUN,@DRXCOR    ; start the DMA

                        MOV     CTIME,R1        ; get the conversion time
                        CLR     R2              ; high byte is zero
                        JSR     PC,MUL          ; multiply by
                        20.                     ; the number of conversions
                        JSR     PC,DIV          ; divide by
                        125.                    ; 125

                30$:    TST     R1              ; loop until R1/R2 is zero
                        BEQ     50$             ;
                40$:    DEC     R1              ;
                        JSR     PC,WT25         ; wait 25 microseconds
                        BR      30$             ;
                50$:    TST     R2              ;
                        BEQ     60$             ; total timeout = number of conversions
                        DEC     R2              ;                 * conversion time * 2
                        BR      40$             ;

                60$:    MOV     #ATT!STAT0!RDY,GOOD     ; get expected SCR contents
                        MOV     @DRXSCR,BAD     ; and actual contents
                        CMP     BAD,GOOD        ; are they the same ?
```

```
TEST 23 - Memory Transfer Conversion test.

   7370 062254  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
   7371 062260  001404                          BEQ     70$                     ; branch if SCR is OK
   7372 062262                                  ERRSOFT 2300,E2300,EGB          ; else print "SCR INCORRECT"
        062262  104457                                                          TRAP    C$ERSOFT
        062264  004374                                                          .WORD   2300
        062266  062613                                                          .WORD   E2300
        062270  023416                                                          .WORD   EGB
   7373 062272                          70$:    CKLOOP                          ; branch to BGNSEG if loop on error set
        062272  104406                                                          TRAP    C$CLP1
   7374
   7375 062274  010437  003020                  MOV     R4,GOOD                 ; get expected ACS contents
   7376 062300  012777  010000  120326          MOV     #ACS,@DRXDBR            ; address the ACS register
   7377 062306  017737  120322  003022          MOV     @DRXDBR,BAD             ; and get the actual contents
   7378 062314  013700  003022                  MOV     BAD,R0                  ; get the contents into R0
   7379 062320  042700  173777                  BIC     #^CUNI,R0               ; and isolate uncertain bit
   7380 062324  050037  003020                  BIS     R0,GOOD                 ; set it in GOOD if it is set
   7381 062330  023737  003022  003020          CMP     BAD,GOOD                ; contents correct ?
   7382 062336  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
   7383 062342  001404                          BEQ     80$                     ; branch if ACS is OK
   7384 062344                                  ERRSOFT 2301,E2301,EGB          ; else print "ACS INCORRECT"
        062344  104457                                                          TRAP    C$ERSOFT
        062346  004375                                                          .WORD   2301
        062350  062702                                                          .WORD   E2301
        062352  023416                                                          .WORD   EGB
   7385 062354                          80$:    ENDSEG
        062354                                                                  10000$:
        062354  104405                                                          TRAP    C$ESEG
   7386
   7387 062356  020327  062474                  CMP     R3,#T23TA2              ; second pass done ?
   7388 062362  001405                          BEQ     90$                     ; if yes, branch
   7389 062364  012703  062474                  MOV     #T23TA2,R3              ; else set up alternate table address
   7390 062370  012704  010002                  MOV     #ACS!MET,R4             ; new expected ACS contents
   7391 062374  000646                          BR      20$                     ; and do second pass
   7392
   7393 062376  005737  002234          90$:    TST     QVMODE                  ; is quick verify mode selected ?
   7394 062402  001006                          BNE     100$                    ; if yes, exit test
   7395 062404  005237  003032                  INC     ITRCNT                  ; else, increment iteration counter
   7396 062410  023727  003032  000010          CMP     ITRCNT,#10              ; iterations completed ?
   7397 062416  001235                          BNE     20$                     ; if not, do another iteration
   7398
   7399 062420                          100$:   EXIT TST
        062420  104432                                                                  TRAP    C$EXIT
        062422  000350                                                                  .WORD   L10060-.
   7400
   7401                                 ; Table for first half of test.
   7402
   7403 062424  000000                  T23TA1: .WORD   0000    ; Control word, mode 0, channel 0
   7404 062426  000000                          .WORD   0000    ; Data, 9.9976V unipolar or 9.9951V bipolar
   7405 062430  000001                          .WORD   0001    ; Control word, mode 0, channel 1
   7406 062432  001777                          .WORD   1777    ; Data,  7.5000V unipolar or 5.0000V bipolar
   7407 062434  000201                          .WORD   0201    ; Control word, mode 2, channel 1
   7408 062436  000201                          .WORD   0201    ; Control word, mode 2, channel 1
   7409 062440  000002                          .WORD   0002    ; Control word, mode 0, channel 2
   7410 062442  003777                          .WORD   3777    ; Data,  5.0000V unipolar or 0.0000V bipolar
   7411 062444  000003                          .WORD   0003    ; Control word, mode 0, channel 3
   7412 062446  005777                          .WORD   5777    ; Data,  2.5000V unipolar or -5.0000V bipolar
   7413 062450  000303                          .WORD   0303    ; Control word, mode 3, channel 3
```

TEST 23 - Memory Transfer Conversion test.

```
7414 062452  000004                          .WORD   0004    ; Control word, mode 0, channel 4
7415 062454  0C7776                          .WORD   7776    ; Data,  0.0024V unipolar or -9.9951V bipolar
7416 062456  000404                          .WORD   0404    ; Control word, mode 4, channel 4
7417 062460  000005                          .WORD   0005    ; Control word, mode 0, channel 5
7418 062462  007777                          .WORD   7777    ; Data,  0.0000V unipolar or -10.0000V bipolar
7419 062464  000605                          .WORD   0605    ; Control word, mode 6, channel 5
7420 062466  000017                          .WORD   0017    ; Control word, mode 0, channel 15
7421 062470  003777                          .WORD   3777    ; Data,  5.0000V unipolar or 0.0000V bipolar
7422 062472  000717                          .WORD   0717    ; Control word, mode 7, channel 15
7423
7424                                 ; Table for second half of test.
7425
7426 062474  000000                  T23TA2: .WORD   0000    ; Control word, mode 0, channel 0
7427 062476  003777                          .WORD   3777    ; Data,  5.0000V unipolar or 0.0000V bipolar
7428 062500  000001                          .WORD   0001    ; Control word, mode 0, channel 1
7429 062502  001777                          .WORD   1777    ; Data,  7.5000V unipolar or 5.0000V bipolar
7430 062504  000201                          .WORD   0201    ; Control word, mode 2, channel 1
7431 062506  000201                          .WORD   0201    ; Control word, mode 2, channel 1
7432 062510  000002                          .WORD   0002    ; Control word, mode 0, channel 2
7433 062512  003777                          .WORD   3777    ; Data,  5.0000V unipolar or 0.0000V bipolar
7434 062514  000003                          .WORD   0003    ; Control word, mode 0, channel 3
7435 062516  005777                          .WORD   5777    ; Data,  2.5000V unipolar or -5.0000V bipolar
7436 062520  000303                          .WORD   0303    ; Control word, mode 3, channel 3
7437 062522  000004                          .WORD   0004    ; Control word, mode 0, channel 4
7438 062524  007776                          .WORD   7776    ; Data,  0.0024V unipolar or -9.9951V bipolar
7439 062526  000404                          .WORD   0404    ; Control word, mode 4, channel 4
7440 062530  000005                          .WORD   0005    ; Control word, mode 0, channel 5
7441 062532  007777                          .WORD   7777    ; Data,  0.0000V unipolar or -10.0000V bipolar
7442 062534  000605                          .WORD   0605    ; Control word, mode 6, channel 5
7443 062536  000017                          .WORD   0017    ; Control word, mode 0, channel 15
7444 062540  000000                          .WORD   0000    ; Data, 9.9976V unipolar or 9.9951V bipolar
7445 062542  000717                          .WORD   0717    ; Control word, mode 7, channel 15
7446
7447                                         .NLIST  BEX
7448 062544      045     123     062  TSHD23::.ASCIZ  /%S2%AMEMORY TRANSFER CONVERSION TEST%N/
7449 062613      104     122     130  E2300:  .ASCIZ  /DRX11-C SCR INCORRECT AFTER MEMORY MODE DMA CONVERSION/
7450 062702      101     103     123  E2301:  .ASCIZ  /ACS REGISTER INCORRECT AFTER MEMORY MODE DMA CONVERSION/
7451                                         .LIST   BEX
7452                                         .EVEN
7453
7454 062772                                  ENDTST
     062772                                                                  L10060:
     062772  104401                                                                TRAP    C$ETST
7455
```

TEST 24 - Calibration Test.

```
                        .SBTTL  TEST 24 - Calibration Test.

                        ;++
                        ; This allows the AAF01-A analogue output to be calibrated. ACS
                        ; bit 11 is first read and a message printed indicating whether
                        ; the module is switched to unipolar or bipolar mode Eg. :
                        ;
                        ; CALIBRATING FOR UNIPOLAR MODE                          .
                        ;
                        ; In bipolar mode, each of the channels selected in the startup
                        ; questions is loaded with the value 7777 and the user is asked to
                        ; check that the DVM reads -10.000 volts +/- 4.88 millivolts on
                        ; the selected channels. If the output is not within this
                        ; tolerance, the user is asked to adjust the offset potentiometer
                        ; RV1 until the outputs are correct. Each channel is then loaded
                        ; with the value 0000 and the user asked to check that the DVM now
                        ; reads 9.9951 volts +/- 4.88 millivolts. If the output is not
                        ; within this tolerance, he is asked to adjust the gain
                        ; potentiometer RV2 until the output is correct.
                        ;
                        ; In unipolar mode, each of the channels selected in the startup
                        ; questions is loaded with the value 7777 and the user is asked to
                        ; check that the DVM reads 0.0000 volts +/- 4.88 millivolts on the
                        ; selected channels. If the output is not within this tolerance,
                        ; the user is asked to adjust the DAC OFFS potentiometer RV1 until
                        ; outputs are correct. Each channel is then loaded with the value
                        ; 0000 and the user asked to check that the DVM now reads 9.9976
                        ; volts +/- 4.88 millivolts. If the output is not within this
                        ; tolerance, he is asked to adjust the gain potentiometer RV2
                        ; until the output is correct.
                        ;
                        ; All outputs are made directly through the DBF register.
                        ;
                        ; After the module has been tested and/or calibrated, the
                        ; following message is output describing how to continue running
                        ; the diagnostic without the calibration test :
                        ;
                        ; THIS TEST CAN BE DISABLED BY ABORTING THE TEST AND SETTING THE
                        ; "UAM" FLAG. EG. "CONTROL C" THEN "CONTINUE/FLAGS:UAM".
                        ; TYPE "CARRIAGE RETURN TO CONTINUE OR "CONTROL C" TO ABORT (A) ?
                        ;--
```

```
062774                                BGNTST
062774                                                                  T24::
062774  005037  003030                CLR     STAFLG                ; flag test has been run since start
063000                                RFLAGS  R0                    ; read operator flags
063000  104421                                                                  TRAP    C$RFLA
063002  032700  001000                BIT     #PNT,R0               ; print test headers ?
063006  001425                        BEQ     20$                   ; if not, branch
063010                                MANUAL                        ; is manual intervention allowed ?
063010  104450                                                                  TRAP    C$MANI
063012                                BNCOMPLETE 10$                ; if not, branch
063012  103011                                                                  BCC     10$
063014                                PRINTF  #TSHD24               ; else, print test header
063014  012746  064036                                                          MOV     #TSHD24,-(SP)
063020  012746  000001                                                          MOV     #1,-(SP)
063024  010600                                                                  MOV     SP,R0
063026  104417                                                                  TRAP    C$PNTF
063030  062706  000004                                                          ADD     #4,SP
063034  000416                        BR      30$                   ; and start the test
063036                        10$:    PRINTF  #TSNS24               ; print "TEST DISABLED BY UAM"
063036  012746  064066                                                          MOV     #TSNS24,-(SP)
063042  012746  000001                                                          MOV     #1,-(SP)
063046  010600                                                                  MOV     SP,R0
063050  104417                                                                  TRAP    C$PNTF
063052  062706  000004                                                          ADD     #4,SP
063056                                EXIT TST                      ; and skip the test
063056  104432                                                                  TRAP    C$EXIT
063060  002300                                                                  .WORD   L10061-.
063062                        20$:    MANUAL                        ; manual intervention allowed ?
063062  104450                                                                  TRAP    C$MANI
063064                                BCOMPLETE 30$                 ; if yes, branch
063064  103402                                                                  BCS     30$
063066                                EXIT TST                      ; else skip the test
063066  104432                                                                  TRAP    C$EXIT
063070  002270                                                                  .WORD   L10061-.

063072  005037  003032        30$:    CLR     ITRCNT                ; clear iteration counter
063076  012705  000001                MOV     #1,R5                 ; initialize pass counter
063102  052777  000040  117516        BIS     #RES,@DRXSCR          ; reset the DRX11-C
063110  004737  026106                JSR     PC,WT500              ; wait for 1 millisecond
063114  004737  026106                JSR     PC,WT500              ; (2*500 microseconds)

063120  012701  007777        40$:    MOV     #DBF!7777,R1          ; initialize output pattern
063124  013702  002224                MOV     FSTCHN,R2             ; save first selected channel
063130  052702  030000                BIS     #CWR,R2               ; for CWR register in R2
063134  013703  002226                MOV     LSTCHN,R3             ; save last selected channel
063140  052703  030000                BIS     #CWR,R3               ; for CWR register in R3
063144  010204                        MOV     R2,R4                 ; start with first selected channel
063146  012777  020000  117460        MOV     #CTA,@DRXDBR          ; address control word 0
063154  052777  000001  117444        BIS     #FNCTO,@DRXSCR        ; in CTA register

063162  005737  003026                TST     MODE                  ; module in unipolar mode ?
063166  001531                        BEQ     100$                  ; if yes, branch
                              ;
                              ; Bipolar calibration
                              ;
063170                                PRINTF  #BICAL,R5             ; print "BIPOLAR MODE"
063170  010546                                                                  MOV     R5,-(SP)
```

TEST 24 - Calibration Test.

```
        063172  012746  064147                                                  MOV     #BICAL,-(SP)
        063176  012746  000002                                                  MOV     #2,-(SP)
        063202  010600                                                          MOV     SP,R0
        063204  104417                                                          TRAP    C$PNTF
        063206  062706  000006                                                  ADD     #6,SP
   7535
   7536 063212  010477  117416          50$:    MOV     R4,@DRXDBR              ; set up to output on selected channel
   7537 063216  052777  000001  117402          BIS     #FNCT0,@DRXSCR          ; transfer to CWR
   7538 063224  010177  117404          MOV     R1,@DRXDBR              ; output the data
   7539 063230  052777  000001  117370          BIS     #FNCT0,@DRXSCR          ; to the DBF register
   7540 063236  005204                  INC     R4                      ; next channel
   7541 063240  020403                  CMP     R4,R3                   ; all channels done ?
   7542 063242  101763                  BLOS    50$                     ; if not, do the next
   7543
   7544 063244                          PRINTF  #BI1                    ; print "CHECK OUTPUTS = -10V"
        063244  012746  064225                                                  MOV     #BI1,-(SP)
        063250  012746  000001                                                  MOV     #1,-(SP)
        063254  010600                                                          MOV     SP,R0
        063256  104417                                                          TRAP    C$PNTF
        063260  062706  000004                                                  ADD     #4,SP
   7545 063264                  60$:    GMANIL  BI2,ANSWER,377,YES      ; ask 'IN TOLERANCE ?"
        063264  104443                                                          TRAP    C$GMAN
        063266  000404                                                          BR      10000$
        063270  064034                                                          .WORD   ANSWER
        063272  000130                                                          .WORD   T$CODE
        063274  064324                                                          .WORD   BI2
        063276  000377                                                          .WORD   377
        063300                                                                  10000$:
   7546 063300  005737  064034          TST     ANSWER                  ; in tolerance ?
   7547 063304  001011                  BNE     70$                     ; if yes, branch
   7548 063306                          PRINTF  #BI3                    ; else print "ADJUST RV1"
        063306  012746  064401                                                  MOV     #BI3,-(SP)
        063312  012746  000001                                                  MOV     #1,-(SP)
        063316  010600                                                          MOV     SP,R0
        063320  104417                                                          TRAP    C$PNTF
        063322  062706  000004                                                  ADD     #4,SP
   7549 063326  000756                  BR      60$                     ; and ask again
   7550
   7551 063330  005001          70$:    CLR     R1                      ; clear output value
   7552 063332  010204                  MOV     R2,R4                   ; start with first selected channel
   7553 063334  010477  117274          80$:    MOV     R4,@DRXDBR              ; set up to output on selected channel
   7554 063340  052777  000001  117260          BIS     #FNCT0,@DRXSCR          ; transfer to CWR
   7555 063346  010177  117262          MOV     R1,@DRXDBR              ; output the data
   7556 063352  052777  000001  117246          BIS     #FNCT0,@DRXSCR          ; to the DBF register
   7557 063360  005204                  INC     R4                      ; next channel
   7558 063362  020403                  CMP     R4,R3                   ; all channels done ?
   7559 063364  101763                  BLOS    80$                     ; if not, do the next
   7560
   7561 063366                          PRINTF  #BI4                    ; print "CHECK OUTPUTS = 9.9951V"
        063366  012746  064503                                                  MOV     #BI4,-(SP)
        063372  012746  000001                                                  MOV     #1,-(SP)
        063376  010600                                                          MOV     SP,R0
        063400  104417                                                          TRAP    C$PNTF
        063402  062706  000004                                                  ADD     #4,SP
   7562 063406                  90$:    GMANIL  BI2,ANSWER,377,YES      ; ask "IN TOLERANCE ?"
        063406  104443                                                          TRAP    C$GMAN
        063410  000404                                                          BR      10001$
```

TEST 24 - Calibration Test.

```
        063412  064034                                                          .WORD   ANSWER
        063414  000130                                                          .WORD   T$CODE
        063416  064324                                                          .WORD   BI2
        063420  000377                                                          .WORD   377
        063422                                                          10001$:
 7563   063422  005737  064034          TST     ANSWER          ; in tolerance ?
 7564   063426  001142                  BNE     160$            ; if yes, calibration is finished
 7565   063430                          PRINTF  #BI5            ; else print "ADJUST RV2"
        063430  012746  064605                                                  MOV     #BI5,-(SP)
        063434  012746  000001                                                  MOV     #1,-(SP)
        063440  010600                                                          MOV     SP,R0
        063442  104417                                                          TRAP    C$PNTF
        063444  062706  000004                                                  ADD     #4,SP
 7566   063450  000756                  BR      90$             ; and ask again
 7567                           ;
 7568                           ; Unipolar calibration
 7569                           ;
 7570   063452                  100$:   PRINTF  #UNICAL,R5      ; print "UNIPOLAR MODE"
        063452  010546                                                          MOV     R5,-(SP)
        063454  012746  064705                                                  MOV     #UNICAL,-(SP)
        063460  012746  000002                                                  MOV     #2,-(SP)
        063464  010600                                                          MOV     SP,R0
        063466  104417                                                          TRAP    C$PNTF
        063470  062706  000006                                                  ADD     #6,SP
 7571
 7572   063474  010477  117134  110$:   MOV     R4,@DRXDBR      ; set up to output on selected channel
 7573   063500  052777  000001  117120  BIS     #FNCTO,@DRXSCR  ; transfer to CWR
 7574   063506  010177  117122          MOV     R1,@DRXDBR      ; output the data
 7575   063512  052777  000001  117106  BIS     #FNCTO,@DRXSCR  ; to the DBF register
 7576   063520  005204                  INC     R4              ; next channel
 7577   063522  020403                  CMP     R4,R3           ; all channels done ?
 7578   063524  101763                  BLOS    110$            ; if not, do the next
 7579
 7580   063526                          PRINTF  #BI6            ; print "CHECK OUTPUTS = 0V"
        063526  012746  064764                                                  MOV     #BI6,-(SP)
        063532  012746  000001                                                  MOV     #1,-(SP)
        063536  010600                                                          MOV     SP,R0
        063540  104417                                                          TRAP    C$PNTF
        063542  062706  000004                                                  ADD     #4,SP
 7581   063546                  120$:   GMANIL  BI2,ANSWER,377,YES      ; ask "IN TOLERANCE ?"
        063546  104443                                                          TRAP    C$GMAN
        063550  000404                                                          BR      10002$
        063552  064034                                                          .WORD   ANSWER
        063554  000130                                                          .WORD   T$CODE
        063556  064324                                                          .WORD   BI2
        063560  000377                                                          .WORD   377
        063562                                                          10002$:
 7582   063562  005737  064034          TST     ANSWER          ; in tolerance ?
 7583   063566  001011                  BNE     130$            ; if yes, branch
 7584   063570                          PRINTF  #BI3            ; else print "ADJUST RV1"
        063570  012746  064401                                                  MOV     #BI3,-(SP)
        063574  012746  000001                                                  MOV     #1,-(SP)
        063600  010600                                                          MOV     SP,R0
        063602  104417                                                          TRAP    C$PNTF
        063604  062706  000004                                                  ADD     #4,SP
 7585   063610  000756                  BR      120$            ; and ask again
 7586
```

TEST 24 - Calibration Test.

