

The main body of the document is a large grid of small, illegible technical diagrams or data tables. Each cell in the grid appears to contain a small schematic or data set, possibly representing a functional test point (TST PT 1) for a device. The text within these cells is too small to read, but the overall layout is a dense, organized array of technical information.

4 *2 W
A ::
1
CVDHACO DHV11-M FUNC TST PART 1 MACRO Y05.02 Monday 01-Apr-85 07:47 Page 2
PROGRAM DOCUMENT

SEQ 000

.REM &

IDENTIFICATION

PRODUCT CODE: AC-T652C-MC
PRODUCT NAME: CVDHACO DHV11-M FUNC TST PART 1
PRODUCT DATE: 29 MARCH 1985
MAINTAINER: Bruce Ribolini - MK Diagnostics Group
AUTHOR: Bert Kleinschmidt
Tony Grimshaw
MODIFIED BY: Bert Kleinschmidt
Anthony Hart
Peter O'Neil

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1983, 1984, 1985 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

***** MODIFICATION HISTORY *****

Original release: 31-OCT-83 (EDITED 11-JUL-83)
Bert Kleinschmidt

Version B0 09-OCT-83 Bert Kleinschmidt
Fixed typographical errors.
Moved test from CVDHB (Part 2) into this program:
Old CVDHB (version A) tests 2 through 8 are
now New CVDHA (version B) tests 20 through 26.
Modified test 13 (Selftest Fail Test) to allow use of CVDHA
in manufacturing with version 0 DHV ROM code.

Version C0 17-JUL-84 Bert Kleinschmidt
Modified control of processor priority throughout the
program to maintain priorities of 5 or lower to allow
LTC interrupts. This guarantees a less than 2 second
response to a Break request while running under APT.

Version C0 9-SEP-84 PETER ONEIL
Modified cleanup code to turn off clock if it was turned on.

29-MAR-85 Howard Marshall
Did formal release of version CVDHACO.

TABLE OF CONTENTS

1.0 GENERAL PROGRAM CONSIDERATIONS 4

1.1 PROGRAM ABSTRACT 4

1.2 SYSTEM REQUIREMENTS 4

1.3 RELATED DOCUMENTS AND STANDARDS 4

1.4 DIAGNOSTIC HIERARCY PREREQUISITES 5

2.0 OPERATING INSTRUCTIONS 6

2.1 COMMANDS 6

2.2 SWITCHES 6

2.3 FLAGS 8

2.4 EXTENDED COMMAND SYNTAX 9

2.4.1 START COMMAND 9

2.4.1.1 Tests Switch (/TESTS:<TEST-LIST>) 9

2.4.1.2 Pass Switch (/PASS:<PASS-CNT>) 9

2.4.1.3 Flags Switch (/FLAGS:<FLAG LIST>) 9

2.4.1.4 End Of Pass Switch (/EOP:<INCR>) 9

2.4.1.5 Effect Of Start Command 9

2.4.2 Restart Command 11

2.4.2.1 Tests, Pass, And Flags Switches 11

2.4.2.2 Units Switch (/UNITS:<UNIT-LIST>) 11

2.4.2.3 Effect Of Restart Command 11

2.4.3 Continue Command 11

2.4.3.1 Flag Switch (/FLAGS:<FLAG-LIST>) 11

2.4.3.2 Effect Of Continue Command 11

2.4.4 Proceed Command 12

2.4.4.1 Flags Switch (/FLAGS:<FLAG-LIST>) 12

2.4.4.2 Effect Of Proceed Command 12

2.4.5 Add Command 12

2.4.6 EFFECT OF ADD COMMAND 12

2.4.7 Drop Command 13

2.4.8 EFFECT OF DROP COMMAND 13

2.4.9 Print Command 13

2.4.9.1 Effect Of Print Command 13

2.4.10 Display Command 13

2.4.10.1 Effect Of Display Command 13

2.4.11 Flags Command 13

2.4.11.1 Effect Of Flags Command 13

2.4.12 Zflags Command 14

2.4.13 Zflags Command 14

2.4.14 Control Characters 14

2.5 HARDWARE QUESTIONS 15

2.6 SOFTWARE QUESTIONS 15

2.7 EXTENDED P-TABLE DIALOGUE 16

2.8 QUICK START-UP PROCEDURE (XXDP*) 19

3.0 ERROR INFORMATION 19

3.1 TYPES OF ERROR MESSAGES 19

3.2 ERROR MESSAGES 20

4.0 PERFORMANCE AND PROGRESS REPORTS 21

5.0 TEST SUMMARIES 21

6.0 EXAMPLE ERROR FREE PASS 23

PROGRAM DOCUMENT

1.0 GENERAL PROGRAM CONSIDERATIONS

1.1 PROGRAM ABSTRACT

CVDHA is part one of the DHV11-M functional verification test. This part of the test verifies that the reset, register access, and interrupt functions of the board are functioning correctly.