```
 7587 063612  005001                 130$:  CLR     R1                  ; reset the output value
 7588 063614  010204                        MOV     R2,R4               ; start with first selected channel
 7589 063616  010477  117012         140$:  MOV     R4,@DRXDBR          ; set up to output on selected channel
 7590 063622  052777  000001  116776        BIS     #FNCT0,@DRXSCR      ; transfer to CWR
 7591 063630  010177  117000                MOV     R1,@DRXDBR          ; output the data
 7592 063634  052777  000001  116764        BIS     #FNCT0,@DRXSCR      ; to the DBF register
 7593 063642  005204                        INC     R4                  ; next channel
 7594 063644  020403                        CMP     R4,R3               ; all channels done ?
 7595 063646  101763                        BLOS    140$                ; if not, do the next
 7596
 7597 063650                                PRINTF  #BI7                ; print "CHECK OUTPUTS = 9.9976V"
      063650  012746  065061                                                 MOV     #BI7,-(SP)
      063654  012746  000001                                                 MOV     #1,-(SP)
      063660  010600                                                        MOV     SP,R0
      063662  104417                                                        TRAP    C$PNTF
      063664  062706  000004                                                 ADD     #4,SP
 7598 063670                         150$:  GMANIL  BI2,ANSWER,377,YES  ; ask "IN TOLERANCE ?"
      063670  104443                                                        TRAP    C$GMAN
      063672  000404                                                        BR      10003$
      063674  064034                                                        .WORD   ANSWER
      063676  000130                                                        .WORD   T$CODE
      063700  064324                                                        .WORD   BI2
      063702  000377                                                        .WORD   377
      063704                                                          10003$:
 7599 063704  005737  064034                TST     ANSWER              ; in tolerance ?
 7600 063710  001011                        BNE     160$                ; if yes, branch
 7601 063712                                PRINTF  #BI5                ; else print "ADJUST RV2"
      063712  012746  064605                                                 MOV     #BI5,-(SP)
      063716  012746  C00001                                                 MOV     #1,-(SP)
      063722  010600                                                        MOV     SP,R0
      063724  104417                                                        TRAP    C$PNTF
      063726  062706  000004                                                 ADD     #4,SP
 7602 063732  000756                        BR      150$                ; and ask again
 7603
 7604 063734  005737  002234         160$:  TST     QVMODE              ; is quick verify mode selected ?
 7605 063740  001011                        BNE     170$                ; if yes, exit test
 7606 063742  005237  003032                INC     ITRCNT              ; else, increment iteration counter
 7607 063746  005205                        INC     R5                  ; and pass counter
 7608 063750  023727  003032  000002        CMP     ITRCNT,#2           ; iterations completed ?
 7609 063756  001402                        BEQ     170$                ; if yes, branch
 7610 063760  000137  063120                JMP     40$                 ; else do another iteration
 7611
 7612 063764                         170$:  PRINTF  #BI8                ; print "YOU CAN CONTINUE..."
      063764  012746  065163                                                 MOV     #BI8,-(SP)
      063770  012746  000001                                                 MOV     #1,-(SP)
      063774  010600                                                        MOV     SP,R0
      063776  104417                                                        TRAP    C$PNTF
      064000  062706  000004                                                 ADD     #4,SP
 7613 064004                                PRINTF  #BI8A               ; ...
      064004  012746  065262                                                 MOV     #BI8A,-(SP)
      064010  012746  000001                                                 MOV     #1,-(SP)
      064014  010600                                                        MOV     SP,R0
      064016  104417                                                        TRAP    C$PNTF
      064020  062706  000004                                                 ADD     #4,SP
 7614 064024  004737  026162                JSR     PC,WRDY             ; wait for operator to continue
 7615 064030                                EXIT TST
      064030  104432                                                        TRAP    C$EXIT
```

```
TEST 24 - Calibration Test.

        064032  001326                                                                  .WORD   L10061-.
   7616
   7617 064034  000001                  ANSWER: .WORD   1                       ; store for operator replies
   7618
   7619                                          .NLIST  BEX
   7620 064036      045     123     062  TSHD24::.ASCIZ  /%S2%ACALIBRATION TEST%N/
   7621 064066      045     123     062  TSNS24::.ASCIZ  /%S2%ACALIBRATION TEST - DISABLED BY "UAM" FLAG%N/
   7622 064147      045     116     045  BICAL:  .ASCIZ  /%N%ACALIBRATING FOR BIPOLAR MODE - PASS %D1%N/
   7623 064225      045     116     045  BI1:    .ASCIZ  \%N%ACHECK EACH CHANNEL OUTPUT IS -10 VOLTS +/- 4.88 MILLIVOLTS\
   7624 064324      101     122     105  BI2:    .ASCIZ  /ARE THE OUTPUTS WITHIN THE ALLOWED TOLERANCE/
   7625 064401      045     116     045  BI3:    .ASCIZ  /%N%AADJUST OFFSET POTENTIOMETER RV1 UNTIL THE OUTPUTS ARE CORRECT/
   7626 064503      045     116     045  BI4:    .ASCIZ  \%N%ACHECK EACH CHANNEL OUTPUT IS 9.9951 VOLTS +/- 4.88 MILLIVOLTS\
   7627 064605      045     116     045  BI5:    .ASCIZ  /%N%AADJUST GAIN POTENTIOMETER RV2 UNTIL THE OUTPUTS ARE CORRECT/
   7628 064705      045     116     045  UNICAL: .ASCIZ  /%N%ACALIBRATING FOR UNIPOLAR MODE - PASS %D1%N/
   7629 064764      045     116     045  BI6:    .ASCIZ  \%N%ACHECK EACH CHANNEL OUTPUT IS 0 VOLTS +/- 4.88 MILLIVOLTS\
   7630 065061      045     116     045  BI7:    .ASCIZ  \%N%ACHECK EACH CHANNEL OUTPUT IS 9.9976 VOLTS +/- 4.88 MILLIVOLTS\
   7631 065163      045     116     045  BI8:    .ASCIZ  /%N%ATHIS TEST CAN BE DISABLED BY ABORTING THE TEST AND SETTING/
   7632 065262      045     116     045  BI8A:   .ASCIZ  \%N%ATHE "UAM" FLAG. EG. "CONTROL C", THEN CONTINUE/FLAGS:UAM"\
   7633                                          .LIST   BEX
   7634                                          .EVEN
   7635
   7636 065360                          ENDTST
        065360                                                                  L10061:
        065360  104401                                                                  TRAP    C$ETST
   7637
```

TEST 25 - Selectable Output Test.

```
                        .SBTTL  TEST 25 - Selectable Output Test.

                        ;++
                        ; This test allows the user to select output patterns for
                        ; monitoring with an oscilloscope.
                        ;
                        ; The test begins by requesting the patterns for the test :
                        ;
                        ; FIRST VALUE IN MILLIVOLTS (A) 0.0 ?
                        ;
                        ; SECOND VALUE IN MILLIVOLTS (A) 10000.0 ?
                        ;
                        ; After the user has typed in each value, the program prints out
                        ; the actual value which will be output and its equivalent bit
                        ; pattern in octal. The program then asks :
                        ;
                        ; USE BOTH VALUES FOR EACH CHANNEL (L) N ?
                        ;
                        ; Two outputs are made on each of the channels selected in the
                        ; startup questions. Either both values are output for each
                        ; channel or the values switch with each new channel according to
                        ; the response to the third question. The control table is set up
                        ; to make all outputs except the last in mode 0. The last is in
                        ; mode 1 to force a return back to the begining of the table.
                        ; Continuous outputs are made with the DRX11-C in double buffer
                        ; mode until the user types control C.
                        ;--
```

TEST 25 - Selectable Output Test.

```
                                        BGNTST
                                                                        T25::
                                        RFLAGS  R0                      ; read operator flags
                                                                                TRAP    C$RFLA
                                        BIT     #PNT,R0                 ; print test headers ?
                                        BEQ     20$                     ; if not, branch
                                        TST     STAFLG                  ; was test specifically selected ?
                                        BEQ     10$                     ; if not, branch
                                        PRINTF  #TSHD25                 ; else, print test header
                                                                                MOV     #TSHD25,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP
                                        BR      30$                     ; and start the test
                                10$:    PRINTF  #TSNS25                 ; print "TEST NOT SELECTED"
                                                                                MOV     #TSNS25,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP
                                        EXIT TST                        ; and skip the test
                                                                                TRAP    C$EXIT
                                                                                .WORD   L10062-.
                                20$:    TST     STAFLG                  ; was test specifically selected ?
                                        BNE     30$                     ; if yes, branch
                                        EXIT TST                        ; else skip the test
                                                                                TRAP    C$EXIT
                                                                                .WORD   L10062-.
                                ;
                                ; Request the test parameters
                                ;
                                30$:    JSR     PC,DECIN                ; decimal input routine
                                        FVALQ                           ; "FIRST VALUE" prompt
                                        N251                            ; store for first value string
                                        JSR     PC,ADCON                ; convert to 12 bit pattern
                                        MOV     R1,FVAL                 ; and save first value
                                        JSR     PC,DACON                ; convert back to actual output value
                                        PRINTF  #RNDOUT                 ; "ROUNDED TO "
                                                                                MOV     #RNDOUT,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP
                                        JSR     PC,DECOUT               ; "nnnnn.nnn"
                                        PRINTF  #OCTOUT,FVAL            ; "MILLIVOLTS - OCTAL VALUE nnnnn"
                                                                                MOV     FVAL,-(SP)
                                                                                MOV     #OCTOUT,-(SP)
                                                                                MOV     #2,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #6,SP
                                        JSR     PC,DECIN                ; decimal input routine
                                        SVALQ                           ; "SECOND VALUE" prompt
                                        N252                            ; store for second value string
                                        JSR     PC,ADCON                ; convert to 12 bit pattern
                                        MOV     R1,SVAL                 ; and save second value
```

TEST 25 - Selectable Output Test.

```
 7697 065574  004737  026352                  JSR     PC,DACON                ; convert back to actual output value
 7698 065600                                  PRINTF  #RNDOUT                 ; "ROUNDED TO "
      065600  012746  023323                                                          MOV     #RNDOUT,-(SP)
      065604  012746  000001                                                          MOV     #1,-(SP)
      065610  010600                                                                  MOV     SP,R0
      065612  104417                                                                  TRAP    C$PNTF
      065614  062706  000004                                                          ADD     #4,SP
 7699 065620  004737  027640                  JSR     PC,DECOUT               ; "nnnnn.nnn"
 7700 065624                                  PRINTF  #OCTOUT,SVAL            ; "MILLIVOLTS - OCTAL VALUE nnnnn"
      065624  013746  003036                                                          MOV     SVAL,-(SP)
      065630  012746  023341                                                          MOV     #OCTOUT,-(SP)
      065634  012746  000002                                                          MOV     #2,-(SP)
      065640  010600                                                                  MOV     SP,R0
      065642  104417                                                                  TRAP    C$PNTF
      065644  062706  000006                                                          ADD     #6,SP
 7701 065650                                  GMANIL  T255,BVALS,377,YES      ; "USE BOTH VALUES FOR EACH CHANNEL ?"
      065650  104443                                                                  TRAP    C$GMAN
      065652  000404                                                                  BR      10000$
      065654  066542                                                                  .WORD   BVALS
      065656  000130                                                                  .WORD   T$CODE
      065660  066705                                                                  .WORD   T255
      065662  000377                                                                  .WORD   377
      065664                                                              10000$:
 7702
 7703 065664  013701  002230          40$:    MOV     CTIME,R1                ; get the selected conversion time
 7704 065670  052701  040000                  BIS     #PCR,R1                 ; address the PCR
 7705 065674  010177  114734                  MOV     R1,@DRXDBR              ; and enter the conversion time
 7706 065700  052777  000001  114720          BIS     #FNCTO,@DRXSCR          ; transfer to PCR
 7707 065706                                  PRINTF  #PM250                  ; "TEST RUNNING"
      065706  012746  066746                                                          MOV     #PM250,-(SP)
      065712  012746  000001                                                          MOV     #1,-(SP)
      065716  010600                                                                  MOV     SP,R0
      065720  104417                                                                  TRAP    C$PNTF
      065722  062706  000004                                                          ADD     #4,SP
 7708                                         ;
 7709                                         ; Set up the DMA output buffer
 7710                                         ;
 7711 065726  013701  002226                  MOV     LSTCHN,R1               ; get last selected channel
 7712 065732  163701  002224                  SUB     FSTCHN,R1               ; calculate number of channels
 7713 065736  005201                          INC     R1                      ; ...
 7714 065740  010137  066540                  MOV     R1,NCHANS               ; and save the result
 7715 065744  012702  003040                  MOV     #BUFOUT,R2              ; get address of DMA buffer
 7716 065750  012703  000040                  MOV     #32.,R3                 ; store 32 pairs of values
 7717 065754  013701  066540                  MOV     NCHANS,R1               ; for each channel
 7718 065760  005737  066542                  TST     BVALS                   ; both values on each channel ?
 7719 065764  001407                          BEQ     60$                     ; branch if not
 7720
 7721 065766  013722  003034          50$:    MOV     FVAL,(R2)+              ; store first output value
 7722 065772  013722  003036                  MOV     SVAL,(R2)+              ; and second output value
 7723 065776  005301                          DEC     R1                      ; all channels set up ?
 7724 066000  001372                          BNE     50$                     ; if not, set up more
 7725 066002  000414                          BR      70$                     ; branch to repeat 32 times
 7726
 7727 066004  013722  003034          60$:    MOV     FVAL,(R2)+              ; store first output value
 7728 066010  013722  003034                  MOV     FVAL,(R2)+              ; twice
 7729 066014  005301                          DEC     R1                      ; all channels set up ?
 7730 066016  001406                          BEQ     70$                     ; branch if yes
```

TEST 25 - Selectable Output Test.

```
7731 066020  013722  003036                 MOV     SVAL,(R2)+              ; store second output value
7732 066024  013722  003036                 MOV     SVAL,(R2)+              ; twice
7733 066030  005301                         DEC     R1                      ; all channels set up ?
7734 066032  001364                         BNE     60$                     ; if not, do more
7735
7736 066034  005303                 70$:    DEC     R3                      ; repeat buffer 32 times ?
7737 066036  001411                         BEQ     90$                     ; if done, branch
7738 066040  012704  003040                 MOV     #BUFOUT,R4              ; get address of dma buffer
7739 066044  013701  066540                 MOV     NCHANS,R1               ; get number of channels
7740 066050  012422                 80$:    MOV     (R4)+,(R2)+             ; else copy another 2 values
7741 066052  012422                         MOV     (R4)+,(R2)+             ; for each channel
7742 066054  005301                         DEC     R1                      ; all channels set up ?
7743 066056  001374                         BNE     80$                     ; if not, set up more
7744 066060  000765                         BR      70$                     ; else check whether to repeat again
7745                                 ;
7746                                 ; Set up the control table
7747                                 ;
7748 066062  012701  020000         90$:    MOV     #CTA,R1                 ; set up for CWR0
7749 066066  012702  030000                 MOV     #CWR,R2                 ; put first control word
7750 066072  053702  002224                 BIS     FSTCHN,R2               ; contents in R2
7751 066076  012703  030000                 MOV     #CWR,R3                 ; and last control word
7752 066102  053703  002226                 BIS     LSTCHN,R3               ; contents in R3
7753 066106  012704  000002         100$:   MOV     #2,R4                   ; set up 2 control words
7754 066112  010177  114516         110$:   MOV     R1,@DRXDBR              ; address the CTA
7755 066116  052777  000001  114502        BIS     #FNCT0,@DRXSCR          ; and transfer the control word address
7756 066124  010277  114504                 MOV     R2,@DRXDBR              ; address the control word register
7757 066130  052777  000001  114470        BIS     #FNCT0,@DRXSCR          ; and transfer the control word
7758 066136  005201                         INC     R1                      ; next control table address
7759 066140  005304                         DEC     R4                      ; 2 words set up ?
7760 066142  001363                         BNE     110$                    ; if not, set up the next
7761 066144  005202                         INC     R2                      ; next channel
7762 066146  020203                         CMP     R2,R3                   ; last channel set up ?
7763 066150  003756                         BLE     100$                    ; if not, do more
7764
7765 066152  005302                         DEC     R2                      ; go back to previous control word
7766 066154  052702  000100                 BIS     #100,R2                 ; set up last control word in mode 1
7767 066160  010277  114450                 MOV     R2,@DRXDBR              ; address the control word register
7768 066164  052777  000001  114434        BIS     #FNCT0,@DRXSCR          ; and transfer the control word
7769
7770 066172  012777  020000  114434         MOV     #CTA,@DRXDBR            ; point to cwr 0
7771 066200  052777  000001  114420         BIS     #FNCT0,@DRXSCR          ;
7772
7773                                 ;
7774                                 ; Set up the DRX11-C and start the DMA outputs
7775                                 ;
7776 066206  013701  066540                 MOV     NCHANS,R1               ; get number of channels again
7777 066212  000301                         SWAB    R1                      ; 2 conversions per channel * 32
7778 066214  006201                         ASR     R1                      ; ...
7779 066216  006201                         ASR     R1                      ; ...
7780 066220  005401                         NEG     R1                      ; convert to wordcount
7781
7782 066222                                 SETVEC  DRXVEC,#INT1,#PRI07     ; set up DRX11-C vector
     066222  012746  000340                                                         MOV     #PRI07,-(SP)
     066226  012746  032224                                                         MOV     #INT1,-(SP)
     066232  013746  002636                                                         MOV     DRXVEC,-(SP)
     066236  012746  000003                                                         MOV     #3,-(SP)
     066242  104437                                                                 TRAP    C$SVEC
```

TEST 25 - Selectable Output Test.