THIS DIAGNOSTIC HAS BEEN WRITTEN FOR USE WITH THE DIAGNOSTIC RUNTIME SERVICES SOFTWARE (SUPERVISOR). THESE SERVICES PROVIDE THE INTERFACE TO THE OPERATOR AND TO THE SOFTWARE ENVIRONMENT. THIS PROGRAM CAN BE USED WITH XXDP+, ACT, APT, SLIDE AND PAPER TAPE. FOR A COMPLETE DESCRIPTION OF THE RUNTIME SERVICES, REFER TO THE XXDP+ USER'S MANUAL. THERE IS A BRIEF DESCRIPTION OF THE RUNTIME SERVICES IN THE OPERATING INSTRUCTIONS-COMMANDS OF THIS DOCUMENT.

1.2 SYSTEM REQUIREMENTS

The following hardware is required to run the DHV FVT:

- o LSI-11 processor with at least 32 Kbytes of RAM.
- o DHV11 boards installed on the Q-bus.
- o Appropriate program load device supporting XXDP+ media or a down-line loading system.

1.3 RELATED DOCUMENTS AND STANDARDS

- o DHV11-M Hardware Manual - This manual describes the functions and uses of the DHV11-M device.
- o XXDP+ User's Manual - Describes the running of diagnostics under the XXDP+ monitor.

PROGRAM DOCUMENT

1.4 DIAGNOSTIC HIERARCY PREREQUISITES

The LSI 11 processor, the Q-BUS, the system memory, the console terminal, and the load media are assumed to have been tested and found working before this program's run.

PROGRAM DOCUMENT

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 +C)
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 STATISTICAL INFORMATION (IF IMPLEMENTED BY THE DIAGNOSTIC - SEE PERFORMANCE AND PROGRESS REPORTS SECTION OF THIS DOCUMENT)
DISPLAY	TYPE A LIST OF ALL DEVICE INFORMATION
FLAGS	TYPE THE STATE OF ALL FLAGS (SEE FLAGS SECTION)
ZFLAGS	CLEAR ALL FLAGS (SEE FLAGS SECTION)

A COMMAND CAN BE RECOGNIZED BY THE FIRST THREE CHARACTERS. SO YOU MAY, FOR EXAMPLE, TYPE "STA" INSTEAD OF "START". MORE INFORMATION CAN BE FOUND WITHIN THE SECTION LABELLED EXTENDED COMMAND SYNTAX

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 "DDDD".

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.

PROGRAM DOCUMENT

THIS LIST WILL CAUSE TESTS 1,5,7,8,9,10 TO BE RUN. ALL OTHER TESTS WILL NOT BE RUN. EXECUTE DDDDD PASSES (DDDDD = 1 TO 64000) SET SPECIFIED FLAGS. SEE THE FLAGS SECTION OF THIS DOCUMENT.

/PASS:DDDDD REPORT END OF PASS MESSAGE AFTER EVERY DDDDD PASSES ONLY. (DDDDD = 1 TO 64000)

/FLAGS:FLGS 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					

PROGRAM DOCUMENT

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
IBR*	INHIBIT ALL ERROR REPORTS EXCEPT FIRST LEVEL (FIRST LEVEL CONTAINS ERROR TYPE, NUMBER, PC, TEST AND UNIT)
IXR*	INHIBIT EXTENDED ERROR REPORTS (THOSE CALLED BY PRINTX MACRO'S)
PRI	DIRECT MESSAGES TO LINE PRINTER
PNT	PRINT TEST NUMBER AS TEST EXECUTES
BOE	"BELL" ON ERROR
UAM	UNATTENDED MODE (NO MANUAL INTERVENTION)
ISR	INHIBIT STATISTICAL REPORTS (DOES NOT APPLY TO DIAGNOSTICS WHICH DO NOT SUPPORT STATISTICAL REPORTING)
IDR	INHIBIT PROGRAM DROPPING OF UNITS
ADR	EXECUTE AUTODROP CODE
LOT	LOOP ON TEST
EVL	EXECUTE EVALUATION (ON DIAGNOSTICS WHICH HAVE EVALUATION SUPPORT)

*SEE THE ERROR INFORMATION SECTION OF THIS DOCUMENT.

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

PROGRAM DOCUMENT

2.4 EXTENDED COMMAND SYNTAX

2.4.1 START COMMAND -

```

*****
STA(RT)/TESTS:<TEST-LIST>/PASS:<PASS-CNT>/FLAGS:
<FLAG-LIST>/EOP:<INCR>
*****

```

2.4.1.1 Tests Switch (/TESTS:<TEST-LIST>) -

<TEST-LIST> Is a sequence of decimal numbers (1:2 etc.) or ranges of decimal numbers (1-5:8-10 etc.), seperated by colons, that specify the tests to be executed. Tests will be executed in numerical order regardless of the order of specification. The default is to execute all tests. On this and all switches, the angle brackets <> are punctuation used in the definition only, and are not to be typed by the operator. See example at end of "Effect of Start Command" section.

2.4.1.2 Pass Switch (/PASS:<PASS-CNT>) -

<PASS-CNT> Is a decimal number indicating the desired number of passes. A pass is defined as the execution of the full diagnostic (all selected tests). The default is non-ending execution. In this case, exit from the program is accomplished either by typing a control/C or by occurrence of an error with the halt on error flag being set. The exit is a return to command mode. See example at end of "Effect of Start Command" section.

2.4.1.3 Flags Switch (/FLAGS:<FLAG LIST>)

<FLAG-LIST> is a sequence of elements of the form <FLAG>, <FLAG=1>, or <FLAG=0>, separated by colons, where <FLAG> has one of the following values:

- HOE Halt on error, causing command mode to be entered when an error is encountered.
- LOE Loop on error, causing the diagnostic to loop continuously within the smallest defined block of coding (segment, subtest, or test) containing the error.
- IER Inhibit error reporting.
- IBE Inhibit basic error reports.
- IXE Inhibit extended error reports.
- PRI Direct all messages to a line printer.
- PNT Print number of test being executed.
- BOE Bell on error.
- UAM Run in unattended mode, bypassing manual

PROGRAM DOCUMENT

intervention.
 ISR Inhibit statistical reports.
 IDU Inhibit dropping of units by diagnostic.
 LOT Loop on test.

The flags named or equated to 1 are set, those equated to 0 are cleared. A flag not specified is cleared. If the flags switch is not given all flags are cleared. See example at end of "Effect of Start Command" section.

2.4.1.4 End Of Pass Switch (/EOP:<INCR>) -

<INCR> Is a decimal number indicating how often (in terms of passes) it is desired that the end of pass message be printed. The default is at the end of every pass. See example at end of "Effect of Start Command" section.

2.4.1.5 Effect Of Start Command -

The effect of the start command is to initiate the hardware parameter dialogue, the software parameter dialogue, the initialization questions, and then the diagnostic commences testing.

The hardware parameter dialogue commences with the question "# UNITS (D)?" to which the operator should reply with the number of units to be tested. Following this are the questions whereby the P-Tables themselves are built. Each P-Table is a core-resident table containing all the hardware information for one complete unit. Each question is followed by the response radix (D for decimal, B for binary, O for octal, L for Yes/No) in parentheses and the default value after the parentheses. For the actual Hardware P-Table questions see the "Hardware Parameters" section.

Following the hardware questions are the software questions to build the software tables, which define operating parameters of the diagnostic program. These Questions are described in the "Software Parameters" section.

EXAMPLE:

STA/TESTS:1:3-4:/PASS:3/FLAGS:IER:HOE=1

This command will cause three passes to be made, with each pass consisting of tests 1,3, and 4. There is no difference between saying <FLAG> and saying <FLAG=1>. The notation <FLAG=0> is meaningful only on a command other than start to clear a flag that was previously set. Note that on all commands only the first three letters are scanned.

PROGRAM DOCUMENT

2.4.2 Restart Command -

```

*****
RES(TART)/TESTS:<TEST-LIST>/PASS:<PASS-CNT>/FLAGS:
<FLAG-LIST>/UNITS:<UNIT-LIST>
*****

```

2.4.2.1 Tests, Pass, And Flags Switches -

<TEST-LIST>, <PASS-CNT>, and <FLAG-LIST> are as in the start command.

2.4.2.2 Units Switch (/UNITS:<UNIT-LIST>) - <UNIT-LIST> IS A SEQUENCE OF DECIMAL NUMBERS (0,1 ETC.) OR RANGES OF DECIMAL NUMBERS (0-5, 8-10 ETC.) THAT SPECIFY THE UNITS TO BE TESTED. THE NUMBERS ARE SEPARATED BY COLONS. THE NUMBERS MAY RANGE FROM 0 THRU N-1 (N IS THE NUMBER OF UNITS SPECIFIED IN THE PREVIOUS START COMMAND). THE NUMBER INDICATES THE POSITION OF THE P-TABLE AS THE DATA WAS ENTERED DURING THE HARDWARE DIAGLOGUE. THE UNITS WHICH ARE SELECTED MUST NOT HAVE BEEN DROPPED BY THE DROP COMMAND. SEE THE DISCUSSION OF ADD AND DROP COMMANDS BELOW. DEFAULT IS TO TEST ALL UNITS WHICH HAVE NOT BEEN DROPPED BY A DROP COMMAND.