```
        066244  062706  000010                                                  ADD     #10,SP
   7783
   7784 066250                          120$:   BGNSEG
        066250  104404                                                          TRAP    C$BSEG
   7785 066252                          130$:   SETPRI  #PRI00          ; drop the priority
        066252  012700  000000                                                  MOV     #PRI00,R0
        066256  104441                                                          TRAP    C$SPRI
   7786 066260  005037  002772                  CLR     INTFLG          ; clear the interrupt flag
   7787 066264  012777  010001  114342          MOV     #ACS!GO,@DRXDBR ; set up to start the conversions
   7788 066272  012777  000001  114326          MOV     #FNCT0,@DRXSCR  ; enable conversions and clear the SCR
   7789 066300  012777  000000  114322          MOV     #BAR1,@DRXCOR   ; multiplex to BAR1
   7790 066306  012777  003040  114316          MOV     #BUFOUT,@DRXADR ; set up DMA address
   7791 066314  012777  000400  114306          MOV     #WCR1,@DRXCOR   ; now select WCR1
   7792 066322  010177  114304                  MOV     R1,@DRXADR      ; and write the word count
   7793 066326  012777  001000  114274          MOV     #BAR2,@DRXCOR   ; multiplex to BAR2
   7794 066334  012777  003040  114270          MOV     #BUFOUT,@DRXADR ; set up DMA address
   7795 066342  012777  001400  114260          MOV     #WCR2,@DRXCOR   ; now select WCR2
   7796 066350  010177  114256                  MOV     R1,@DRXADR      ; and write the word count
   7797 066354  012777  020100  114246          MOV     #RUN!ALT,@DRXCOR ; start DMA in alternate buffer mode
   7798 066362  052777  000100  114236          BIS     #IE,@DRXSCR     ; enable interrupts
   7799                                 ;
   7800                                 ; Loop until control C. Report DRX11-C errors or interrupt timeouts.
   7801                                 ;
   7802 066370  005004                  CLR     R4                      ; clear a timeout counter
   7803 066372                          140$:   BREAK                   ; allow operator breakin
        066372  104422                                                          TRAP    C$BRK
   7804 066374  017702  114226          MOV     @DRXSCR,R2              ; save the SCR contents
   7805 066400  005702                  TST     R2                      ; SCR attention bit set ?
   7806 066402  004737  026306          CALL    INSERT                  ; skip branch if error insert selected
   7807 066406  100015                  BPL     150$                    ; if attention bit not set, branch
   7808 066410  042777  020000  114212  BIC     #RUN,@DRXCOR            ; else stop the DMA
   7809 066416  012777  010000  114210  MOV     #ACS,@DRXDBR            ; address the ACS register
   7810 066424  017703  114204          MOV     @DRXDBR,R3              ; and save the contents
   7811 066430                          ERRSOFT 2500,E2500,ESA          ; and print "ATTENTION BIT SET"
        066430  104457                                                          TRAP    C$ERSOFT
        066432  004704                                                          .WORD   2500
        066434  067044                                                          .WORD   E2500
        066436  023670                                                          .WORD   ESA
   7812 066440  000704                  BR      130$                    ; start again
   7813
   7814 066442  005304                  150$:   DEC     R4                      ; is timeout loop finished ?
   7815 066444  001352                  BNE     140$                    ; if not, wait longer
   7816 066446  005737  002772          TST     INTFLG                  ; else, have we had an interrupt ?
   7817 066452  001405                  BEQ     160$                    ; if not, branch
   7818 066454  005037  002772          CLR     INTFLG                  ; else clear the interrupt indicator
   7819 066460  004737  026306          CALL    INSERT                  ; skip branch if error insert selected
   7820 066464  000742                  BR      140$                    ; loop until control C is typed
   7821
   7822 066466                          160$:   SETPRI  #PRI07          ; stop any more interrupts
        066466  012700  000340                                                  MOV     #PRI07,R0
        066472  104441                                                          TRAP    C$SPRI
   7823 066474  042777  020000  114126          BIC     #RUN,@DRXCOR            ; from the DRX11-C
   7824 066502  017702  114120          MOV     @DRXSCR,R2              ; save the SCR contents
   7825 066506  012777  010000  114120          MOV     #ACS,@DRXDBR            ; address the ACS register
   7826 066514  017703  114114          MOV     @DRXDBR,R3              ; and save the contents
   7827 066520                          ERRSOFT 2501,E2501,ESA          ; and print "INTERRUPT TIMEOUT"
        066520  104457                                                          TRAP    C$ERSOFT
```

TEST 25 - Selectable Output Test.

```
        066522  004705                                                          .WORD   2501
        066524  067142                                                          .WORD   F2501
        066526  023670                                                          .WORD   ESA
   7828 066530                          ENDSEG
        066530                                                          10001$:
        066530  104405                                                          TRAP    C$ESEG
   7829 066532  000646                  BR      120$                    ; start again
   7830
   7831 066534                          EXIT TST
        066534  104432                                                          TRAP    C$EXIT
        066536  000466                                                          .WORD   L10062-.
   7832
   7833 066540  000000          NCHANS: .WORD   0                       ; number of selected channels
   7834 066542  000000          BVALS:  .WORD   0                       ; store for both values answer
   7835
   7836                                 .NLIST  BEX
   7837
   7838 066544     060     056     060 N251:   .ASCIZ  /0.0       /     ; store for first input value string
   7839 066557     061     060     060 N252:   .ASCIZ  /10000.0   /     ; store for second input value string
   7840
   7841 066572     045     123     062 TSHD25::.ASCIZ  /%S2%ASELECTABLE OUTPUT TEST%N/
   7842 066630     045     123     062 TSNS25: .ASCIZ  /%S2%ASELECTABLE OUTPUT TEST - NOT SELECTED%N/
   7843
   7844 066705     125     123     105 T255:   .ASCIZ  /USE BOTH VALUES FOR EACH CHANNEL/
   7845 066746     045     101     124 PM250:  .ASCIZ  /%ATEST RUNNING. TYPE CONTROL C TO RETURN TO THE SUPERVISOR.%N/
   7846
   7847 067044     104     122     130 E2500:  .ASCIZ  /DRX11-C SCR ATTENTION BIT SET DURING DOUBLE BUFFER DMA OUTPUT/
   7848 067142     111     116     124 E2501:  .ASCIZ  /INTERRUPT TIMOUT DURING DOUBLE BUFFER DMA OUTPUT/
   7849                                 .LIST   BEX
   7850                                 .EVEN
   7851
   7852 067224                          ENDTST
        067224                                                          L10062:
        067224  104401                                                          TRAP    C$ETST
   7853
```

```
                        .SBTTL  TEST 26 - Loopback Test using ADF01.

                        ;++
                        ; This tests that the AAF01-A conversion accuracy is within a
                        ; specified tolerance.
                        ;
                        ; The test begins by requesting the following parameters :
                        ;
                        ; ADF01 DRX11-C ADDRESS (O) 164210 ?
                        ;
                        ; In reply, you should enter the address of the DRX11-C to which
                        ; an ADF01 is connected. This ADF01 will be used as the other half
                        ; of a looped pair. This question is only asked following a
                        ; "START" command.
                        ;
                        ; CHANGE LOOPBACK PARAMETERS (L) ?
                        ;
                        ; This allows the parameters used for loopback testing to be
                        ; changed. Answering "no" forces the use of the default
                        ; parameters, which cause the most thorough testing, and
                        ; eliminates the need for any more questions. If "Y" is typed, the
                        ; following questions are also asked :
                        ;
                        ; PATTERN (1=RAMP, 2=RANDOM, 3=PAIR) (D) 1 ?
                        ;
                        ; This selects how output is made from the AAF01-A. A sequence of
                        ; 2048 outputs is made. If "1" was typed in response to this
                        ; question, the output ramps up from the lowest voltage to the
                        ; highest and back again each pass of the test. If "2" is
                        ; selected, random values are output. Responding with "3" results
                        ; in the output alternating according tc the answers to the next
                        ; two questions.
                        ;
                        ; FIRST VALUE IN MILLIVOLTS (A) 0.0 ?
                        ;
                        ; SECOND VALUE IN MILLIVOLTS (A) 10000.0 ?
                        ;
                        ; After the user has typed in each value, the program prints out
                        ; the actual value which will be output and its equivalent bit
                        ; pattern in octal. The program then asks :
                        ;
                        ; ALTERNATING CHANNELS (L) Y ?
                        ;
                        ; Normally, 2048 values are output to the first of the selected
                        ; channels and checked to be correct. Outputs are then made to the
                        ; next channel and so on until each channel has been used. If the
                        ; answer to the "alternating channels" question is "yes", 2048
                        ; outputs are made alternating between the first and second of the
                        ; selected channels. After checking the results, a further 2048
                        ; outputs are made to the third and fourth of the selected
                        ; channels and checked. This continues until all channels have
                        ; been checked. If the number of selected channels is odd, the
                        ; last channel is paired with the subsequent channel.
                        ;
                        ; If a paired data pattern and alternating channels are both
                        ; selected, the effect is that the first of the data values is
                        ; output 1024 times to the first of the channels in the pair and
```

```
                        ; the second value to the second channel. This results in a static
                        ; output on each channel, each second channel having the same
                        ; voltage.
                        ;
                        ; After the parameters have been input, a system reset is made and
                        ; the output from each selected channel is read via the
                        ; DRX11-C/ADF01 and checked to be zero.
                        ;
                        ; Starting with the first of the selected channels, 2048 outputs
                        ; are made under DMA and the program waits for DMA ready. The
                        ; output is read back via the DRX11-C/ADF01 analogue input. The
                        ; RATE CLK OUT signal is used to synchronize the two devices. When
                        ; the transfer is complete, the received data is compared with
                        ; that which was output and checked to be within the specified
                        ; tolerance.
                        ;
                        ; The test then proceeds with the next channel and continues until
                        ; all selected channels have been tested.
                        ;
                        ; If "alternating channels" was selected at the start of the test,
                        ; then the list of selected channels is split into pairs. The test
                        ; sequences through the pairs alternating output between each
                        ; channel in the pair.
                        ;--
```

TEST 26 - Loopback Test using ADF01.

```
                                        BGNTST
                                                                        T26::
                                        RFLAGS  R0                      ; read operator flags
                                                                                TRAP    C$RFLA
                                        BIT     #PNT,R0                 ; print test headers ?
                                        BEQ     20$                     ; if not, branch
                                        TST     STAFLG                  ; was test specifically selected ?
                                        BEQ     10$                     ; if not, branch
                                        PRINTF  #TSHD26                 ; else, print test header
                                                                                MOV     #TSHD26,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP
                                        BR      30$                     ; and start the test
                                10$:    PRINTF  #TSNS26                 ; print "TEST NOT SELECTED"
                                                                                MOV     #TSNS26,-(SP)
                                                                                MOV     #1,-(SP)
                                                                                MOV     SP,R0
                                                                                TRAP    C$PNTF
                                                                                ADD     #4,SP
                                        EXIT TST                        ; and skip the test
                                                                                TRAP    C$EXIT
                                                                                .WORD   L10063-.
                                20$:    TST     STAFLG                  ; was test specifically selected ?
                                        BNE     30$                     ; if yes, branch
                                        EXIT TST                        ; else skip the test
                                                                                TRAP    C$EXIT
                                                                                .WORD   L10063-.

                                30$:    CLR     ITRCNT                  ; clear iteration counter
                                        BIS     #RES,@DRXSCR            ; reset the DRX11-C
                                        JSR     PC,WT500                ; wait for 1 millisecond
                                        JSR     PC,WT500                ; (2* 0u microseconds)
                                ;
                                ; Request the test parameters
                                ;
                                        TST     QFLAG1                  ; was it a start command ?
                                        BNE     40$                     ; if not, branch
                                        INC     QFLAG1                  ; else flag address question is being asked
                                        MEMORY  BUFIN                   ; get the address of free memory
                                                                                TRAP    C$MEM
                                                                                MOV     R0,BUFIN
                                        ADD     #100,BUFIN              ; allow space for parameters
                                        MOV     L$LUN,R1                ; get current unit number
                                        ASL     R1                      ; convert to an offset
                                        MOV     ADFADD(R1),ADDST        ; get the default ADF address
                                        GMANID  T261,ADDST,0,-1,160000,177776,YES ; "ADF ADDRESS ?"
                                                                                TRAP    C$GMAN
                                                                                BR      10000$
                                                                                .WORD   ADDST
                                                                                .WORD   T$CODE
                                                                                .WORD   T261
                                                                                .WORD   -1
                                                                                .WORD   T$LOLIM
                                                                                .WORD   T$HILIM
```

W

....B1
....C1
....D1
....E1
....F1
....G1
....H1
....I1
....J1
....K1
....L1
....M1
....N1

....B2
....C2
....D2
....E2
....F2
....G2
....H2
....I2
....J2
....K2
....L2
....M2
....N2

....B3
....C3
....D3
....E3
....F3
....G3
....H3
....I3
....J3
....K3
....L3
....M3
....N3

....B4
....C4
....D4
....E4
....F4
....G4
....H4
....I4
....J4
....K4
....L4
....M4
....N4

....B5
....C5
....D5
....E5
....F5
....G5
....H5
....I5
....J5
....K5
....L5
....M5
....N5

....B6
....C6
....D6
....E6
....F6
....G6
....H6
....I6
....J6
....K6
....L6
....M6
....N6

....B7
....C7
....D7
....E7
....F7
....G7
....H7
....I7
....J7
....K7
....L7
....M7
....N7

....B8
....C8
....D8
....E8
....F8
....G8
....H8
....I8
....J8
....K8
....L8
....M8
....N8

....B9
....C9
....D9
....E9
....F9
....G9
....H9
....I9
....J9
....K9
....L9
....M9
....N9

....B10
....C10
....D10
....E10
....F10
....G10
....H10
....I10
....J10
....K10
....L10
....M10
....N10

....B11
....C11
....D11
....E11
....F11
....G11
....H11
....I11
....J11
....K11
....L11
....M11
....N11

....B12
....C12
....D12
....E12
....F12
....G12
....H12
....I12
....J12
....K12
....L12
....M12
....N12

....B13
....C13
....D13
....E13
....F13
....G13
....H13
....I13
....J13
....K13
....L13
....M13
....N13

....B14
....C14
....D14
....E14
....F14
....G14
....H14
....I14
....J14
....K14
....L14
....M14
....N14

....B15
....C15
....D15
....E15
....F15
....G15
....H15
....I15
....J15
....K15
....L15
....M15
....N15

....B16
....C16
....D16
....E16
....F16
....G16
....H16
....I16
....J16
....K16
....L16
....M16

TEST 26 - Loopback Test using ADF01.

```
      067430                                                                  10000$:
 7968 067430  013761  072034  002650          MOV     ADDST,ADFADD(R1)       ; save the address
 7969 067436  005737  003016          40$:    TST     QFLAG2                 ; start, restart or continue ?
 7970 067442  001131                          BNE     60$                    ; if not, branch
 7971 067444  005237  003016                  INC     QFLAG2                 ; else flag questions are being asked
 7972 067450                                  GMANIL  T262,CHANGE,377,NO     ; "CHANGE PARAMETERS ?"
      067450  104443                                                          TRAP    C$GMAN
      067452  000404                                                          BR      10001$
      067454  072036                                                          .WORD   CHANGE
      067456  000120                                                          .WORD   T$CODE
      067460  072247                                                          .WORD   T262
      067462  000377                                                          .WORD   377
      067464                                                                  10001$:
 7973 067464  005737  072036                  TST     CHANGE                 ; check the answer
 7974 067470  001516                          BEQ     60$                    ; if no, branch
 7975 067472                                  GMANID  T263,PAT,D,-1,1,3,YES  ; "PATTERN ?"
      067472  104443                                                          TRAP    C$GMAN
      067474  000406                                                          BR      10002$
      067476  072040                                                          .WORD   PAT
      067500  000052                                                          .WORD   T$CODE
      067502  072302                                                          .WORD   T263
      067504  177777                                                          .WORD   -1
      067506  000001                                                          .WORD   T$LOLIM
      067510  000003                                                          .WORD   T$HILIM
      067512                                                                  10002$:
 7976 067512  023727  072040  000003          CMP     PAT,#3                 ; pair selected ?
 7977 067520  001074                          BNE     50$                    ; if not, don't ask for values
 7978 067522  004737  026710                  JSR     PC,DECIN               ; decimal input routine
 7979 067526  023236                          FVALQ                          ; "FIRST VALUE" prompt
 7980 067530  072052                          N261                           ; store for first value string
 7981 067532  004737  026510                  JSR     PC,ADCON               ; convert to 12 bit pattern
 7982 067536  010137  003034                  MOV     R1,FVAL                ; and save first value
 7983 067542  004737  026352                  JSR     PC,DACON               ; convert back to actual output value
 7984 067546                                  PRINTF  #RNDOUT                ; "ROUNDED TO "
      067546  012746  023323                                                          MOV     #RNDOUT,-(SP)
      067552  012746  000001                                                          MOV     #1,-(SP)
      067556  010600                                                                  MOV     SP,R0
      067560  104417                                                                  TRAP    C$PNTF
      067562  062706  000004                                                          ADD     #4,SP
 7985 067566  004737  027640                  JSR     PC,DECOUT              ; "nnnnn.nnn"
 7986 067572                                  PRINTF  #OCTOUT,FVAL           ; "MILLIVOLTS - OCTAL VALUE nnnnn"
      067572  013746  003034                                                          MOV     FVAL,-(SP)
      067576  012746  023341                                                          MOV     #OCTOUT,-(SP)
      067602  012746  000002                                                          MOV     #2,-(SP)
      067606  010600                                                                  MOV     SP,R0
      067610  104417                                                                  TRAP    C$PNTF
      067612  062706  000006                                                          ADD     #6,SP
 7987 067616  004737  026710                  JSR     PC,DECIN               ; decimal input routine
 7988 067622  023270                          SVALQ                          ; "SECOND VALUE" prompt
 7989 067624  072065                          N262                           ; store for second value string
 7990 067626  004737  026510                  JSR     PC,ADCON               ; convert to 12 bit pattern
 7991 067632  010137  003036                  MOV     R1,SVAL                ; and save second value
 7992 067636  004737  026352                  JSR     PC,DACON               ; convert back to actual output value
 7993 067642                                  PRINTF  #RNDOUT                ; "ROUNDED TO "
      067642  012746  023323                                                          MOV     #RNDOUT,-(SP)
      067646  012746  000001                                                          MOV     #1,-(SP)
      067652  010600                                                                  MOV     SP,R0
```

TEST 26 - Loopback Test using ADF01.

```
        067654  104417                                                          TRAP    C$PNTF
        067656  062706  000004                                                  ADD     #4,SP
  7994  067662  004737  027640          JSR     PC,DECOUT               ; "nnnnn.nnn"
  7995  067666                          PRINTF  #OCTOUT,SVAL            ; "MILLIVOLTS - OCTAL VALUE nnnnn"
        067666  013746  003036                                                  MOV     SVAL,-(SP)
        067672  012746  023341                                                  MOV     #OCTOUT,-(SP)
        067676  012746  000002                                                  MOV     #2,-(SP)
        067702  010600                                                          MOV     SP,R0
        067704  104417                                                          TRAP    C$PNTF
        067706  062706  000006                                                  ADD     #6,SP
  7996  067712                  50$:    GMANIL  T264,ALTST,377,YES      ; "ALTERNATING CHANNELS ?"
        067712  104443                                                          TRAP    C$GMAN
        067714  000404                                                          BR      10003$
        067716  072042                                                          .WORD   ALTST
        067720  000130                                                          .WORD   T$CODE
        067722  072345                                                          .WORD   T264
        067724  000377                                                          .WORD   377
        067726                                                          10003$:
  7997                          ;
  7998                          ; Set up the addresses for the ADF01's DRX11-C
  7999                          ;
  8000  067726  013701  002074  60$:    MOV     L$LUN,R1                ; get current unit number
  8001  067732  006301                  ASL     R1                      ; convert to an offset
  8002  067734  062701  002650          ADD     #ADFADD,R1              ; point to ADF address store with R1
  8003  067740  011100                  MOV     (R1),R0                 ; get the SCR address of the ADF's DRX
  8004  067742  010037  002640          MOV     R0,ADFSCR               ; and save it
  8005  067746  062700  000002          ADD     #2,R0                   ; add 2
  8006  067752  010037  002642          MOV     R0,ADFCOR               ; to give COR address of ADF's DRX
  8007  067756  062700  000002          ADD     #2,R0                   ; add 2
  8008  067762  010037  002644          MOV     R0,ADFADR               ; to give ADR address of ADF's DRX
  8009  067766  062700  000002          ADD     #2,R0                   ; add 2
  8010  067772  010037  002646          MOV     R0,ADFDBR               ; to give DBR address of ADF's DRX
  8011                          ;
  8012                          ; Check that the ADF01 is there
  8013                          ;
  8014  067776                  SETVEC  #4,#NXM,#PRI07          ; set up NXM trap service routine
        067776  012746  000340                                                  MOV     #PRI07,-(SP)
        070002  012746  032204                                                  MOV     #NXM,-(SP)
        070006  012746  000004                                                  MOV     #4,-(SP)
        070012  012746  000003                                                  MOV     #3,-(SP)
        070016  104437                                                          TRAP    C$SVEC
        070020  062706  000010                                                  ADD     #10,SP
  8015  070024  013701  002640          MOV     ADFSCR,R1               ; get first register address
  8016  070030  012702  000004          MOV     #4,R2                   ; test 4 registers
  8017  070034  005003                  CLR     R3                      ; clear the error flag
  8018
  8019  070036                  70$:    BGNSEG                          ;
        070036  104404                                                          TRAP    C$BSEG
  8020  070040  005037  002770          CLR     NXMFLG                  ; clear the NXM flag
  8021  070044  005711                  TST     (R1)                    ; test register address
  8022  070046  005737  002770          TST     NXMFLG                  ; was there a trap ?
  8023  070052  004737  026306          CALL    INSERT                  ; skip branch if error insert selected
  8024  070056  001405                  BEQ     80$                     ; if no trap, branch
  8025  070060  005203                  INC     R3                      ; else flag the error
  8026  070062                  ERRHRD  2600,E2600,ERDNR        ; print "ADDRESSING ERROR"
        070062  104456                                                          TRAP    C$ERHRD
        070064  005050                                                          .WORD   2600
```

TEST 26 - Loopback Test using ADF01.

```
        070066  072372                                                          .WORD   E2600
        070070  023460                                                          .WORD   ERDNR
   8027 070072                          80$:    ENDSEG                          ;
        070072                                                                          10004$:
        070072  104405                                                          TRAP    C$ESEG
   8028
   8029 070074  062701  000002                  ADD     #2,R2                   ; point to the next register
   8030 070100  005302                          DEC     R2                      ; all registers tested ?
   8031 070102  001355                          BNE     70$                     ; if not, branch
   8032
   8033 070104  005703                          TST     R3                      ; was there an error ?
   8034 070106  001404                          BEQ     90$                     ; if not, branch
   8035 070110                                  DODU    L$LUN                   ; else drop the unit under test
        070110  013700  002074                                                  MOV     L$LUN,R0
        070114  104451                                                          TRAP    C$DODU
   8036 070116                                  DOCLN                           ; and run the clean up routine
        070116  104444                                                          TRAP    C$DCLN
   8037
   8038 070120                          90$:    CLRVEC  #4                      ; restore DRS trap catcher
        070120  012700  000004                                                  MOV     #4,R0
        070124  104436                                                          TRAP    C$CVEC
   8039                                 ;
   8040                                 ; Check that the ADF01 and AAF01-A are in the same mode
   8041                                 ;
   8042 070126                                  BGNSEG
        070126  104404                                                          TRAP    C$BSEG
   8043 070130  012777  010000  112510          MOV     #ACS,@ADFDBR            ; address the ADF ACS register
   8044 070136  017701  112504                  MOV     @ADFDBR,R1              ; save the contents in R1
   8045 070142  012777  010000  112464          MOV     #ACS,@DRXDBR            ; address the AAF ACS register
   8046 070150  017702  112460                  MOV     @DRXDBR,R2              ; save the contents in R2
   8047 070154  042701  173777                  BIC     #^CUNI,R1               ; isolate the mode bits
   8048 070160  042702  173777                  BIC     #^CUNI,R2               ; in R1 and R2
   8049 070164  020102                          CMP     R1,R2                   ; are they the same ?
   8050 070166  001423                          BEQ     120$                    ; if yes, branch
   8051 070170  002405                          BLT     100$                    ; else, branch if ADF is unipolar
   8052 070172  012701  073074                  MOV     #UNIERR,R1              ; set up error message
   8053 070176  012702  073105                  MOV     #BIERR,R2               ; "ADF = UNIPOLAR, AAF = BIPOLAR"
   8054 070202  000404                          BR      110$                    ;
   8055 070204  012701  073105          100$:   MOV     #BIERR,R1               ; set up error message
   8056 070210  012702  073074                  MOV     #UNIERR,R2              ; "ADF = BIPOLAR, AAF = UNIPOLAR"
   8057 070214                          110$:   ERRHRD  2601,E2601,EAAAF        ; print "NOT IN SAME MODE"
        070214  104456                                                          TRAP    C$ERHRD
        070216  005051                                                          .WORD   2601
        070220  072442                                                          .WORD   E2601
        070222  023726                                                          .WORD   EAAAF
   8058 070224                                  ENDSEG                          ;
        070224                                                                          10005$:
        070224  104405                                                          TRAP    C$ESEG
   8059
   8060 070226                                  DODU    L$LUN                   ; drop the unit under test
        070226  013700  002074                                                  MOV     L$LUN,R0
        070232  104451                                                          TRAP    C$DODU
   8061 070234                                  DOCLN                           ; and run the clean up routine
        070234  104444                                                          TRAP    C$DCLN
   8062                                 ;
   8063                                 ; Reset the AAF01-A and set up the conversion rate
   8064                                 ;
```

TEST 26 - Loopback Test using ADF01.

```
8065 070236                            120$:   BCNSEG
     070236  104404                                                          TRAP    C$BSEG
8066 070240  005037  003032                    CLR     ITRCNT                ; clear iteration counter
8067 070244  052777  000040  112354            BIS     #RES,@DRXSCR          ; reset the DRX11-C
8068 070252  004737  026106                    JSR     PC,WT500              ; wait for 1 millisecond
8069 070256  004737  026106                    JSR     PC,WT500              ; (2*500 microseconds)
8070 070262  013701  002230                    MOV     CTIME,R1              ; get the selected conversion time
8071 070266  052701  040000                    BIS     #PCR,R1               ; address the PCR
8072 070272  010177  112336                    MOV     R1,@DRXDBR            ; and enter the conversion time
8073 070276  052777  000001  112322            BIS     #FNCT0,@DRXSCR        ; transfer to PCR
8074                                   ;
8075                                   ; Check that all outputs have been reset to 0 volts
8076                                   ;
8077 070304  013737  002226  066540            MOV     LSTCHN,NCHANS         ; get last selected channel
8078 070312  163737  002224  066540            SUB     FSTCHN,NCHANS         ; calculate number of channels
8079 070320  005237  066540                    INC     NCHANS                ; ...
8080 070324  013737  002224  031006            MOV     FSTCHN,FCHIN          ; get first channel for input
8081 070332  013737  002226  031010            MOV     LSTCHN,LCHIN          ; and last channel
8082 070340  013737  066540  031012            MOV     NCHANS,NCONS          ; and number of conversions
8083 070346  005037  031014                    CLR     EXTCLK                ; don't use external clock
8084 070352  004737  030522                    JSR     PC,ADFIN              ; start the DMA input
8085 070356  005001                            CLR     R1                    ; clear a timeout counter
8086 070360  105777  112254            130$:   TSTB    @ADFSCR               ; is READY set ?
8087 070364  100420                            BMI     140$                  ; if yes, branch
8088 070366  005301                            DEC     R1                    ; decrement timeout counter
8089 070370  004737  026306                    CALL    INSERT                ; skip branch if error insert selected
8090 070374  001371                            BNE     130$                  ; if not timed out, branch
8091 070376  017702  112236                    MOV     @ADFSCR,R2            ; save the SCR contents
8092 070402  012777  010000  112236            MOV     #ACS,@ADFDBR          ; address the ACS register
8093 070410  017703  112232                    MOV     @ADFDBR,R3            ; and save the contents
8094 070414                                    ERRSOFT 2602,E2602,ESA        ; print "TIMEOUT ON DMA INPUT"
     070414  104457                                                          TRAP    C$ERSOFT
     070416  005052                                                          .WORD   2602
     070420  072504                                                          .WORD   E2602
     070422  023670                                                          .WORD   ESA
8095 070424                                    CKLOOP                        ; branch to BGNSEG if loop on error set
     070424  104406                                                          TRAP    C$CLP1
8096
8097 070426  012737  000200  003020  140$:   MOV     #RDY,GOOD             ; save expected SCR contents
8098 070434  017702  112200                    MOV     @ADFSCR,R2            ; and get actual contents
8099 070440  020237  003020                    CMP     R2,GOOD               ; are they the same ?
8100 070444  004737  026306                    CALL    INSERT                ; skip branch if error insert selected
8101 070450  001411                            BEQ     150$                  ; branch if OK
8102 070452  012777  010000  112166            MOV     #ACS,@ADFDBR          ; address the ACS register
8103 070460  017703  112162                    MOV     @ADFDBR,R3            ; and save the contents
8104 070464                                    ERRSOFT 2603,E2603,ESA        ; print "ERROR ON DMA INPUT"
     070464  104457                                                          TRAP    C$ERSOFT
     070466  005053                                                          .WORD   2603
     070470  072531                                                          .WORD   E2603
     070472  023670                                                          .WORD   ESA
8105 070474                            150$:   CKLOOP                        ; branch to BGNSEG if loop on error set
     070474  104406                                                          TRAP    C$CLP1
8106
8107 070476  013701  002224                    MOV     FSTCHN,R1             ; get first channel to test
8108 070502  013702  023040                    MOV     BUFIN,R2              ; get DMA input buffer address
8109 070506  012737  007777  003020            MOV     #7777,GOOD            ; 0 volts in unipolar mode
8110 070514  005737  003026                    TST     MODE                  ; in unipolar mode ?
```

TEST 26 - Loopback Test using ADF01.

```
8111 070520  001403                        BEQ     160$                    ; if yes, branch
8112 070522  012737  003777  003020        MOV     #3777,GOOD              ; else set up 0 volts in bipolar mode
8113 070530  012237  003022          160$: MOV     (R2)+,BAD               ; and get actual value
8114 070534  013700  003020                MOV     GOOD,R0                 ; place the difference
8115 070540  163700  003022                SUB     BAD,R0                  ; in R0
8116 070544  101001                        BHI     170$                    ; if positive, branch
8117 070546  005400                        NEG     R0                      ; else make positive
8118 070550  020037  002232          170$: CMP     R0,TOL                  ; is the difference too big ?
8119 070554  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
8120 070560  003404                        BLE     180$                    ; if difference not too big, branch
8121 070562                                ERRSOFT 2604,E2604,ECHGB        ; else, print "OUTPUTS NOT ZERO"
     070562  104457                                                                TRAP    C$ERSOFT
     070564  005054                                                                .WORD   2604
     070566  072554                                                                .WORD   E2604
     070570  023764                                                                .WORD   ECHGB
8122 070572                          180$: CKLOOP                          ; branch to BGNSEG if loop on error set
     070572  104406                                                                TRAP    C$CLP1
8123 070574  005201                        INC     R1                      ; next channel
8124 070576  020137  002226                CMP     R1,LSTCHN               ; all channels done ?
8125 070602  003752                        BLE     160$                    ; if not, go back
8126 070604                                ENDSEG
     070604                                                                10006$:
     070604  104405                                                                TRAP    C$ESEG
8127                                       ;
8128                                       ; Set up the DMA output buffer
8129                                       ;
8130 070606  012701  003040          190$: MOV     #BUFOUT,R1              ; get address of DMA buffer in R1
8131 070612  023727  072040  000001        CMP     PAT,#1                  ; ramp pattern selected ?
8132 070620  001036                        BNE     230$                    ; if not, branch
8133 070622  012702  004000                MOV     #2048.,R2               ; set up 2048 values for output
8134 070626  013700  072044                MOV     LASTV,R0                ; get previous last value
8135 070632  005700                        TST     R0                      ; was it zero ?
8136 070634  001403                        BEQ     200$                    ; if yes, do upwards ramp
8137 070636  020027  004000                CMP     R0,#2048.               ; are we half way up ?
8138 070642  001007                        BNE     210$                    ; if not, must be downward ramp
8139 070644  010021          200$: MOV     R0,(R1)+                ; load the buffer
8140 070646  005200                        INC     R0                      ; next value
8141 070650  005302                        DEC     R2                      ; all values written ?
8142 070652  001374                        BNE     200$                    ; if not, write next
8143 070654  010037  072044                MOV     R0,LASTV                ; store last value
8144 070660  000450                        BR      270$                    ; go to set up the control table
8145 070662  020027  010000          210$: CMP     R0,#4096.               ; are we half way down a ramp ?
8146 070666  001001                        BNE     220$                    ; if yes, continue down
8147 070670  005300                        DEC     R0                      ; else start a downward ramp
8148 070672  010021          220$: MOV     R0,(R1)+                ; write value
8149 070674  005300                        DEC     R0                      ; next value
8150 070676  005302                        DEC     R2                      ; all values written ?
8151 070700  001374                        BNE     220$                    ; if not, write next
8152 070702  010037  072044                MOV     R0,LASTV                ; else save the last value
8153 070706  100035                        BPL     270$                    ; if it was negative
8154 070710  005037  072044                CLR     LASTV                   ; reset it to zero
8155 070714  000432                        BR      270$                    ; go to set up control table
8156
8157 070716  023727  072040  000003  230$: CMP     PAT,#3                  ; pattern pair selected ?
8158 070724  001011                        BNE     250$                    ; if not, branch
8159 070726  012702  002000                MOV     #1024.,R2               ; set up 2048 word alternating pattern
8160 070732  013721  003034          240$: MOV     FVAL,(R1)+              ; first selected value
```

TEST 26 - Loopback Test using ADF01.

```
 8161 070736  013721  003036                    MOV     SVAL,(R1)+              ; second selected value
 8162 070742  005302                            DEC     R2                      ; buffer set up ?
 8163 070744  001372                            BNE     240$                    ; if not, go back
 8164 070746  000415                            BR      270$                    ; go to set up control table
 8165
 8166 070750  023727  072040  000002  250$:     CMP     PAT,#2                  ; random pattern selected ?
 8167 070756  001011                            BNE     270$                    ; if not, branch
 8168 070760  012702  004000                    MOV     #2048.,R2               ; set up 2048 words
 8169 070764  004737  032142          260$:     JSR     PC,RANDOM               ; get a random pattern
 8170 070770  042700  170000                    BIC     #170000,R0              ; use only 12 bits
 8171 070774  010021                            MOV     R0,(R1)+                ; save the pattern
 8172 070776  005302                            DEC     R2                      ; buffer set up ?
 8173 071000  001371                            BNE     260$                    ; if not, go back
 8174                                 ;
 8175                                 ; Set up the AAF01-A control table - start with 32 dummy outputs and
 8176                                 ; then loop on first 2 control words for 32 more outputs
 8177                                 ;
 8178 071002  013737  002224  072046  270$:     MOV     FSTCHN,CHAN             ; start with first selected channel
 8179 071010                          280$:     BGNSEG
      071010  104404                                                            TRAP    C$BSEG
 8180 071012  012702  020000                    MOV     #CTA,R2                 ; set up to address CTA register
 8181 071016  012737  000100  072050            MOV     #64.,BLOCK              ; do 32 outputs 64 times to give 2048
 8182                                                                           ; conversions
 8183 071024  012705  003040                    MOV     #BUFOUT,R5              ; save DMA output buffer address
 8184
 8185 071030  010277  111600                    MOV     R2,@DRXDBR              ; address the CTA, CWR0
 8186 071034  052777  000001  111564            BIS     #FNCT0,@DRXSCR          ; and transfer the control word address
 8187 071042  013701  072046                    MOV     CHAN,R1                 ; convert using the current channel
 8188 071046  052701  030000                    BIS     #CWR,R1                 ; address the control word register
 8189 071052  010177  111556                    MOV     R1,@DRXDBR              ; and
 8190 071056  052777  000001  111542            BIS     #FNCT0,@DRXSCR          ; transfer the control word
 8191
 8192 071064  005202                            INC     R2                      ; address the CTA, CWR1
 8193 071066  010277  111542                    MOV     R2,@DRXDBR              ;
 8194 071072  052777  000001  111526            BIS     #FNCT0,@DRXSCR          ; and transfer the control word address
 8195 071100  013701  072046                    MOV     CHAN,R1                 ; convert using the current channel
 8196 071104  052701  030100                    BIS     #CWR!100,R1             ; in mode 1
 8197 071110  063701  072042                    ADD     ALTST,R1                ; if alternating channels, use next one
 8198 071114  010177  111514                    MOV     R1,@DRXDBR              ; for second control word
 8199 071120  052777  000001  111500            BIS     #FNCT0,@DRXSCR          ; transfer the control word
 8200
 8201 071126  005202                  290$:     INC     R2                      ; address the CTA, CWR2
 8202 071130  010277  111500                    MOV     R2,@DRXDBR              ;
 8203 071134  052777  000001  111464            BIS     #FNCT0,@DRXSCR          ; and transfer the control word address
 8204 071142  012777  030200  111464            MOV     #CWR!200,@DRXDBR        ; set next 32 CWR's
 8205 071150  052777  000001  111450            BIS     #FNCT0,@DRXSCR          ; for dummy outputs to get clock started
 8206 071156  020227  020041                    CMP     R2,#CTA!33.             ; 32 dummies set up ?
 8207 071162  001361                            BNE     290$                    ; if not, set up more
 8208
 8209 071164  012777  030500  111442            MOV     #CWR!500,@DRXDBR        ; CWR31 to return to CWR0
 8210 071172  052777  000001  111426            BIS     #FNCT0,@DRXSCR          ; transfer the control word address
 8211 071200  012777  020002  111426  300$:     MOV     #CTA!2,@DRXDBR          ; start at CWR2
 8212 071206  052777  000001  111412            BIS     #FNCT0,@DRXSCR          ; transfer the control word address
 8213                                 ;
 8214                                 ; Start the ADF01 DMA input
 8215                                 ;
 8216 071214  013737  072046  031006            MOV     CHAN,FCHIN              ; set up first channel for input
```

TEST 26 - Loopback Test using ADF01.