2.4.2.3 Effect Of Restart Command -

The restart command differs from the start command in that the P-Tables from the previous start command (there must have been one) are used, instead of new ones being built. The software dialogue may optionally be reexecuted (operator will be asked). The command can be used after command mode has been reentered in any of the three normal ways: a) the requested number of passes have been made, b) an error was encountered with the halt on error flag set, or c) a control/C was entered by the operator.

2.4.3 Continue Command -

```

*****
CON(TINUE)/PASS:<PASS-CNT>/FLAGS:<FLAG-LIST>
*****

```

PROGRAM DOCUMENT

2.4.3.1 Flag Switch (/FLAGS:<FLAG-LIST>) -

<FLAG-LIST> Is same as in the start command, but unspecified flags retain their current value.

2.4.3.2 Effect Of Continue Command -

Continue must follow a start or restart, and command mode must have been entered due to a halt on error or a control/C. The effect of the command is to go to the beginning of the test that was being executed when the halt or control/C took place. Software dialogue may optionally be reexecuted. Hardware parameters may not be changed.

2.4.4 Proceed Command -

PRO(CEED)/FLAGS:<FLAG-LIST>

2.4.4.1 Flags Switch (/FLAGS:<FLAG-LIST>) -

<FLAG-LIST> Is as in the start command, but unspecified flags retain their current value.

2.4.4.2 Effect Of Proceed Command -

Proceed must follow a start, restart, or continue. Command mode must have been entered via a halt on error. The effect of the command is to begin execution at the location following the error call. Neither hardware nor software parameters may be altered.

2.4.5 Add Command -

ADD/UNITS:<UNIT-LIST>

2.4.6 EFFECT OF ADD COMMAND - THE UNITS SPECIFIED ARE ADDED TO THE TEST SEQUENCE. EACH UNIT MUST HAVE A P-TABLE IN MEMORY DUE TO AN EARLIER HARDWARE DIALOGUE. THIS COMMAND MUST BE FOLLOWED BY A RESTART OR CONTINUE. THE UNITS SWITCH MUST BE SPECIFIED. THE ADD COMMAND IS MEANINGFUL ONLY FOR UNITS THAT WERE PREVIOUSLY DROPPED.

PROGRAM DOCUMENT

2.4.7 Drop Command -

DRO(P)/UNITS:<UNIT-LIST>

2.4.8 EFFECT OF DROP COMMAND - THE UNITS SPECIFIED WILL BE DROPPED FROM TESTING. THE UNITS WILL BE RESELECTED ONLY BY THE EXECUTION OF AN ADD OR START COMMAND. THE UNITS SWITCH MUST BE ENTERED. THIS COMMAND MUST BE FOLLOWED BY A RESTART OR A CONTINUE COMMAND.

2.4.9 Print Command -

PRI(NT)

2.4.9.1 Effect Of Print Command - Error summary reporting is not implemented in this diagnostic, so this command has no effect.

2.4.10 Display Command -

DIS(PLAY)/UNITS:<UNIT-LIST>

2.4.10.1 Effect Of Display Command -

The hardware P-Tables for all units are printed in the format in which they were entered.

2.4.11 Flags Command -

FLA(GS)

2.4.11.1 Effect Of Flags Command -

The current settings of all flags are printed.

PROGRAM DOCUMENT

2.4.12 Zflags Command -

ZFL(AGS)

2.4.13 Zflags Command -

All flags are cleared.

2.4.14 Control Characters -

- C A control/C (C) entered during the execution of a diagnostic causes a return to command mode.

- Z A control/Z (Z) entered during one of the two operator dialogues-- hardware P-Table dialogue or software P-Table dialogue causes the defaults to be taken for the remainder of that dialogue.

- O A control/O (O) entered during the execution of a diagnostic causes all teletype output to be suppressed for the remainder of the diagnostic or until another control/O is typed, which restores normal teletype output.

2.5 HARDWARE QUESTIONS

WHEN A DIAGNOSTIC IS STARTED, THE RUNTIME SERVICES WILL PROMPT THE USER FOR HARDWARE INFORMATION BY TYPING "CHANGE HW (L) ?" YOU MUST ANSWER "Y" AFTER A START COMMAND UNLESS THE HARDWARE INFORMATION HAS BEEN "PRELOADED" USING THE SETUP UTILITY (SEE CHAPTER 6 OF THE XXDP+ USER'S MANUAL). WHEN YOU ANSWER THIS QUESTION WITH A "Y", THE RUNTIME SERVICES WILL ASK FOR THE NUMBER OF UNITS (IN DECIMAL). YOU WILL THEN BE ASKED THE FOLLOWING QUESTIONS FOR EACH UNIT:

1. CSR ADDRESS - This question requests the CSR address of the specified DHV11. The default answer for this question is the lowest address in the PDP-11 floating address space in which a DHV11-M can be placed (160460 Octal).
2. INTERRUPT VECTOR ADDRESS This question requests the interrupt vector address of the specified DHV11.
3. ACTIVE LINES BIT MAP - This question requests an octal bit map of the serial communication lines on the DHV11 which are being selected for testing. If the bit in the bit map is set which corresponds to a particular line (i.e. bit 3 for line 3) that line will be tested by the FVT.
4. BR Level This questions requests the interrupt BR level of the DHV11.