```
 8217 071222  013737  072046  031010          MOV     CHAN,LCHIN              ; last channel same as the first
 8218 071230  063737  072042  031010          ADD     ALTST,LCHIN             ; except if using alternating channels
 8219 071236  012737  000100  031012          MOV     #64.,NCONS              ; 64 conversions (32 are dummies)
 8220 071244  012737  000004  031014          MOV     #ECE,EXTCLK             ; use external clock
 8221 071252  004737  030522                  JSR     PC,ADFIN                ; start the DMA input
 8222                                 ;
 8223                                 ; Start the AAF01-A DMA output
 8224                                 ;
 8225 071256  012777  010001  111350          MOV     #ACS!GO,@DRXDBR         ; set up to start the conversions
 8226 071264  012777  000001  111334          MOV     #FNCTO,@DRXSCR          ; enable conversions and clear the SCR
 8227 071272  012777  000000  111330          MOV     #BAR1,@DRXCOR           ; multiplex to BAR1
 8228 071300  010577  111326                  MOV     R5,@DRXADR              ; set up DMA address
 8229 071304  012777  000400  111316          MOV     #WCR1,@DRXCOR           ; now select WCR1
 8230 071312  012777  177740  111312          MOV     #-32.,@DRXADR           ; and write the word count
 8231 071320  012777  020000  111302          MOV     #RUN,@DRXCOR            ; start the DMA
 8232                                 ;
 8233                                 ; Wait for the conversions to complete
 8234                                 ;
 8235 071326  013701  002230                  MOV     CTIME,R1                ; get the conversion time
 8236 071332  005002                          CLR     R2                      ; high byte is zero
 8237 071334  004737  027776                  JSR     PC,MUL                  ; multiply by
 8238 071340  000100                          64.                             ; the number of conversions
 8239 071342  004737  030072                  JSR     PC,DIV                  ; divide by
 8240 071346  000175                          125.                            ; 125
 8241
 8242 071350  005701                  310$:   TST     R1                      ; loop until R1/R2 is zero
 8243 071352  001410                          BEQ     330$                    ;
 8244 071354  005301                  320$:   DEC     R1                      ;
 8245 071356  004737  026114                  JSR     PC,WT25                 ; wait 25 microseconds
 8246 071362  032777  100000  111250          BIT     #ATT,@ADFSCR            ; has the transfer finished ?
 8247 071370  001005                          BNE     340$                    ; if yes, jump out of the wait loop
 8248 071372  000766                          BR      310$                    ;
 8249 071374  005702                  330$:   TST     R2                      ;
 8250 071376  001402                          BEQ     340$                    ; total timeout = number of conversions
 8251 071400  005302                          DEC     R2                      ;                 * conversion time * 2
 8252 071402  000764                          BR      320$                    ;
 8253                                 ;
 8254                                 ; Check that the DMA loopback completed correctly
 8255                                 ;
 8256 071404  012737  000200  003020  340$:   MOV     #RDY,GOOD               ; save expected ADF SCR contents
 8257 071412  017702  111222                  MOV     @ADFSCR,R2              ; and get actual contents
 8258 071416  020237  003020                  CMP     R2,GOOD                 ; are they the same ?
 8259 071422  004737  026306                  CALL    INSERT                  ; skip branch if error insert selected
 8260 071426  001411                          BEQ     350$                    ; branch if OK
 8261 071430  012777  010000  111210          MOV     #ACS,@ADFDBR            ; address the ACS register
 8262 071436  017703  111204                  MOV     @ADFDBR,R3              ; and save the contents
 8263 071442                                  ERRSOFT 2603,E2603,ESA          ; print "ERROR ON DMA INPUT"
      071442  104457                                                                  TRAP    C$ERSOFT
      071444  005053                                                                  .WORD   2603
      071446  072531                                                                  .WORD   E2603
      071450  023670                                                                  .WORD   ESA
 8264 071452                          350$:   CKLOOP                          ; branch to BGNSEG if loop on error set
      071452  104406                                                                  TRAP    C$CLP1
 8265
 8266 071454  012737  100600  003020  360$:   MOV     #ATT!STAT0!RDY,GOOD     ; get expected SCR contents
 8267 071462  017737  111140  003022          MOV     @DRXSCR,BAD             ; and actual contents
 8268 071470  023737  003022  003020          CMP     BAD,GOOD                ; are they the same ?
```

TEST 26 - Loopback Test using ADF01.

```
 8269 071476  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 8270 071502  001404                        BEQ     370$                    ; branch if SCR is OK
 8271 071504                                ERRSOFT 2605,E2605,EGB          ; else print "SCR INCORRECT"
      071504  104457                                                            TRAP    C$ERSOFT
      071506  005055                                                            .WORD   2605
      071510  072622                                                            .WORD   E2605
      071512  023416                                                            .WORD   EGB
 8272 071514                        370$:   CKLOOP                          ; branch to BGNSEG if loop on error set
      071514  104406                                                            TRAP    C$CLP1
 8273
 8274 071516  012737  010000  003020        MOV     #ACS,GOOD               ; get expected ACS contents
 8275 071524  012777  010000  111102        MOV     #ACS,@DRXDBR            ; address the ACS register
 8276 071532  017737  111076  003022        MOV     @DRXDBR,BAD             ; and get the actual contents
 8277 071540  013700  003022                MOV     BAD,R0                  ; get the contents into R0
 8278 071544  042700  171777                BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
 8279 071550  050037  003020                BIS     R0,GOOD                 ; set them in GOOD if they are set
 8280 071554  023737  003022  003020        CMP     BAD,GOOD                ; contents correct ?
 8281 071562  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 8282 071566  001404                        BEQ     380$                    ; branch if ACS is OK
 8283 071570                                ERRSOFT 2606,E2606,EGB          ; else print "ACS INCORRECT"
      071570  104457                                                            TRAP    C$ERSOFT
      071572  005056                                                            .WORD   2606
      071574  072671                                                            .WORD   E2606
      071576  023416                                                            .WORD   EGB
 8284 071600                        380$:   CKLOOP                          ; branch to BGNSEG if loop on error set
      071600  104406                                                            TRAP    C$CLP1
 8285                               ;
 8286                               ; Check that loopback data is within tolerance
 8287                               ;
 8288 071602  013701  072046                MOV     CHAN,R1                 ; check current channel
 8289 071606  013702  023040                MOV     BUFIN,R2                ; and input buffer in R2
 8290 071612  062702  000100                ADD     #64.,R2                 ; ignore dummy inputs
 8291 071616  012704  000040                MOV     #32.,R4                 ; check 32 words
 8292
 8293 071622  012537  003020        390$:   MOV     (R5)+,GOOD              ; save expected value
 8294 071626  012237  003022                MOV     (R2)+,BAD               ; and get actual value
 8295 071632  013700  003020                MOV     GOOD,R0                 ; place the difference
 8296 071636  163700  003022                SUB     BAD,R0                  ; in R0
 8297 071642  101001                        BHI     400$                    ; if positive, branch
 8298 071644  005400                        NEG     R0                      ; else make positive
 8299 071646  020037  002232        400$:   CMP     R0,TOL                  ; is the difference too big ?
 8300 071652  004737  026306                CALL    INSERT                  ; skip branch if error insert selected
 8301 071656  003414                        BLE     410$                    ; if difference not too big, branch
 8302 071660                                ERRSOFT 2607,E2607,ECHGB        ; else, print "DATA NOT WITHIN TOLERANCE"
      071660  104457                                                            TRAP    C$ERSOFT
      071662  005057                                                            .WORD   2607
      071664  073016                                                            .WORD   E2607
      071666  023764                                                            .WORD   ECHGB
 8303 071670                                PRINTX  #MSG26                  ; and "RUN TEST 24 TO CALIBRATE"
      071670  012746  072744                                                    MOV     #MSG26,-(SP)
      071674  012746  000001                                                    MOV     #1,-(SP)
      071700  010600                                                            MOV     SP,R0
      071702  104415                                                            TRAP    C$PNTX
      071704  062706  000004                                                    ADD     #4,SP
 8304 071710                        410$:   CKLOOP                          ; branch to BGNSEG if loop on error set
      071710  104406                                                            TRAP    C$CLP1
 8305
```

TEST 26 - Loopback Test using ADF01.

```
8306 071712  020137  072046                 CMP     R1,CHAN          ; are we checking the current channel ?
8307 071716  001003                         BNE     420$             ; if not, branch
8308 071720  063701  072042                 ADD     ALTST,R1         ; else go to alternate channel
8309 071724  000402                         BR      430$             ; (this does nothing if not
8310 071726  163701  072042          420$:  SUB     ALTST,R1         ; in alternating mode)
8311
8312 071732  005304                 430$:  DEC     R4               ; all buffer checked ?
8313 071734  001332                         BNE     390$             ; if not, go back
8314 071736  005337  072050                 DEC     BLOCK            ; 64 DMA blocks done ?
8315 071742  001402                         BEQ     440$             ; if yes, branch
8316 071744  000137  071200                 JMP     300$             ; else, do more
8317 071750                         440$:  ENDSEG
     071750                                                                      10007$:
     071750  104405                                                                TRAP    C$ESEG
8318
8319 071752  013701  072046                 MOV     CHAN,R1          ; get current channel
8320 071756  063701  072042                 ADD     ALTST,R1         ; if alternating, skip next channel
8321 071762  020137  002226                 CMP     R1,LSTCHN        ; all channels done ?
8322 071766  002005                         BGE     450$             ; if yes, exit test
8323 071770  005201                         INC     R1               ; else, go to next channel
8324 071772  010137  072046                 MOV     R1,CHAN          ; and save new channel number
8325 071776  000137  071010                 JMP     280$             ; test the next channel
8326
8327 072002  005737  002234          450$:  TST     QVMODE           ; is quick verify mode selected ?
8328 072006  001010                         BNE     460$             ; if yes, exit test
8329 072010  005237  003032                 INC     ITRCNT           ; else, increment iteration counter
8330 072014  023727  003032  000004         CMP     ITRCNT,#4        ; iterations completed ?
8331 072022  001402                         BEQ     460$             ; if yes, branch
8332 072024  000137  070606                 JMP     190$             ; else do another iteration
8333
8334 072030                          460$:  EXIT TST
     072030  104432                                                                TRAP    C$EXIT
     072032  001064                                                                .WORD   L10063-.
8335
8336 072034  000000                  ADDST:  .WORD   0                ; store for ADF01 address
8337 072036  000000                  CHANGE: .WORD   0                ; store for "CHANGE PARAMETERS" answer
8338 072040  000001                  PAT:    .WORD   1                ; store for "PATTERN" answer
8339 072042  000001                  ALTST:  .WORD   1                ; store for "ALTERNATING CHANNELS" answer
8340 072044  000000                  LASTV:  .WORD   0                ; ramp up or down value
8341
8342 072046  000000                  CHAN:   .WORD   0                ; current channel to test
8343 072050  000000                  BLOCK:  .WORD   0                ; counter for number of blocks DMA'd
8344
8345                                         .NLIST  BEX
8346
8347 072052    060    056    060    N261:   .ASCIZ  /0.0       /       ; store for first input value string
8348 072065    061    060    060    N262:   .ASCIZ  /10000.0   /       ; store for second input value string
8349
8350 072100    045    123    062    TSHD26::.ASCIZ  /%S2%ALOOPBACK TEST USING ADF01%N/
8351 072141    045    123    062    TSNS26::.ASCIZ  /%S2%ALOOPBACK TEST USING ADF01 - NOT SELECTED%N/
8352
8353 072221    101    104    106    T261:   .ASCIZ  /ADF01 DRX11-C ADDRESS/
8354 072247    103    110    101    T262:   .ASCIZ  /CHANGE LOOPBACK PARAMETERS/
8355 072302    120    101    124    T263:   .ASCIZ  /PATTERN (1=RAMP, 2=RANDOM, 3=PAIR)/
8356 072345    101    114    124    T264:   .ASCIZ  /ALTERNATING CHANNELS/
8357
8358 072372    101    104    106    E2600:  .ASCIZ  /ADF01 DRX11-C REGISTER ADDRESSING ERROR/
```

TEST 26 - Loopback Test using ADF01.

```
 8359 072442      101     104     106  E2601:  .ASCIZ  /ADF01 NOT IN SAME MODE AS AAF01-A/
 8360 072504      124     111     115  E2602:  .ASCIZ  /TIMEOUT ON DMA INPUT/
 8361 072531      105     122     122  E2603:  .ASCIZ  /ERROR ON DMA INPUT/
 8362 072554      117     125     124  E2604:  .ASCIZ  /OUTPUTS NOT ZERO AFTER A SYSTEM RESET/
 8363 072622      104     122     130  E2605:  .ASCIZ  /DRX11-C SCR INCORRECT IN LOOPBACK TEST/
 8364 072671      101     103     123  E2606:  .ASCIZ  /ACS REGISTER INCORRECT AFTER LOOPBACK TEST/
 8365 072744      045     116     045  MSG26:  .ASCIZ  /%N%ARUN TEST 24 TO CALIBRATE THE MODULE%N/
 8366 073016      114     117     117  E2607:  .ASCIZ  /LOOP BACK DATA NOT WITHIN SPECIFIED TOLERANCE/
 8367
 8368 073074      125     116     111  UNIERR: .ASCIZ  /UNIPOLAR/
 8369 073105      102     111     120  BIERR:  .ASCIZ  /BIPOLAR/
 8370                                  .LIST   BEX
 8371                                  .EVEN
 8372
 8373 073116                          ENDTST
      073116                                                  L10063:
      073116  104401                                                    TRAP    C$ETST
 8374
```

```
                        .SBTTL  TEST 27 - Loopback Test using IEX11-A Interface.

                        ;++
                        ; This tests that the AAF01-A conversion accuracy is within a
                        ; specified tolerance. The output from the AAF01-A is read using
                        ; a digital voltmeter connected to an IEX11-A interface. This
                        ; provides more accurate measurement than the ADF01 used in test
                        ; 26.
                        ;
                        ; The test begins by requesting the following parameters :
                        ;
                        ; IEX11-A ADDRESS (O) 164100 ?
                        ;
                        ; In reply, you should enter the address of the IEX11-A to which
                        ; a digital voltmeter is connected. This IEX11-A will be used to
                        ; read back the outputs of the AAF01-A. This question is only
                        ; asked following a "start" command.
                        ;
                        ; CHANGE LOOPBACK PARAMETERS (L) ?
                        ;
                        ; This allows the parameters used for loopback testing to be
                        ; changed. Answering "no" forces the use of the default
                        ; parameters, which cause the most thorough testing, and
                        ; eliminates the need for any more questions. If "Y" is typed, the
                        ; following questions are also asked :
                        ;
                        ; LOOPBACK TOLERANCE IN MILLIVOLTS (A) 10.0 ?
                        ;
                        ; If the difference between output and input values is greater
                        ; than this value, an error message will be output.
                        ;
                        ; PATTERN (1=RAMP, 2=RANDOM, 3=PAIR) (D) 1 ?
                        ;
                        ; This selects how output is made from the AAF01-A. A sequence of
                        ; 4095 decimal outputs is made. If "1" was typed in response to
                        ; this question, the output ramps up from the lowest voltage to
                        ; the highest and back again. If "2" is selected, random values
                        ; are output. Responding with "3" results in the output
                        ; alternating according to the answers to the next two questions.
                        ;
                        ; FIRST VALUE IN MILLIVOLTS (A) 0.0 ?
                        ;
                        ; SECOND VALUE IN MILLIVOLTS (A) 10000.0 ?
                        ;
                        ; After the user has typed in each value, the program prints out
                        ; the actual value which will be output and its equivalent bit
                        ; pattern in octal.
                        ;
                        ; After the parameters have been input, a system reset is made.
                        ;
                        ; Using the channel selected as "FIRST CHANNEL" in the startup
                        ; questions, 4096 outputs are made. After each output, the IEX11-A
                        ; is used to check that the resulting voltage is within the
                        ; specified tolerance. At the end of each pass, the maximum
                        ; deviation from the expected value is printed.
                        ;--
```

TEST 27 - Loopback Test using IEX11-A Interface.

```
 8433 073120                            BGNTST
      073120                                                    T27::
 8434
 8435 073120                            RFLAGS  R0              ; read operator flags
      073120  104421                                                   TRAP    C$RFLA
 8436 073122  032700  001000            BIT     #PNT,R0         ; print test headers ?
 8437 073126  001426                    BEQ     20$             ; if not, branch
 8438 073130  005737  003030            TST     STAFLG          ; was test specifically selected ?
 8439 073134  001411                    BEQ     10$             ; if not, branch
 8440 073136                            PRINTF  #TSHD27         ; else, print test header
      073136  012746  074757                                           MOV     #TSHD27,-(SP)
      073142  012746  000001                                           MOV     #1,-(SP)
      073146  010600                                                   MOV     SP,R0
      073150  104417                                                   TRAP    C$PNTF
      073152  062706  000004                                           ADD     #4,SP
 8441 073156  000417                    BR      30$             ; and start the test
 8442 073160                    10$:    PRINTF  #TSNS27         ; print "TEST NOT SELECTED"
      073160  012746  075034                                           MOV     #TSNS27,-(SP)
      073164  012746  000001                                           MOV     #1,-(SP)
      073170  010600                                                   MOV     SP,R0
      073172  104417                                                   TRAP    C$PNTF
      073174  062706  000004                                           ADD     #4,SP
 8443 073200                            EXIT TST                ; and skip the test
      073200  104432                                                   TRAP    C$EXIT
      073202  002274                                                   .WORD   L10064-.
 8444 073204  005737  003030    20$:    TST     STAFLG          ; was test specifically selected ?
 8445 073210  001002                    BNE     30$             ; if yes, branch
 8446 073212                            EXIT TST                ; else skip the test
      073212  104432                                                   TRAP    C$EXIT
      073214  002262                                                   .WORD   L10064-.
 8447
 8448 073216  005037  003032    30$:    CLR     ITRCNT          ; clear iteration counter
 8449 073222  052777  000040  107376    BIS     #RES,@DRXSCR    ; reset the DRX11-C
 8450 073230  004737  026106            JSR     PC,WT500        ; wait for 1 millisecond
 8451 073234  004737  026106            JSR     PC,WT500        ; (2*500 microseconds)
 8452                                   ;
 8453                                   ; Request the test parameters
 8454                                   ;
 8455 073240  005737  003014            TST     QFLAG1          ; has address already been asked ?
 8456 073244  001012                    BNE     40$             ; if yes, branch
 8457 073246  005237  003014            INC     QFLAG1          ; else flag that we are asking now
 8458 073252                            GMANID  T270,IEXADD,0,-1,160000,177776,YES ; "IEX ADDRESS ?"
      073252  104443                                                   TRAP    C$GMAN
      073254  000406                                                   BR      10000$
      073256  032022                                                   .WORD   IEXADD
      073260  000032                                                   .WORD   T$CODE
      073262  075116                                                   .WORD   T270
      073264  177777                                                   .WORD   -1
      073266  160000                                                   .WORD   T$LOLIM
      073270  177776                                                   .WORD   T$HILIM
      073272                                                    10000$:
 8459 073272  005737  003016    40$:    TST     QFLAG2          ; start, restart or continue command ?
 8460 073276  001133                    BNE     50$             ; if not, branch
 8461 073300  005237  003016            INC     QFLAG2          ; else flag questions are being asked
 8462 073304                            GMANIL  T262,CHANGE,377,NO      ; "CHANGE PARAMETERS ?"
      073304  104443                                                   TRAP    C$GMAN
      073306  000404                                                   BR      10001$
```

TEST 27 - Loopback Test using IEX11-A Interface.

```
        073310  072036                                                          .WORD   CHANGE
        073312  000120                                                          .WORD   T$CODE
        073314  072247                                                          .WORD   T262
        073316  000377                                                          .WORD   377
        073320                                                          10001$:
  8463  073320  005737  072036          TST     CHANGE                  ; check the answer
  8464  073324  001520                 BEQ     50$                     ; if no, branch
  8465  073326  004737  026710          JSR     PC,DECIN                ; decimal input routine
  8466  073332  075136                 T271                            ; "LOOPBACK TOLERANCE IN MV ?"
  8467  073334  074744                 N27T                            ; store for tolerance value string
  8468  073336  010137  074704          MOV     R1,LTOL1                ; and save tolerance
  8469  073342  010237  074706          MOV     R2,LTOL2                ; high and low word
  8470  073346                         GMANID  T263,PAT,D,-1,1,3,YES   ; "PATTERN ?"
        073346  104443                                                          TRAP    C$GMAN
        073350  000406                                                          BR      10002$
        073352  072040                                                          .WORD   PAT
        073354  000052                                                          .WORD   T$CODE
        073356  072302                                                          .WORD   T263
        073360  177777                                                          .WORD   -1
        073362  000001                                                          .WORD   T$LOLIM
        073364  000003                                                          .WORD   T$HILIM
        073366                                                          10002$:
  8471  073366  023727  072040  000003  CMP     PAT,#3                  ; pair selected ?
  8472  073374  001074                 BNE     50$                     ; if not, don't ask for values
  8473  073376  004737  026710          JSR     PC,DECIN                ; decimal input routine
  8474  073402  023236                 FVALQ                           ; "FIRST VALUE" prompt
  8475  073404  074716                 N271                            ; store for first value string
  8476  073406  004737  026510          JSR     PC,ADCON                ; convert to 12 bit pattern
  8477  073412  010137  003034          MOV     R1,FVAL                 ; and save first value
  8478  073416  004737  026352          JSR     PC,DACON                ; convert back to actual output value
  8479  073422                         PRINTF  #RNDOUT                 ; "ROUNDED TO "
        073422  012746  023323                                                  MOV     #RNDOUT,-(SP)
        073426  012746  000001                                                  MOV     #1,-(SP)
        073432  010600                                                          MOV     SP,R0
        073434  104417                                                          TRAP    C$PNTF
        073436  062706  000004                                                  ADD     #4,SP
  8480  073442  004737  027640          JSR     PC,DECOUT               ; "nnnnn.nnn"
  8481  073446                         PRINTF  #OCTOUT,FVAL            ; "MILLIVOLTS - OCTAL VALUE nnnnn"
        073446  013746  003034                                                  MOV     FVAL,-(SP)
        073452  012746  023341                                                  MOV     #OCTOUT,-(SP)
        073456  012746  000002                                                  MOV     #2,-(SP)
        073462  010600                                                          MOV     SP,R0
        073464  104417                                                          TRAP    C$PNTF
        073466  062706  000006                                                  ADD     #6,SP
  8482  073472  004737  026710          JSR     PC,DECIN                ; decimal input routine
  8483  073476  023270                 SVALQ                           ; "SECOND VALUE" prompt
  8484  073500  074731                 N272                            ; store for second value string
  8485  073502  004737  026510          JSR     PC,ADCON                ; convert to 12 bit pattern
  8486  073506  010137  003036          MOV     R1,SVAL                 ; and save second value
  8487  073512  004737  026352          JSR     PC,DACON                ; convert back to actual output value
  8488  073516                         PRINTF  #RNDOUT                 ; "ROUNDED TO "
        073516  012746  023323                                                  MOV     #RNDOUT,-(SP)
        073522  012746  000001                                                  MOV     #1,-(SP)
        073526  010600                                                          MOV     SP,R0
        073530  104417                                                          TRAP    C$PNTF
        073532  062706  000004                                                  ADD     #4,SP
  8489  073536  004737  027640          JSR     PC,DECOUT               ; "nnnnn.nnn"
```

TEST 27 - Loopback Test using IEX11-A Interface.

```
 8490 073542                            PRINTF  #OCTOUT,SVAL          ; "MILLIVOLTS - OCTAL VALUE nnnnn"
      073542  013746  003036                                                    MOV     SVAL,-(SP)
      073546  012746  023341                                                    MOV     #OCTOUT,-(SP)
      073552  012746  000002                                                    MOV     #2,-(SP)
      073556  010600                                                            MOV     SP,R0
      073560  104417                                                            TRAP    C$PNTF
      073562  062706  000006                                                    ADD     #6,SP
 8491                                   ;
 8492                                   ; Check that the IEx11-A is there
 8493                                   ;
 8494 073566                            50$:    SETVEC  #4,#NXM,#PRI07  ; set up NXM trap service routine
      073566  012746  000340                                                    MOV     #PRI07,-(SP)
      073572  012746  032204                                                    MOV     #NXM,-(SP)
      073576  012746  000004                                                    MOV     #4,-(SP)
      073602  012746  000003                                                    MOV     #3,-(SP)
      073606  104437                                                            TRAP    C$SVEC
      073610  062706  000010                                                    ADD     #10,SP
 8495 073614  013701  032022            MOV     IEXADD,R1               ; get first register address
 8496 073620  012702  000010            MOV     #8.,R2                  ; test 8 registers
 8497 073624  005003                    CLR     R3                      ; clear the error flag
 8498
 8499 073626                            60$:    BGNSEG                  ;
      073626  104404                                                            TRAP    C$BSEG
 8500 073630  005037  002770            CLR     NXMFLG                  ; clear the NXM flag
 8501 073634  005711                    TST     (R1)                    ; test register address
 8502 073636  005737  002770            TST     NXMFLG                  ; was there a trap ?
 8503 073642  004737  026306            CALL    INSERT                  ; skip branch if error insert selected
 8504 073646  001405                    BEQ     70$                     ; if no trap, branch
 8505 073650  005203                    INC     R3                      ; else flag the error
 8506 073652                            ERRHRD  2700,E2700,ERDNR        ; print "ADDRESSING ERROR"
      073652  104456                                                            TRAP    C$ERHRD
      073654  005214                                                            .WORD   2700
      073656  075177                                                            .WORD   E2700
      073660  023460                                                            .WORD   ERDNR
 8507 073662                            70$:    ENDSEG                  ;
      073662                                                                10003$:
      073662  104405                                                            TRAP    C$ESEG
 8508
 8509 073664  062701  000002            ADD     #2,R1                   ; point to the next register
 8510 073670  005302                    DEC     R2                      ; all registers tested ?
 8511 073672  001355                    BNE     60$                     ; if not, branch
 8512
 8513 073674  005703                    TST     R3                      ; was there an error ?
 8514 073676  001404                    BEQ     80$                     ; if not, branch
 8515 073700                            DODU    L$LUN                   ; else drop the unit under test
      073700  013700  002074                                                    MOV     L$LUN,R0
      073704  104451                                                            TRAP    C$DODU
 8516 073706                            DOCLN                           ; and run the clean up routine
      073706  104444                                                            TRAP    C$DCLN
 8517
 8518 073710                            80$:    CLRVEC  #4              ; restore DRS trap catcher
      073710  012700  000004                                                    MOV     #4,R0
      073714  104436                                                            TRAP    C$CVEC
 8519                                   ;
 8520                                   ; Reset the AAF01-A and set up the conversion rate
 8521                                   ;
 8522 073716  005037  003032            CLR     ITRCNT                  ; clear iteration counter
```

TEST 27 - Loopback Test using IEX11-A Interface.

```
8523 073722  052777  000040  106676          BIS     #RES,@DRXSCR            ; reset the DRX11-C
8524 073730  004737  026106                  JSR     PC,WT500                ; wait for 1 millisecond
8525 073734  004737  026106                  JSR     PC,WT500                ; (2*500 microseconds)
8526 073740  013701  002230                  MOV     CTIME,R1                ; get the selected conversion time
8527 073744  052701  040000                  BIS     #PCR,R1                 ; address the PCR
8528 073750  010177  106660                  MOV     R1,@DRXDBR              ; and enter the conversion time
8529 073754  052777  000001  106644          BIS     #FNCTO,@DRXSCR          ; transfer to PCR
8530                                ;
8531                                ; Set up the output buffer
8532                                ;
8533 073762  005037  074712          90$:    CLR     HDEV                    ; clear maximum deviation high
8534 073766  005037  074714                  CLR     LDEV                    ; and low words
8535 073772                                  BGNSEG
     073772  104404                                                          TRAP    C$BSEG
8536 073774  012701  003040                  MOV     #BUFOUT,R1              ; get address of buffer in R1
8537 074000  023727  072040  000001          CMP     PAT,#1                  ; ramp pattern selected ?
8538 074006  001024                          BNE     130$                    ; if not, branch
8539 074010  005737  074710                  TST     UPSW                    ; was the last ramp upwards ?
8540 074014  001011                          BNE     110$                    ; if yes, branch
8541 074016  005237  074710                  INC     UPSW                    ; else flag this is upward ramp
8542 074022  005002                          CLR     R2                      ; start with 0
8543 074024  010221                  100$:   MOV     R2,(R1)+                ; load the buffer
8544 074026  005202                          INC     R2                      ; next value
8545 074030  020227  007777                  CMP     R2,#4095.               ; all values written ?
8546 074034  003773                          BLE     100$                    ; if not, write next
8547 074036  000442                          BR      170$                    ; go to set up the control table
8548 074040  005037  074710          110$:   CLR     UPSW                    ; flag this is downward ramp
8549 074044  012702  007777                  MOV     #4095.,R2               ; start with highest value
8550 074050  010221                  120$:   MOV     R2,(R1)+                ; write value
8551 074052  005302                          DEC     R2                      ; next value
8552 074054  001375                          BNE     120$                    ; until all done down to 0
8553 074056  000432                          BR      170$                    ; go to set up control table
8554
8555 074060  023727  072040  000003  130$:   CMP     PAT,#3                  ; pattern pair selected ?
8556 074066  001011                          BNE     150$                    ; if not, branch
8557 074070  012702  004000                  MOV     #2048.,R2               ; set up 4096 word alternating pattern
8558 074074  013721  003034          140$:   MOV     FVAL,(R1)+              ; first selected value
8559 074100  013721  003036                  MOV     SVAL,(R1)+              ; second selected value
8560 074104  005302                          DEC     R2                      ; buffer set up ?
8561 074106  001372                          BNE     140$                    ; if not, go back
8562 074110  000415                          BR      170$                    ; go to set up control table
8563
8564 074112  023727  072040  000002  150$:   CMP     PAT,#2                  ; random pattern selected ?
8565 074120  001011                          BNE     170$                    ; if not, branch
8566 074122  012702  010000                  MOV     #4096.,R2               ; set up 4096 words
8567 074126  004737  032142          160$:   JSR     PC,RANDOM               ; get a random pattern
8568 074132  042700  170000                  BIC     #170000,R0              ; use only 12 bits
8569 074136  010021                          MOV     R0,(R1)+                ; save the pattern
8570 074140  005302                          DEC     R2                      ; buffer set up ?
8571 074142  001371                          BNE     160$                    ; if not, go back
8572                                ;
8573                                ; Output to the AAF01-A
8574                                ;
8575 074144  012705  003040          170$:   MOV     #BUFOUT,R5              ; get the address of the output buffer
8576
8577 074150  012777  020000  106456  180$:   MOV     #CTA,@DRXDBR            ; address control word 0
8578 074156  052777  000001  106442          BIS     #FNCTO,@DRXSCR          ; in CTA register
```

TEST 27 - Loopback Test using IEX11-A Interface.

```
8579 074164  013702  002224                    MOV     FSTCHN,R2               ; get first selected channel
8580 074170  052702  030000                    BIS     #CWR,R2                 ; for CWR register in R2
8581 074174  010277  106434                    MOV     R2,@DRXDBR              ; set up to output on selected channel
8582 074200  052777  000001  106420            BIS     #FNCTO,@DRXSCR          ; transfer to CWR
8583 074206  011577  106422                    MOV     (R5),@DRXDBR            ; output the data
8584 074212  052777  000001  106406            BIS     #FNCTO,@DRXSCR          ; to the DBF register
8585                                    ;
8586                                    ; Wait for the conversion to complete
8587                                    ;
8588 074220  013701  002230                    MOV     CTIME,R1                ; get the conversion time
8589 074224  005002                            CLR     R2                      ; high byte is zero
8590 074226  004737  030072                    JSR     PC,DIV                  ; divide by
8591 074232  000175                            125.                            ; 125
8592
8593 074234  005701                    190$:   TST     R1                      ; loop until R1/R2 is zero
8594 074236  001404                            BEQ     210$                    ;
8595 074240  005301                    200$:   DEC     R1                      ;
8596 074242  004737  026114                    JSR     PC,WT25                 ; wait 25 microseconds
8597 074246  000772                            BR      190$                    ;
8598 074250  005702                    210$:   TST     R2                      ;
8599 074252  001402                            BEQ     220$                    ; total timeout = number of conversions
8600 074254  005302                            DEC     R2                      ;                 * conversion time * 2
8601 074256  000770                            BR      200$                    ;
8602                                    ;
8603                                    ; Check that the ACS register is correct
8604                                    ;
8605 074260  012737  010000  003020    220$:   MOV     #ACS,GOOD               ; get expected ACS contents
8606 074266  012777  010000  106340            MOV     #ACS,@DRXDBR            ; address the ACS register
8607 074274  017737  106334  003022            MOV     @DRXDBR,BAD             ; and get the actual contents
8608 074302  013700  003022                    MOV     BAD,R0                  ; get the contents into R0
8609 074306  042700  171777                    BIC     #^C<UNI!CMP>,R0         ; and isolate uncertain bits
8610 074312  050037  003020                    BIS     R0,GOOD                 ; set them in GOOD if they are set
8611 074316  023737  003022  003020            CMP     BAD,GOOD                ; contents correct ?
8612 074324  004737  026306                    CALL    INSERT                  ; skip branch if error insert selected
8613 074330  001404                            BEQ     230$                    ; branch if ACS is OK
8614 074332                                    ERRSOFT 2701,E2701,EGB          ; else print "ACS INCORRECT"
     074332  104457                                                                    TRAP    C$ERSOFT
     074334  005215                                                                    .WORD   2701
     074336  075241                                                                    .WORD   E2701
     074340  023416                                                                    .WORD   EGB
8615 074342                            230$:   CKLOOP                          ; branch to BGNSEG if loop on error set
     074342  104406                                                                    TRAP    C$CLP1
8616                                    ;
8617                                    ; Check that the loopback data is within tolerance
8618                                    ;
8619 074344  004737  031016                    JSR     PC,IEXIN                ; get looped back value
8620 074350  010103                            MOV     R1,R3                   ; high word in R3
8621 074352  010204                            MOV     R2,R4                   ; low word in R4
8622 074354  005737  032140                    TST     ERFLA                   ; error on input ?
8623 074360  004737  026306                    CALL    INSERT                  ; skip branch if error insert selected
8624 074364  001404                            BEQ     240$                    ; branch if no input error
8625 074366                                    ERRSOFT 2702,E2702,EN           ; else print "ERROR ON INPUT"
     074366  104457                                                                    TRAP    C$ERSOFT
     074370  005216                                                                    .WORD   2702
     074372  075320                                                                    .WORD   E2702
     074374  024030                                                                    .WORD   EN
8626 074376                            240$:   CKLOOP                          ; branch to BGNSEG if loop on error set
```

TEST 27 - Loopback Test using IEX11-A Interface.

```
        074376  104406                                                          TRAP    C$CLP1
  8627
  8628 074400  011501                          MOV     (R5),R1                ; get current output pattern
  8629 074402  004737  026352                  JSR     PC,DACON               ; convert to analogue value in R1 & R2
  8630
  8631 074406                                  PUSH$   <R1,R2>                ; save the good values
  8632 074412  160416                          SUB     R4,(SP)                ; subtract low actual from low good
  8633 074414  002004                          BGE     250$                   ; branch if no carry
  8634 074416  062716  001750                  ADD     #1000.,(SP)            ; else add 1000
  8635 074422  005366  000002                  DEC     2(SP)                  ; from high word
  8636 074426  160366  000002          250$:   SUB     R3,2(SP)               ; subtract high actual from high good
  8637 074432  103013                          BCC     270$                   ; if no carry, branch
  8638 074434  022626                          CMP     (SP)+,(SP)+            ; else clear the stack
  8639 074436                                  PUSH$   <R3,R4>                ; and put actual values on the stack
  8640 074442  160216                          SUB     R2,(SP)                ; subtract low good from low actual
  8641 074444  002004                          BGE     260$                   ; branch if no carry
  8642 074446  062716  001750                  ADD     #1000.,(SP)            ; else add 1000
  8643 074452  005366  000002                  DEC     2(SP)                  ; from high word
  8644 074456  160166  000002          260$:   SUB     R1,2(SP)               ; subtract high good from high actual
  8645
  8646                                                                        ; difference between good and actual
  8647                                                                        ; is now on the stack
  8648 074462  026637  000002  074704  270$:   CMP     2(SP),LTOL1            ; compare difference and tolerance high
  8649 074470  004737  026306                  CALL    INSERT                 ; skip branch if error insert selected
  8650 074474  002412                          BLT     290$                   ; branch if difference is low enough
  8651 074476  003005                          BGT     280$                   ; report error if difference too high
  8652 074500  021637  074706                  CMP     (SP),LTOL2             ; if equal, must compare the low words
  8653 074504  004737  0263C6                  CALL    INSERT                 ; skip branch if error insert selected
  8654 074510  003404                          BLE     290$                   ; branch if difference is low enough
  8655 074512                          280$:   ERRSOFT 2703,E2703,EGBDEC      ; else, print "DATA NOT WITHIN TOLERANCE"
       074512  104457                                                                  TRAP    C$ERSOFT
       074514  005217                                                                  .WORD   2703
       074516  075347                                                                  .WORD   E2703
       074520  024066                                                                  .WORD   EGBDEC
  8656 074522                          290$:   CKLOOP                         ; branch to BGNSEG if loop on error set
       074522  104406                                                                  TRAP    C$CLP1
  8657 074524  026637  000002  074712          CMP     2(SP),HDEV             ; compare deviation high word with highest so far ?
  8658 074532  002411                          BLT     310$                   ; if lower, don't do anything
  8659 074534  003003                          BGT     300$                   ; if higher, save the new value
  8660 074536  021637  074714                  CMP     (SP),LDEV              ; if the same, compare the low words
  8661 074542  003405                          BLE     310$                   ; if lower or equal, do nothing
  8662 074544  016637  000002  074712  300$:   MOV     2(SP),HDEV             ; save the new deviation high
  8663 074552  011637  074714                  MOV     (SP),LDEV              ; and low words
  8664 074556  022626                  310$:   CMP     (SP)+,(SP)+            ; tidy up the stack
  8665
  8666 074560  005725                          TST     (R5)+                  ; point to next output value
  8667 074562  020527  023040                  CMP     R5,#BUFOUT+8192.       ; all values in buffer output ?
  8668 074566  101002                          BHI     320$                   ; if yes, branch
  8669 074570  000137  074150                  JMP     180$                   ; else jump back
  8670 074574                          320$:   ENDSEG
       074574                                                                                  10004$:
       074574  104405                                                                  TRAP    C$ESEG
  8671
  8672 074576  013701  074712          330$:   MOV     HDEV,R1                ; save the highest deviation high word
  8673 074602  013702  074714                  MOV     LDEV,R2                ; and low word
  8674 074606                                  PRINTF  #DEV1                  ; print "MAXIMUM DEVIATION = "
       074606  012746  075425                                                          MOV     #DEV1,-(SP)
```

TEST 27 - Loopback Test using IEX11-A Interface.

```
        074612  012746  000001                                                  MOV     #1,-(SP)
        074616  010600                                                          MOV     SP,R0
        074620  104417                                                          TRAP    C$PNTF
        074622  062706  000004                                                  ADD     #4,SP
  8675  074626  004737  027640                  JSR     PC,DECOUT          ; xxx.xxx
  8676  074632                                 PRINTF  #DEV2              ; "MILLIVOLTS"
        074632  012746  075456                                                  MOV     #DEV2,-(SP)
        074636  012746  000001                                                  MOV     #1,-(SP)
        074642  010600                                                          MOV     SP,R0
        074644  104417                                                          TRAP    C$PNTF
        074646  062706  000004                                                  ADD     #4,SP
  8677
  8678  074652  005737  002234                  TST     QVMODE             ; is quick verify mode selected ?
  8679  074656  001010                          BNE     340$               ; if yes, exit test
  8680  074660  005237  003032                  INC     ITRCNT             ; else, increment iteration counter
  8681  074664  023727  003032  000001          CMP     ITRCNT,#1          ; iterations completed ?
  8682  074672  001402                          BEQ     340$               ; if yes, branch
  8683  074674  000137  073762                  JMP     90$                ; else do another iteration
  8684
  8685  074700                          340$:   EXIT TST
        074700  104432                                                          TRAP    C$EXIT
        074702  000574                                                          .WORD   L10064-.
  8686
  8687  074704  000012                  LTOL1:  .WORD   10.                 ; store for tolerance high word
  8688  074706  000000                  LTOL2:  .WORD   0                   ; store for tolerance low word
  8689  074710  000000                  UPSW:   .WORD   0                   ; ramp up or down switch
  8690  074712  000000                  HDEV:   .WORD   0                   ; deviation high word
  8691  074714  000000                  LDEV:   .WORD   0                   ; deviation low word
  8692
  8693                                           .NLIST  BEX
  8694
  8695  074716      060     056     060  N271:   .ASCIZ  /0.0       /         ; store for first input value string
  8696  074731      061     060     060  N272:   .ASCIZ  /10000.0   /         ; store for second input value string
  8697  074744      061     060     056  N27T:   .ASCIZ  /10.0      /         ; store for tolerance input string
  8698
  8699  074757      045     123     062  TSHD27::.ASCIZ  /%S2%ALOOPBACK TEST USING IEX11-A INTERFACE%N/
  8700  075034      045     123     062  TSNS27::.ASCIZ  /%S2%ALOOPBACK TEST USING IEX11-A - NOT SELECTED%N/
  8701
  8702  075116      111     105     130  T270:   .ASCIZ  /IEX11-A ADDRESS/
  8703  075136      114     117     117  T271:   .ASCIZ  /LOOPBACK TOLERANCE IN MILLIVOLTS/
  8704  075177      111     105     130  E2700:  .ASCIZ  /IEX11-A REGISTER ADDRESSING ERROR/
  8705  075241      101     103     123  E2701:  .ASCIZ  /ACS REGISTER INCORRECT AFTER OUTPUT TO IEX11-A/
  8706  075320      105     122     122  E2702:  .ASCIZ  /ERROR ON IEX11-A INPUT/
  8707  075347      114     117     117  E2703:  .ASCIZ  /LOOP BACK DATA NOT WITHIN SPECIFIED TOLERANCE/
  8708
  8709  075425      045     116     045  DEV1:   .ASCIZ  /%N%AMAXIMUM DEVIATION = /
  8710  075456      045     101     040  DEV2:   .ASCIZ  /%A MILLIVOLTS%N/
  8711
  8712                                           .LIST   BEX
  8713                                           .EVEN
  8714
  8715  075476                                   ENDTST
        075476                                                          L10064:
        075476  104401                                                          TRAP    C$ETST
  8716
```

```
                        .SBTTL  TEST 28 - Direct Read/write Test.