2.6 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 WILL GOVERN SOME DIAGNOSTIC SPECIFIC OPERATION MODES. YOU WILL BE PROMPTED BY "CHANGE SW (L) ?" IF YOU WISH TO CHANGE ANY PARAMETERS, ANSWER BY TYPING "Y". The following Software P-Table questions are asked by the program if the operator indicates that the Software Parameters are to be changed:

1. REPORT UNIT NUMBER AS EACH UNIT IS TESTED - This question asks whether the program should report the number of the unit which it is testing as it begins to test each unit.
2. NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE - This question asks for the number of data errors which should be reported individually by this program for each line for each transmission test. Errors which are not reported individually are reported in summary error reports.
3. ROM VERSION PRINTOUT ON THE FIRST PASS - This question asks whether the program should print out the versions of the on board 8051 processor ROMs during the first pass of the program.

PROGRAM DOCUMENT

2.7 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 HAVE A MULTIPLEXED DEVICE SUCH AS A MASS STORAGE CONTROLLER WITH SEVERAL DRIVES OR A COMMUNICATION DEVICE WITH SEVERAL LINES, THIS BECOMES TEDIOUS SINCE MOST OF THE ANSWERS ARE REPETITIOUS.

TO ILLUSTRATE A MORE EFFICIENT METHOD, SUPPOSE YOU ARE TESTING A FICTIONAL DEVICE, THE XY11. SUPPOSE THIS DEVICE CONSISTS OF A CONTROL MODULE WITH EIGHT UNITS (SUB-DEVICES) ATTACHED TO IT. THESE UNITS ARE DESCRIBED BY THE OCTAL NUMBERS 0 THROUGH 7. THERE IS ONE HARDWARE PARAMETER THAT CAN VARY AMONG UNITS CALLED THE Q-FACTOR. THIS Q-FACTOR MAY BE 0 OR 1. BELOW IS A SIMPLE WAY TO BUILD A TABLE FOR ONE XY11 WITH EIGHT UNITS.

```
# UNITS (0) ? 8<CR>
```

```
UNIT 1
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 0<CR>
Q-FACTOR (0) 0 ? 1<CR>
```

```
UNIT 2
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 1<CR>
Q-FACTOR (0) 1 ? 0<CR>
```

```
UNIT 3
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 2<CR>
Q-FACTOR (0) 0 ? <CR>
```

```
UNIT 4
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 3<CR>
Q-FACTOR (0) 0 ? <CR>
```

```
UNIT 5
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 4<CR>
Q-FACTOR (0) 0 ? <CR>
```

```
UNIT 6
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 5<CR>
Q-FACTOR (0) 0 ? <CR>
```

```
UNIT 7
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 6<CR>
Q-FACTOR (0) 0 ? 1<CR>
```

```
UNIT 8
CSR ADDRESS (0) 160000<CR>
SUB DEVICE # (0) ? 7<CR>
```

PROGRAM DOCUMENT

Q-FACTOR (0) 1 ? <CR>

NOTICE THAT THE DEFAULT VALUE FOR THE Q-FACTOR CHANGES WHEN A NON-DEFAULT RESPONSE IS GIVEN. BE CAREFUL WHEN SPECIFYING MULTIPLE UNITS!

AS YOU CAN SEE FROM THE ABOVE EXAMPLE, THE HARDWARE PARAMETERS DO NOT VARY SIGNIFICANTLY FROM UNIT TO UNIT. THE PROCEDURE SHOWN IS NOT VERY EFFICIENT.

THE RUNTIME SERVICES CAN TAKE MULTIPLE UNIT SPECIFICATIONS HOWEVER. LET'S BUILD THE SAME TABLE USING THE MULTIPLE SPECIFICATION FEATURE.