                        ;++
                        ; This allows the registers of the AAF01-A to be read from or
                        ; written to (where the hardware allows). Multiplexing through the
                        ; DRX11-C to the AAF01-A and further to the control table is
                        ; performed by the program.
                        ;
                        ; Normally, the program will use unit 0. However, any unit can be
                        ; selected by starting the test with the /UNIT switch (see section
                        ; 2.2 of the user documemtation).
                        ;
                        ; Registers are specified using 3 or 4 character mnemonics. After
                        ; a register has been read and if required a new value input,
                        ; another register will automatically be opened according to the
                        ; sequence shown below.
                        ;
                        ;  -> BAR1      SCR     -> DBF    -> control word addresed by CTA
                        ; !   WCR1      COR    !   ACS   !        next control word
                        ; !   BAR2      ADR    !   PCR   !        next control word
                        ; !   WCR2      DBR    !   CTA   !               !
                        ; !   WCO              !         !               v
                        ; !   ACO        !     !    !    !        last control word
                        ; !                -----      ----
                        ; !    !                          !               !
                        ;  ----                           !               !
                        ;                                  ---------------
                        ;                                  !             !
                        ;                                   ---------------
                        ;
                        ; To exit from any sequence, "control Z" can be typed to return to
                        ; the "REG" prompt. Typing "CONTROL C" returns the user to the
                        ; supervisor. The following example shows how the test can be
                        ; used.
                        ;--
```

TEST 28 - Direct Read/write Test.

```
                                        BGNTST
                                                                T28::
                                        RFLAGS  R0              ; read operator flags
                                                                        TRAP    C$RFLA
                                        BIT     #PNT,R0         ; print test headers ?
                                        BEQ     20$             ; if not, branch
                                        TST     STAFLG          ; was test specifically selected ?
                                        BEQ     10$             ; if not, branch
                                        PRINTF  #TSHD28         ; else, print test header
                                                                        MOV     #TSHD28,-(SP)
                                                                        MOV     #1,-(SP)
                                                                        MOV     SP,R0
                                                                        TRAP    C$PNTF
                                                                        ADD     #4,SP
                                        BR      ASKREG          ; and start the test
                                10$:    PRINTF  #TSNS28         ; print "TEST NOT SELECTED"
                                                                        MOV     #TSNS28,-(SP)
                                                                        MOV     #1,-(SP)
                                                                        MOV     SP,R0
                                                                        TRAP    C$PNTF
                                                                        ADD     #4,SP
                                        EXIT TST                ; and skip the test
                                                                        TRAP    C$EXIT
                                                                        .WORD   L10065-.
                                20$:    TST     STAFLG          ; was test specifically selected ?
                                        BNE     ASKREG          ; if yes, branch
                                        EXIT TST                ; else skip the test
                                                                        TRAP    C$EXIT
                                                                        .WORD   L10065-.

                                ASKREG: GMANID  REGQ,REG,A,377,1,4,YES  ; prompt for a register
                                                                        TRAP    C$GMAN
                                                                        BR      10000$
                                                                        .WORD   REG
                                                                        .WORD   T$CODE
                                                                        .WORD   REGQ
                                                                        .WORD   377
                                                                        .WORD   T$LOLIM
                                                                        .WORD   T$HILIM
                                                                10000$:
                                        MOV     #REGTAB,R1      ; get the table address
                                10$:    CMP     REG,(R1)        ; compare 1st half of register mnemonic
                                        BEQ     30$             ; branch if match
                                20$:    ADD     #12,R1          ; else go to next line of table
                                        CMP     R1,#TABEND      ; at the end of the table ?
                                        BEQ     ASKREG          ; if yes, ask for the register again
                                        BR      10$             ; else check the next mnemonic
                                30$:    CMP     REG+2,2(R1)     ; compare 2nd half of register mnemonic
                                        BNE     20$             ; if no match, keep looking
                                        JMP     @6(R1)          ; else jump to service address

                                OPBAR1: MOV     @DRXCOR,R2      ; save the current COR contents
                                10$:    MOV     10(R1),@DRXCOR  ; address the selected register
                                        MOV     @DRXADR,CONS    ; save the contents
                                        MOV     R2,@DRXCOR      ; restore the original COR contents
                                        MOV     (R1),REG        ; save the new register mnemonic
```

TEST 28 - Direct Read/write Test.

```
   8784 075710  016137  000002  076514          MOV     2(R1),REG+2             ; for printing
   8785 075716                                  GMANID  REG,CONS,0,-1,0,-1,YES  ; accept new input
        075716  104443                                                          TRAP    C$GMAN
        075720  000406                                                          BR      10001$
        075722  076510                                                          .WORD   CONS
        075724  000032                                                          .WORD   T$CODE
        075726  076512                                                          .WORD   REG
        075730  177777                                                          .WORD   -1
        075732  000000                                                          .WORD   T$LOLIM
        075734  177777                                                          .WORD   T$HILIM
        075736                                                              10001$:
   8786 075736  013700  177562                  MOV     @#177562,R0             ; read the keyboard buffer
   8787 075742  042700  177600                  BIC     #177600,R0              ; isolate the character bits
   8788 075746  020027  000032                  CMP     R0,#32                  ; was control Z typed ?
   8789 075752  001711                          BEQ     ASKREG                  ; if yes, request new register
   8790 075754  032761  002000  000010          BIT     #2000,10(R1)            ; don't allow writing
   8791 075762  001006                          BNE     20$                     ; to the WCO and ACO registers
   8792 075764  016177  000010  104636          MOV     10(R1),@DRXCOR          ; address the selected register
   8793 075772  013777  076510  104632          MOV     CONS,@DRXADR            ; and write the new contents
   8794 076000  062701  000012          20$:    ADD     #12,R1                  ; go to next register in sequence
   8795 076004  020127  076614                  CMP     R1,#SCRTAB              ; out of this loop ?
   8796 076010  001325                          BNE     10$                     ; if not, open next register
   8797 076012  012701  076520                  MOV     #REGTAB,R1              ; else start the loop again
   8798 076016  000722                          BR      10$                     ; and open BAR1
   8799
   8800 076020  013702  002626          OPSCR:  MOV     DRXSCR,R2               ; get the SCR address
   8801 076024  066102  000010                  ADD     10(R1),R2               ; offset to the selected register
   8802 076030  011237  076510                  MOV     (R2),CONS               ; save the contents
   8803 076034  011137  076512                  MOV     (R1),REG                ; save the new register mnemonic
   8804 076040  016137  000002  076514          MOV     2(R1),REG+2             ; for printing
   8805 076046                                  GMANID  REG,CONS,0,-1,0,-1,YES  ; accept new input
        076046  104443                                                          TRAP    C$GMAN
        076050  000406                                                          BR      10002$
        076052  076510                                                          .WORD   CONS
        076054  000032                                                          .WORD   T$CODE
        076056  076512                                                          .WORD   REG
        076060  177777                                                          .WORD   -1
        076062  000000                                                          .WORD   T$LOLIM
        076064  177777                                                          .WORD   T$HILIM
        076066                                                              10002$:
   8806 076066  013700  177562                  MOV     @#177562,R0             ; read the keyboard buffer
   8807 076072  042700  177600                  BIC     #177600,R0              ; isolate the character bits
   8808 076076  020027  000032                  CMP     R0,#32                  ; was control Z typed ?
   8809 076102  001635                          BEQ     ASKREG                  ; if yes, request new register
   8810 076104  013712  076510                  MOV     CONS,(R2)               ; else write the new contents
   8811 076110  062701  000012                  ADD     #12,R1                  ; go to next register in sequence
   8812 076114  020127  076664                  CMP     R1,#DBFTAB              ; out of this loop ?
   8813 076120  001337                          BNE     OPSCR                   ; if not, open next register
   8814
   8815 076122  017702  104506          OPDBF:  MOV     @DRXDBR,R2              ; save the current DBR contents
   8816 076126  016177  000010  104500  10$:     MOV     10(R1),@DRXDBR          ; address the selected register
   8817 076134  017737  104474  076510          MOV     @DRXDBR,CONS            ; save the contents
   8818 076142  010277  104466                  MOV     R2,@DRXDBR              ; restore the original DBR contents
   8819 076146  042737  170000  076510          BIC     #170000,CONS            ; remove the subaddress
   8820 076154  011137  076512                  MOV     (R1),REG                ; save the new register mnemonic
   8821 076160  016137  000002  076514          MOV     2(R1),REG+2             ; for printing
   8822 076166                                  GMANID  REG,CONS,0,-1,0,7777,YES ; accept new input
```

TEST 28 - Direct Read/write Test.

```
        076166  104443                                                          TRAP    C$GMAN
        076170  000406                                                          BR      10003$
        076172  076510                                                          .WORD   CONS
        076174  000032                                                          .WORD   T$CODE
        076176  076512                                                          .WORD   REG
        076200  177777                                                          .WORD   -1
        076202  000000                                                          .WORD   T$LOLIM
        076204  007777                                                          .WORD   T$HILIM
        076206                                                              10003$:
  8823  076206  013700  177562                MOV     @#177562,R0             ; read the keyboard buffer
  8824  076212  042700  177600                BIC     #177600,R0              ; isolate the character bits
  8825  076216  020027  000032                CMP     R0,#32                  ; was control Z typed ?
  8826  076222  001002                        BNE     20$                     ; if not, branch
  8827  076224  000137  075576                JMP     ASKREG                  ; else request new register
  8828  076230  056137  000010  076510  20$:  BIS     10(R1),CONS             ; set the subaddress bits
  8829  076236  016177  000010  104370        MOV     10(R1),@DRXDBR          ; address the selected register
  8830  076244  013777  076510  104362        MOV     CONS,@DRXDBR            ; and write the new contents
  8831  076252  052777  000001  104346        BIS     #FNCTO,@DRXSCR          ; transfer to the register
  8832  076260  062701  000012                ADD     #12,R1                  ; go to next register in sequence
  8833  076264  020127  076734                CMP     R1,#TABEND              ; out of this loop ?
  8834  076270  001316                        BNE     10$                     ; if not, open next register
  8835
  8836  076272  017702  104336        OPCWR:  MOV     @DRXDBR,R2              ; save the current DBR contents in R2
  8837  076276  012777  020000  104330        MOV     #CTA,@DRXDBR            ; address the CTA
  8838  076304  017703  104324                MOV     @DRXDBR,R3              ; get the current CTA contents in R3
  8839  076310  010304                        MOV     R3,R4                   ; and R4
  8840
  8841  076312  010477  104316        10$:    MOV     R4,@DRXDBR              ; address the CTA
  8842  076316  052777  000001  104302        BIS     #FNCTO,@DRXSCR          ; transfer to CTA
  8843  076324  012777  030000  104302        MOV     #CWR,@DRXDBR            ; read the control word
  8844  076332  017737  104276  076510        MOV     @DRXDBR,CONS            ; save the contents
  8845  076340  010377  104270                MOV     R3,@DRXDBR              ; restore the original CTA contents
  8846  076344  052777  000001  104254        BIS     #FNCTO,@DRXSCR          ; transfer to CTA
  8847  076352  010277  104256                MOV     R2,@DRXDBR              ; and the original DBR contents
  8848  076356  042737  177000  076510        BIC     #177000,CONS            ; remove the subaddress
  8849  076364  010405                        MOV     R4,R5                   ; get CTA contents
  8850  076366  042705  176000                BIC     #176000,R5              ; isolate the CWR number
  8851  076372  004737  077054                JSR     PC,SETCWR               ; set up "CWR xxxx" message
  8852  076376                                GMANID  CWROUT,CONS,0,-1,0,-1,YES ; accept new input
        076376  104443                                                          TRAP    C$GMAN
        076400  000406                                                          BR      10004$
        076402  076510                                                          .WORD   CONS
        076404  000032                                                          .WORD   T$CODE
        076406  077156                                                          .WORD   CWROUT
        076410  177777                                                          .WORD   -1
        076412  000000                                                          .WORD   T$LOLIM
        076414  177777                                                          .WORD   T$HILIM
        076416                                                              10004$:
  8853  076416  013700  177562                MOV     @#177562,R0             ; read the keyboard buffer
  8854  076422  042700  177600                BIC     #177600,R0              ; isolate the character bits
  8855  076426  020027  000032                CMP     R0,#32                  ; was control Z typed ?
  8856  076432  001002                        BNE     20$                     ; if not, branch
  8857  076434  000137  075576                JMP     ASKREG                  ; else request new register
  8858  076440  010477  104170        20$:    MOV     R4,@DRXDBR              ; reset the CTA to
  8859  076444  052777  000001  104154        BIS     #FNCTO,@DRXSCR          ; the current control word
  8860  076452  052737  030000  076510        BIS     #CWR,CONS               ; set the subaddress bits for the CWR
  8861  076460  013777  076510  104146        MOV     CONS,@DRXDBR            ; and write the new CWR contents
```

TEST 28 - Direct Read/write Test.

```
 8862 076466  052777  000001  104132          BIS     #FNCTO,@DRXSCR          ; transfer to CWR
 8863 076474  005204                          INC     R4                      ; get next CWR address
 8864 076476  042704  002000                  BIC     #2000,R4                ; modulo 1024
 8865 076502  000703                          BR      10$                     ; and open next CWR
 8866
 8867 076504                                  EXIT TST
      076504  104432                                                          TRAP    C$EXIT
      076506  000462                                                          .WORD   L10065-.
 8868
 8869 076510  000000                  CONS:   .WORD   0                       ; store for new register contents
 8870
 8871                                         .NLIST  BEX
 8872 076512     123     103     122  REG:    .ASCIZ  /SCR /                  ; store for operator input
 8873                                         .EVEN
 8874
 8875                                 ;        String          Service         Offset or
 8876                                 ;                        Address         subaddress
 8877
 8878 076520  040502  030522  000000  REGTAB: "BA,"R1,0,       OPBAR1,         BAR1
 8879 076532  041527  030522  000000          "WC,"R1,0,       OPBAR1,         WCR1
 8880 076544  040502  031122  000000          "BA,"R2,0,       OPBAR1,         BAR2
 8881 076556  041527  031122  000000          "WC,"R2,0,       OPBAR1,         WCR2
 8882 076570  041527  000117  000000          "WC,'O ,0,       OPBAR1,         WCO
 8883 076602  041501  000117  000000          "AC,'O ,0,       OPBAR1,         ACO
 8884
 8885 076614  041523  000122  000000  SCRTAB: "SC,'R ,0,       OPSCR,          0
 8886 076626  047503  000122  000000          "CO,'R ,0,       OPSCR,          2
 8887 076640  042101  000122  000000          "AD,'R ,0,       OPSCR,          4
 8888 076652  041104  000122  000000          "DB,'R ,0,       OPSCR,          6
 8889
 8890 076664  041104  000106  000000  DBFTAB: "DB,'F ,0,       OPDBF,          DBF
 8891 076676  041501  000123  000000          "AC,'S ,0,       OPDBF,          ACS
 8892 076710  041520  000122  000000          "PC,'R ,0,       OPDBF,          PCR
 8893 076722  052103  000101  000000          "CT,'A ,0,       OPDBF,          CTA
 8894 076734                          TABEND:
 8895
 8896 076734     045     123     062  TSHD28::.ASCIZ  \%S2%ADIRECT READ/WRITE TEST%N\
 8897 076772     045     123     062  TSNS28: .ASCIZ  \%S2%ADIRECT READ/WRITE TEST - NOT SELECTED%N\
 8898
 8899 077047     122     105     107  REGQ:   .ASCIZ  /REG/
 8900                                         .LIST   BEX
 8901                                         .EVEN
 8902
 8903                                 ;
 8904                                 ; Routine to set up string "CWROUT" to contain "CWR xxxx", where xxxx
 8905                                 ; is contained in R5. R5 is destroyed.
 8906                                 ;
 8907 077054  012700  077162          SETCWR: MOV     #CWRN,R0                ; get address of CWR number
 8908 077060  012710  030060                  MOV     #30060,(R0)             ; set it to "0000" ascii
 8909 077064  012760  030060  000002          MOV     #30060,2(R0)            ; ...
 8910 077072  020527  001750          10$:    CMP     R5,#1000.               ; number greater than 1000 ?
 8911 077076  002404                          BLT     20$                     ; if not, branch
 8912 077100  105210                          INCB    (R0)                    ; else increment the print digit
 8913 077102  162705  001750                  SUB     #1000.,R5               ; and subtract 1000 from the number
 8914 077106  000771                          BR      10$                     ; repeat until number is less than 1000
 8915 077110  005200                  20$:    INC     R0                      ; point to next print digit
 8916 077112  020527  000144          30$:    CMP     R5,#100.                ; number greater than 100 ?
```

TEST 28 - Direct Read/write Test.

```
 8917 077116  002404                          BLT     40$                     ; if not, branch
 8918 077120  105210                          INCB    (R0)                    ; else increment the print digit
 8919 077122  162705  000144                  SUB     #100.,R5                ; and subtract 100 from the number
 8920 077126  000771                          BR      30$                     ; repeat until number is less than 100
 8921 077130  005200                  40$:    INC     R0                      ; point to next print digit
 8922 077132  020527  000012          50$:    CMP     R5,#10.                 ; number greater than 10 ?
 8923 077136  002404                          BLT     60$                     ; if not, branch
 8924 077140  005210                          INC     (R0)                    ; else increment the print digit
 8925 077142  162705  000012                  SUB     #10.,R5                 ; and subtract 10 from the number
 8926 077146  000771                          BR      50$                     ; repeat until number is less than 10
 8927 077150  005200                  60$:    INC     R0                      ; point to next print digit
 8928 077152  150510                          BISB    R5,(R0)                 ; save last digit
 8929
 8930 077154  000207                          RTS     PC
 8931
 8932                                         .NLIST  BEX
 8933                                         .EVEN
 8934 077156     103     127     122  CWROUT: .ASCII  /CWR /
 8935 077162     170     170     170  CWRN:   .ASCIZ  /xxxx/
 8936                                         .LIST   BEX
 8937                                         .EVEN
 8938
 8939
 8940 077170                                 ENDTST
      077170                                                     L10065:
      077170  104401                                                    TRAP    C$ETST
 8941
 8942 077172                                 ENDMOD
 8943
```

TEST 28 - Direct Read/write Test.