UNITS (0) ? 8<CR>

UNIT 1
 CSR ADDRESS (0) ? 160000<CR>
 SUB-DEVICE # (0) ? 0,1<CR>
 Q-FACTOR (0) 0 ? 1,0<CR>

UNIT 3
 CSR ADDRESS (0) ? 160000<CR>
 SUB-DEVICE # (0) ? 2-5<CR>
 Q-FACTOR (0) 0 ? 0<CR>

UNIT 7
 CSR ADDRESS (0) ? 160000<CR>
 SUB-DEVICE # (0) ? 6,7<CR>
 Q-FACTOR (0) 0 ? 1<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 THE FIRST PASS, TWO ENTRIES ARE BUILT SINCE TWO SUB-DEVICES AND Q-FACTORS WERE SPECIFIED. THE SERVICES ASSUME THAT THE CSR ADDRESS IS 160000 FOR BOTH SINCE IT WAS SPECIFIED ONLY ONCE. IN THE SECOND PASS, FOUR ENTRIES WERE BUILT. THIS IS BECAUSE FOUR SUB-DEVICES WERE SPECIFIED. THE "-" CONSTRUCT TELLS THE RUNTIME SERVICES TO INCREMENT THE DATA FROM THE FIRST NUMBER TO THE SECOND. IN THIS CASE, SUB-DEVICES 2, 3, 4 AND 5 WERE SPECIFIED. (IF THE SUB-DEVICE WERE SPECIFIED BY ADDRESSES, THE INCREMENT WOULD BE BY 2 SINCE ADDRESSES MUST BE ON AN EVEN BOUNDARY.) THE CSR ADDRESSES AND Q-FACTORS FOR THE FOUR ENTRIES ARE ASSUMED TO BE 160000 AND 0 RESPECTIVELY SINCE THEY WERE ONLY SPECIFIED ONCE. THE LAST TWO UNITS ARE SPECIFIED IN THE THIRD PASS.

THE WHOLE PROCESS COULD HAVE BEEN ACCOMPLISHED IN ONE PASS AS SHOWN BELOW.

UNITS (0) ? 8<CR>

UNIT 1
 CSR ADDRESS (0) ? 160000<CR>
 SUB DEVICE # (0) ? 0-7<CR>
 Q-FACTOR (0) 0 ? 0,1,0,....,1,1<CR>

AS YOU CAN SEE FROM THIS EXAMPLE, NULL REPLIES (COMMAS ENCLOSING
A NULL FIELD) TELL THE RUNTIME SERVICES TO REPEAT THE LAST REPLY.

PROGRAM DOCUMENT

2.8 QUICK START-UP PROCEDURE (XXDP+)

TO START-UP THIS PROGRAM:

1. BOOT XXDP+
2. GIVE THE DATE AND ANSWER THE LSI AND 50HZ (IF THERE IS A CLOCK AND THE QUESTION IS ASKED) QUESTIONS
3. TYPE "R NAME", WHERE NAME IS THE NAME OF THE BIN OR BIC FILE FOR THIS PROGRAM
4. TYPE "START"
5. ANSWER THE "CHANGE HW" QUESTION WITH "Y"
6. ANSWER ALL THE HARDWARE QUESTIONS
7. ANSWER THE "CHANGE SW" QUESTION WITH "N"

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING ONLY THE DEFAULTS FOR FLAGS AND SOFTWARE PARAMETERS. FOR DEFAULT INFORMATION SEE THE SECTIONS WITHIN THIS DOCUMENT ON FLAGS, AND HARDWARE QUESTIONS.

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 (SEE THE FLAGS SECTION OF THIS DOCUMENT).

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

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 (SEE THE FLAGS SECTION OF THIS DOCUMENT). THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL MESSAGE.

EXTENDED ERROR MESSAGES CONTAIN SUPPLEMENTARY ERROR INFORMATION SUCH AS REGISTER CONTENTS OR GOOD/BAD DATA. THESE ARE ALWAYS PRINTED UNLESS THE "IER", "IBR" OR "IXR" FLAGS ARE SET (SEE THE

PROGRAM DOCUMENT

FLAGS SECTION OF THIS DOCUMENT).
THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL ERROR
MESSAGE AND ANY ASSOCIATED BASIC ERROR MESSAGES.

3.2 ERROR MESSAGES

This program is intended to provide a go/no-go indication of the functionality of DHV11-M boards. To execute the program in this mode the operator can run with the inhibit basic error reporting switch. In this mode the program prints error messages which contain the error message header described above, plus the name of the failing test. For a list of the test names in this program see the test summaries section of this document. An example of such an error message is the following:

```
CVDHA DVC FTL ERR 01603 ON UNIT 02 TST 015 SUB 000 PC: 015244  
DEVICE REGISTER WORD READ/WRITE TEST
```

This error indicates that a fatal error was encountered within the test which tests the read/write capability of the DHV11-M registers.

If the operator requires more extensive error reporting he can run with all error reporting enabled by not using the inhibit reporting switches. The above error message would then become the following:

```
CVDHA DVC FTL ERR 01603 ON UNIT 02 TST 015 SUB 000 PC: 015244  
DEVICE REGISTER WORD READ/WRITE TEST  
BAD BIT(S) IN DEVICE TBUFFAD1 REGISTER FOR LINE 7 (D).  
EXPECTED DATA: 000000 (0).  
ACTUAL DATA: 000023 (0).
```

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. FOR FUTURE INFORMATION SEE THE SWITCHES SECTION OF THIS DOCUMENT.