```
                                .TITLE CLOSE SECTION

                                        BGNMOD

                                $PATCH::
                                        .BLKW   300

                                        .BLKB   400-<.&377>             ; SHIFT TO CORRECT FOR LSI BUG
                                        LASTAD

                                                                                .EVEN
100000  100014                                                                  .WORD T$FREE
100002  000004                                                                  .WORD T$SIZE
100004                          L$LAST::
                                        ENDMOD
```

TEST 28 - Direct Read/write Test.

```
 9006
 9007
 9008
 9021
 9022 100004                                  BGNSETUP        1.
 9023 100004                                  BGNPTAB
      100004  000000                                                    .WORD   0
      100006  000002                                                    .WORD   L10070-./2-1
      100010                                                        L10066:
 9024 100010  164200  000270                 .WORD   164200,270
 9025 100014                                  ENDPTAB
      100014                                                        L10070:
 9026 100014                                  ENDSETUP
 9027         000001                  .END
```

Symbol table

AC0	= 002400 G	BI8	065163	C$GMAN	= 000043	DUMP	025124 G	E1502	052424
ACS	= 010000 G	BI8A	065262	C$GPHR	= 000042	EAAAF	023726 G	E1600	053261
ADCON	026510 G	BLOCK	072050	C$GPLO	= 000030	ECE	= 000004 G	E1601	053340
ADDST	072034	BOE	= 000400 G	C$GPRI	= 000040	ECGB	023514 G	E1602	053406
ADFADD	002650 G	BUF	= 100000 G	C$INIT	= 000011	ECHGB	023764 G	E1700	054431
ADFADR	002644 G	BUFIN	023040 G	C$INLP	= 000020	ECHK	023404 G	E1701	054474
ADFCOR	002642 G	BUFOUT	003040 G	C$MANI	= 000050	ECNT	002710 G	E1702	054537
ADFDBR	002646 G	BURST	= 004000 G	C$MEM	= 000031	EF.CON	= 000036 G	E1703	054605
ADFIN	030522 G	BVALS	066542	C$MSG	= 000023	EF.NEW	= 000035 G	E1800	055443
ADFSCR	002640 G	CHAN	072046	C$OPEN	= 000034	EF.PWR	= 000034 G	E1801	055522
ADR	= 000020 G	CHANGE	072036	C$PNTB	= 000014	EF.RES	= 000037 G	E1802	055570
ALT	= 000100 G	CHAR	032704	C$PNTF	= 000017	EF.STA	= 000040 G	E1900	056670
ALTST	072042	CHKMAX	024760 G	C$PNTS	= 000016	EGB	023416 G	E1901	056727
ANSWER	064034	CLINT	032214 G	C$PNTX	= 000015	EGBCC	023622 G	E1902	057012
ASK	032234	CLKFLG	035132	C$QIO	= 000377	EGBDEC	024066 G	E1903	057076
ASKREG	075576	CMP	= 002000 G	C$RDBU	= 000007	EIGB	023560 G	E200	036404
ASSEMB	= 000010	CNSAVE	032070	C$REFG	= 000047	EMG1	024234	E2000	060011
ATT	= 100000 G	CNT25	026132 G	C$RESE	= 000033	EMG2	024303	E2001	060060
BAD	003022 G	CNT25M	026126 G	C$REVI	= 000003	EMG3	024347	E201	036475
BAR1	= 000000 G	CNT500	026130 G	C$RFLA	= 000021	EMG4	024443	E2100	060750
BAR2	= 001000 G	CONS	076510	C$RPT	= 000025	EMG5	024526	E2200	061644
BICAL	064147	CONT	033740	C$SEFG	= 000046	EMG6	024555	E2201	061726
BIERR	073105	COUT	= 000400 G	C$SPRI	= 000041	EMG7	024610	E2300	062613
BIMES	036644	CRLF	026134 G	C$SVEC	= 000037	EMG8	024657	E2301	062702
BIT0	= 000001 G	CTA	= 020000 G	C$TPRI	= 000013	EMG9A	024704	E2500	067044
BIT00	= 000001 G	CTIME	002230 G	DACON	026352 G	EMG9B	024714	E2501	067142
BIT01	= 000002 G	CWR	= 030000 G	DBE	= 000100 G	EMG9C	024740	E2600	072372
BIT02	= 000004 G	CWRN	077162	DBF	= 000000 G	EN	024030 G	E2601	072442
BIT03	= 000010 G	CWROUT	077156	DBFTAB	076664	END	034132	E2602	072504
BIT04	= 000020 G	C$AU	= 000052	DCM	= 000020 G	END1	034244	E2603	072531
BIT05	= 000040 G	C$AUTO	= 000061	DECEX	027750	EOC	= 000200 G	E2604	072554
BIT06	= 000100 G	C$BRK	= 000022	DECIN	026710 G	ERDNR	023460 G	E2605	072622
BIT07	= 000200 G	C$BSEG	= 000004	DECIN1	027471	ERFLA	032140	E2606	072671
BIT08	= 000400 G	C$BSUB	= 000002	DECIN2	027527	ESA	023670 G	E2607	073016
BIT09	= 001000 G	C$CEFG	= 000045	DECIN3	027610	EVL	= 000004 G	E2700	075177
BIT1	= 000002 G	C$CLCK	= 000062	DECOUT	027640 G	EXTCLK	031014	E2701	075241
BIT10	= 002000 G	C$CLEA	= 000012	DECO1	027756	E$END	= 002100	E2702	075320
BIT11	= 004000 G	C$CLOS	= 000035	DECO2	027762	E$LOAD	= 000035	E2703	075347
BIT12	= 010000 G	C$CLP1	= 000006	DECO3	027766	E100	035731	E300	037377
BIT13	= 020000 G	C$CVEC	= 000036	DEV1	075425	E1000	045243	E301	037444
BIT14	= 040000 G	C$DCLN	= 000044	DEV2	075456	E1001	045322	E302	037515
BIT15	= 100000 G	C$DODU	= 000051	DFPTBL	002216 G	E1002	045371	E400	040211
BIT2	= 000004 G	C$DRPT	= 000024	DIAGMC	= 000000	E1100	046227	E401	040256
BIT3	= 000010 G	C$DU	= 000053	DIV	030072 G	E1101	046305	E402	040327
BIT4	= 000020 G	C$EDIT	= 000003	DIV1	030102 G	E1102	046353	E500	041055
BIT5	= 000040 G	C$ERDF	= 000055	DMP1	025574	E1200	047211	E501	041120
BIT6	= 000100 G	C$ERHR	= 000056	DMP2	025652	E1201	047270	E502	041167
BIT7	= 000200 G	C$ERRO	= 000060	DMP3	025744	E1202	047336	E600	041776
BIT8	= 000400 G	C$ERSF	= 000054	DMP4	026036	E1300	050214	E601	042047
BIT9	= 001000 G	C$ERSO	= 000057	DROPD	035416	E1301	050275	E700	042606
BI1	064225	C$ESCA	= 000010	DROPED	002750 G	E1302	050346	E701	042654
BI2	064324	C$ESEG	= 000005	DRXADR	002632 G	E1400	051311	E702	042727
BI3	064401	C$ESUB	= 000003	DRXCOR	002630 G	E1401	051354	E800	043507
BI4	064503	C$ETST	= 000001	DRXDBR	002634 G	E1402	051417	E801	043553
BI5	064605	C$EXIT	= 000032	DRXSCR	002626 G	E1403	051453	E802	043611
BI6	064764	C$GETB	= 000026	DRXVEC	002636 G	E1500	052277	E900	044351
BI7	065061	C$GETW	= 000027	DSR	= 000020 G	E1501	052356	E901	044421

Symbol table

```
FCHIN   031006          G6      002522          LASTFA  026350          L$RPT   032234 G        L10060  062772
FNCT0 = 000001 G        G7      002544          LASTV   072044          L$SOFT  002334 G        L10061  065360
FNCT1 = 000002 G        G8      002566          LCHIN   031010          L$SPC   002056 G        L10062  067224
FNCT2 = 000004 G        HDEV    074712          LCLOCK  034372          L$SPCP  002020 G        L10063  073116
FNCT3 = 000010 G        HDIV    030310          LDEV    074714          L$SPTP  002024 G        L10064  075476
FSTCHN  002224 G        HEL     032304          LDIV    030312          L$STA   002030 G        L10065  077170
FVAL    003034 G        HELP  = 000000          LF      026156          L$SW    002224 G        L10066  100010
FVALQ   023236 G        HMUL    030314          LMUL    030316          L$TEST  002114 G        L10070  100014
F$AU  = 000015          HOE   = 100000 G        LOE   = 040000 G        L$TIML  002014 G        MAINT = 010000 G
F$AUTO= 000020          HPNR1   032134          LOT   = 000010 G        L$UNIT  002012 G        MET   = 000002 G
F$BGN = 000040          HPNR2   032136          LRES    030070          L10000  002222          MNT   = 001000 G
F$CLEA= 000007          HPPS1   031442          LSTCHN  002226 G        L10001  002240          MODE    003026 G
F$DU  = 000016          HPSAVE  032024          LTOL1   074704          L10002  002262          MSG26   072744
F$END = 000041          HRES    030066          LTOL2   074706          L10003  002420          MUL     027776 G
F$HARD= 000004          IBE   = 010000 G        L$ACP   002110 G        L10004  023414          MUL1    030006 G
F$HW  = 000013          IDU   = 000040 G        L$APT   002036 G        L10005  023456          NCHANS  066540
F$INIT= 000006          IE    = 000100 G        L$AU    035450 G        L10006  023512          NCONS   031012
F$JMP = 000050          IER   = 020000 G        L$AUT   002070 G        L10007  023556          NERRS   025062
F$MOD = 000000          IEXADD  032022 G        L$AUTO  035204 G        L10010  023620          NEWST   033730
F$MSG = 000011          IEXBCR= 000014          L$CCP   002106 G        L10011  023666          NEX   = 040000 G
F$PROT= 000021          IEXBDR= 000012          L$CLEA  035266 G        L10012  023724          NH    = 000035
F$PWR = 000017          IEXCSR= 000010          L$CO    002032 G        L10013  023762          NO      033516
F$RPT = 000012          IEXICR= 000004          L$DEPO  002011 G        L10014  024026          NOCLOK  034644
F$SEG = 000003          IEXIDR= 000006          L$DESC  023212 G        L10015  024064          NR1     027342
F$SOFT= 000005          IEXIIR= 000002          L$DESP  002076 G        L10016  024202          NR2     027344
F$SRV = 000010          IEXIN   031016 G        L$DEVP  002060 G        L10017  032212          NTESTS= 000034
F$SUB = 000002          IEXISR= 000000          L$DISP  002124 G        L10020  032214          NWORDS  030520
F$SW  = 000014          IEXMCR= 000016          L$DLY   002116 G        L10021  032222          NXM     032204 G
F$TEST= 000001          INIUUT  033770          L$DTP   002040 G        L10022  032232          NXMFLG  002770 G
GETNUM  027361          INOUT = 040000 G        L$DTYP  002034 G        L10023  033522          NXTUUT  034002
GO    = 000001 G        INSERT  026306 G        L$DU    035362 G        L10025  034252          N251    066544
GOOBAD  024204          INT     032216 G        L$DUT   002072 G        L10026  035264          N252    066557
GOOD    003020 G        INTFLG  002772 G        L$DVTY  023202 G        L10027  035360          N261    072052
G$CNTO= 000200          INT1    032224 G        L$EF    002052 G        L10030  035446          N262    072065
G$DELM= 000372          ISR   = 000100 G        L$ENVI  002044 G        L10031  035460          N27T    074744
G$DISP= 000003          ITRCNT  003032 G        L$ETP   002102 G        L10032  035774          N271    074716
G$EXCP= 000400          IXE   = 004000 G        L$EXP1  002046 G        L10033  036716          N272    074731
G$MILI= 000002          I$AU  = 000041          L$EXP4  002064 G        L10034  037600          OCTOUT  023341 G
G$LOLI= 000001          I$AUTO= 000041          L$EXP5  002066 G        L10035  040412          OFF   = 020000 G
G$NO  = 000000          I$CLN = 000041          L$HARD  002242 G        L10036  041250          ONEFIL= 000001
G$OFFS= 000400          I$DU  = 000041          L$HIME  002120 G        L10037  042116          OPBAR1  075660
G$OFSI= 000376          I$HRD = 000041          L$HPCP  002016 G        L10040  043012          OPCWR   076272
G$PRMA= 000001          I$INIT= 000041          L$HPTP  002022 G        L10041  043650          OPDBF   076122
G$PRMD= 000002          I$MOD = 000041          L$HW    002216 G        L10042  044470          OPSCR   076020
G$PRML= 000000          I$MSG = 000041          L$ICP   002104 G        L10043  045440          O$APTS= 000000
G$RADA= 000140          I$PROT= 000040          L$INIT  033532 G        L10044  046422          O$AU  = 000001
G$RADB= 000000          I$PTAB= 000041          L$LADP  002026 G        L10045  047404          O$BGNR= 000001
G$RADD= 000040          I$PWR = 000041          L$LAST  100004 G        L10046  050420          O$BGNS= 000001
G$RADL= 000120          I$RPT = 000041          L$LOAD  002100 G        L10047  051510          O$DU  = 000001
G$RADO= 000020          I$SEG = 000041          L$LUN   002074 G        L10050  052472          O$ERRT= 000000
G$XFER= 000004          I$SETU= 000041          L$MREV  002050 G        L10051  053454          O$GNSW= 000001
G$YES = 000010          I$SFT = 000041          L$NAME  002000 G        L10052  054654          O$POIN= 000001
G1      002262          I$SRV = 000041          L$PRIO  002042 G        L10053  055636          O$SETU= 000001
G2      002302          I$SUB = 000041          L$PROT  033524 G        L10054  057162          PADD    003024 G
G3      002420          I$TST = 000041          L$PRT   002112 G        L10055  060142          PAT     072040
G4      002436          J$JMP = 000167          L$REPP  002062 G        L10056  061022          PCR   = 040000 G
G5      002453          KLINT   034506          L$REV   002010 G        L10057  062000          PFLAG1  002774 G
```

Symbol table

```
PM250   066746          SFPTBL  002224 G        TSHD27  074757 G        T$TSTS= 000001          T25     065362 G
PNT   = 001000 G        SNUM    027346          TSHD28  076734 G        T$$AU = 010031          T255    066705
PREX    032700          STAFLG  003030 G        TSHD3   037344 G        T$$AUT= 010026          T26     067226 G
PRI   = 002000 G        START   033532          TSHD4   040156 G        T$$CLE= 010027          T261    072221
PRI00 = 000000 G        STAT    032514          TSHD5   041016 G        T$$DAT= 010070          T262    072247
PRI01 = 000040 G        STAT0 = 000400 G        TSHD6   041734 G        T$$DU = 010030          T263    072302
PRI02 = 000100 G        STAT1 = 001000 G        TSHD7   042534 G        T$$HAR= 010002          T264    072345
PRI03 = 000140 G        STAT2 = 002000 G        TSHD8   043452 G        T$$HW = 010000          T27     073120 G
PRI04 = 000200 G        STAT3 = 004000 G        TSHD9   044310 G        T$$INI= 010025          T270    075116
PRI05 = 000240 G        SVAL    003036 G        TSNS24  064066 G        T$$MSG= 010016          T271    075136
PRI06 = 000300 G        SVALQ   023270 G        TSNS25  066630          T$$PC = 000001          T28     075500 G
PRI07 = 000340 G        SVCGBL= 000000          TSNS26  072141 G        T$$PRO= 010024          T3      036720 G
PR1     033112          SVCINS= 000001          TSNS27  075034 G        T$$PTA= 010067          T4      037602 G
PR2     033133          SVCSUB= 000001          TSNS28  076772          T$$RPT= 010023          T5      040414 G
PR2A    033247          SVCTAG= 000001          TT      033004          T$$SEG= 010004          T6      041252 G
PR3     033303          SVCTST= 000001          TTINT   034776          T$$SOF= 010003          T7      042120 G
PR3A    033401          S$LSYM= 010000          T$ARGC= 000001          T$$SRV= 010022          T8      043014 G
PR4     033440          TABEND  076734          T$CODE= 000032          T$$SW = 010001          T9      043652 G
PR5     033471          TADS    032714          T$ERRN= 005217          T$$TES= 010065          UAM   = 000200 G
PWRFL   033756          TIMMSG  035134          T$EXCP= 000000          T1      035462 G        UNI   = 004000 G
QFLAG1  003014 G        TITLE   032346          T$FLAG= 000040          T10     044472 G        UNICAL  064705
QFLAG2  003016 G        TKB   = 177562          T$FREE= 100014          T10TAB  045164          UNIERR  073074
QVMODE  002234 G        TKS   = 177560          T$GMAN= 000000          T11     045442 G        UNIMES  036572
RA      032200 G        TNUM    033106          T$HILI= 177777          T11TAB  046134          UNITS   034250
RANDOM  032142 G        TOL     002232 G        T$LAST= 000001          T12     046424 G        UPSW    074710
RB      032202 G        TPB   = 177566          T$LOLI= 000000          T12TAB  047116          USCLOK  034600
RDUMP   002236 G        TPS   = 177564          T$LSYM= 010000          T13     047406 G        VBPTAB  023122 G
RDY   = 000200 G        TSHD1   035700 G        T$LTNO= 000034          T13TAB  050100          VUPTAB  023042 G
REG     076512          TSHD10  045214 G        T$NEST= 177777          T14     050422 G        WAIT  = 010000 G
REGQ    077047          TSHD11  046200 G        T$NSO = 000000          T15     051512 G        WCO   = 002000 G
REGTAB  076520          TSHD12  047162 G        T$NS1 = 000001          T15TAB  052204          WCR1  = 000400 G
RES   = 000040 G        TSHD13  050130 G        T$NS2 = 000003          T16     052474 G        WCR2  = 001400 G
RESTRT  033712          TSHD14  051252 G        T$PCNT= 000000          T16TAB  053166          WFLG    026210
RET     033050          TSHD15  052250 G        T$PTAB= 010067          T17     053456 G        WRDY    026162 G
RFLG    032712          TSHD16  053232 G        T$PTHV= 000001          T18     054656 G        WRDY1   026212
RNDOUT  023323 G        TSHD17  054402 G        T$PTNU= 000001          T18TAB  055350          WT      026120
RS      025542          TSHD18  055414 G        T$SAVL= 177777          T19     055640 G        WT25    026114 G
RUN   = 020000 G        TSHD19  056632 G        T$SEGL= 177777          T2      035776 G        WT25M   026100 G
SAVCNT  035050          TSHD2   036362 G        T$SEKO= 010004          T20     057164 G        WT500   026106 G
SBE   = 000010 G        TSHD20  057756 G        T$SIZE= 000004          T20TAB  057734          X$ALWA= 000000
SBR   = 000040 G        TSHD21  060722 G        T$SUBN= 000000          T21     060144 G        X$FALS= 000040
SCRTAB  076614          TSHD22  061602 G        T$TAGL= 177777          T22     061024 G        X$OFFS= 000400
SETCLK  034254          TSHD23  062544 G        T$TAGN= 010071          T23     062002 G        X$TRUE= 000020
SETCWR  077054          TSHD24  064036 G        T$TEMP= 000000          T23TA1  062424          YES     033512
SETEX   035130          TSHD25  066572 G        T$TEST= 000034          T23TA2  062474          $PATCH  077172 G
SETTAB  030320 G        TSHD26  072100 G        T$TSTM= 177777          T24     062774 G

. ABS.  100014    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 255
Work  file writes: 255
Size of work file: 29808 Words  ( 117 Pages)
```

E3

CLOSE SECTION MACRO V05.00 Friday 01-Feb-85 08:58 Page 109-4

SEQ

Symbol table

Size of core pool: 19990 Words (76 Pages)
Operating system: RSX-11M/PLUS

Elapsed time: 00:16:13.06
AAF01,AAF01/-SP=[50,200]SVC/ML,[7,106]AAF01.SRC