5.0 TEST SUMMARIES

The following tests are included within CVDHA:

1. Device register address test - Verifies that the UUT registers will respond with the proper Q-BUS handshaking when accessed. Verifies that the UUT is at the proper address.
2. MASTER.RESET (Selftest) test - Verifies that the MASTER.RESET bit clears within a specified time of it being set.
3. MASTER.RESET (Skip Selftest) test - Verifies that the MASTER.RESET bit clears within a short time after it is set if the skip selftest sequence is used.
4. RX.CHARACTER field test - Verifies that the data bits of the codes in the FIFO after a reset and skip selftest are consistent with the skip selftest codes.
5. Reception flag field test - Verifies that the 3 data status bits (OVERRUN, FRAMING, and PARITY error bits) are all set on all of the skip selftest codes in the FIFO after a reset and skip selftest sequence.
6. RX.DATA.AVAIL test - Verifies that the RX.DATA.AVAIL bit is set when the skip selftest codes are in the FIFO and that it clears after they are read.
7. RX.DATA.VALID test - Verifies that the RX.DATA.VALID bit is set for each valid skip selftest code in the FIFO and clear after all valid codes are read.
8. RX.LINE field test - Verifies that the RX.LINE fields are correct for the skip selftest codes.
9. BMP run test - This test runs the BMP and verifies that it does not fail within a specified period. This test should signal problems that the BMP codes could cause with later tests.
10. Skip selftest test - This test verifies that if the selftest is skipped the proper codes are placed in the FIFO and that no errors are encountered.

PROGRAM DOCUMENT

11. DIAGNOSTIC.FAIL (Skip selftest) test - This test verifies, by using the skip selftest sequence, that the DIAGNOSTIC.FAIL bit will go to both the active and inactive states.
12. Selftest run test - Verifies that no errors are found by the execution of the selftest.
13. Selftest fail test - Verifies that the selftest will report errors correctly when it is forced to fail.
14. ROM version printout test - If requested, reports the version numbers of the 8051 ROMs.
15. Word access read/write test - Verifies that the registers respond correctly to word read and write accesses.
16. Word access read/modify/write test - Verifies that the registers respond correctly to read/modify/write word accesses.
17. Byte access read/write test - Verifies that the registers respond correctly to byte read and write accesses.
18. Byte access read/modify/write test - Verifies that the registers respond correctly to read/modify/write byte accesses.
19. ID.BIT test - Verifies that the ID.BIT reads as a zero.
20. No TX.DATA.VALID/No TX.ACTION test - Verifies that if a data word is written without the TX.DATA.VALID bit set, no TX.ACTION is generated. This test does not require that characters are TXed.
21. TX.DATA.VALID / TX.ACTION test - Verifies that if a data word is written with the TX.DATA.VALID bit set, it generates a corresponding TX.ACTION. This test does not require that characters are TXed.
22. TX.ENABLE inactive test - Verifies that if the TX.ENABLE bit is clear no transmission occurs.
23. TX.ENABLE active test - Verifies that TX occurs if the TX.ENABLE is set.
24. Interrupts test - Verifies that the TX and RX interrupts are functioning correctly.
25. BR level test - Verifies that the UUT generates TX and RX interrupts at the correct BR level.
26. DIAG field (BMP) test - Verifies that a request for BMP code reporting is answered by the UUT within the specified time.

PROGRAM DOCUMENT

27. Report BMP codes test - This pseudo test reports the first 32 BMP codes which were discovered in the FIFO during the execution of the other tests. This avoids the interruption of other tests by these codes, if they are not critical to the tests being performed.

6.0 EXAMPLE ERROR FREE PASS

The following is an example of an error free pass dialogue:

.R CVDHACO
CVDHACO.BIC

DRS
CVDHA-C-0
DHV11-M FUNC TST PART1
UNIT IS DHV11-M
RESTART ADDR: 147670
DR>STA

CHANGE HW (L) ? Y

UNITS (D) ? 2

UNIT 0
CSR ADDRESS: (0) 160460 ? +Z

UNIT 1
CSR ADDRESS: (0) 160460 ? 160040
INTERRUPT VECTOR ADDRESS: (0) 300 ? 320
ACTIVE LINE BIT MAP: (0) 377 ? <CR>
INTERRUPT BR LEVEL: (0) 4? <CR>

CHANGE SW (L) ? Y

REPORT UNIT NUMBER AS EACH UNIT IS TESTED: (L) Y ? <CR>
NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE: (D) 0 ? 4
ROM VERSION PRINTOUT ON THE FIRST PASS: (L) Y ?

TESTING UNIT : 0(D)

ROM VERSION NUMBERS: PROC_1 = 2(D) PROC_2 = 2(D)

TESTING UNIT : 1(D)

ROM VERSION NUMBERS: PROC_1 = 2(D) PROC_2 = 2(D)

CVDHA EOP 1
0 CUMULATIVE ERRORS

TESTING UNIT : 0(D)

+C
DR> EXIT

PROGRAM DOCUMENT

```

1073      E
1074      .LIST SEQ,LOC,BIN,MEB
1075      .NLIST CND
1076      ;*****
1077      ;
1078      ;           VDHA.PHD
1079      ;
1080      ;*****
1081
1082
1083
1084      .SBTTL  Program Header
1085
1086
1087      .MCALL  SVC
1088 000000      SVC           ; INITIALIZE SUPERVISOR MACROS
1089
1090      ;*****
1091      ;           IF STRUCTURED MACROS ARE TO BE USED, ADD ".MCALL STRUCT" AND "STRUCT"
1092      ;           TO INITIALIZE THE STRUCTURED MACROS.
1093
1094      000001      SVCINS= 1      ; LIST INSTRUCTIONS, SHIFTED RIGHT
1095      000001      SVCTST= 1     ; LIST TEST TAGS, SHIFTED RIGHT
1096      000001      SVCSUB= 1    ; LIST SUBTEST TAGS, SHIFTED RIGHT
1097      000001      SVCGBL= 1    ; LIST GLOBAL TAGS, SHIFTED RIGHT
1098      000001      SVCTAG= 1    ; LIST OTHER TAGS, SHIFTED RIGHT
1099
1100      ;           CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
1101      ;           TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
1102      ;           SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
1103      ;           CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.
1104      ;*****
1105
1106 000000      .ENABL  ABS
1107      ;.ENABL  AMA
1108      =           2000
1109
1110 002000      BGNMOD
1111
1112      ;**
1113      ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
1114      ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
1115      ;--
1116
1117 002000      POINTER BGNRPT,BGNSW,BGNSFT,BGNDU,ERRTBL
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136 002000      HEADER  CVDHA,C,0,16,0,PRI07
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200

```

```

L$NAME::
        .ASCII /C/
        .ASCII /V/
        .ASCII /D/
        .ASCII /H/
        .ASCII /A/
        .BYTE  0

```

Program Header

002006	000		
002007	000		
002010			
002010	103	L\$REV::	.BYTE 0
002011			.BYTE 0
002011	060	L\$DEPO::	.ASCII /C/
002012			.ASCII /O/
002012	000000	L\$UNIT::	.WORD 0
002014			
002014	000016	L\$TIML::	.WORD 16
002016			
002016	036354	L\$HPCP::	.WORD L\$HARD
002020			
002020	036546	L\$SPCP::	.WORD L\$SOFT
002022			
002022	002214	L\$HPTP::	.WORD L\$HW
002024			
002024	002226	L\$SPTP::	.WORD L\$SW
002026			
002026	037064	L\$LADP::	.WORD L\$LAST
002030			
002030	000000	L\$STA::	.WORD 0
002032			
002032	000000	L\$CO::	.WORD 0
002034			
002034	000000	L\$DTYP::	.WORD 0
002036			
002036	000000	L\$APT::	.WORD 0
002040			
002040	002124	L\$DTP::	.WORD L\$DISPATCH
002042			
002042	000340	L\$PRIO::	.WORD PRI07
002044			
002044	000000	L\$ENVI::	.WORD 0
002046			
002046	000000	L\$EXP1::	.WORD 0
002050			
002050	004	L\$MREV::	.WORD 0
002051	000		.BYTE C\$REVISION
002052			.BYTE C\$EDIT
002052	000000	L\$EF::	.WORD 0
002054	000000		.WORD 0
002056			
002056	000000	L\$SPC::	.WORD 0
002060			
002060	004036	L\$DEVP::	.WORD L\$DVTYP
002062			
002062	024234	L\$REPP::	.WORD L\$RPT
002064			
002064	000000	L\$EXP4::	.WORD 0
002066			
002066	000000	L\$EXP5::	.WORD 0
002070			
002070	000000	L\$AUT::	.WORD 0
002072			
002072	025156	L\$DUT::	.WORD 0
002074			.WORD L\$DU
		L\$LUN::	

Program Header

002074 000000
 002076 004046
 002100 104035
 002102 003766
 002104 024250
 002106 025120
 002110 025116
 002112 024242
 002114 000000
 002116 000000
 002120 000000

1137

L\$DESP:: .WORD 0
 L\$LOAD:: .WORD L\$DESC
 L\$ETP:: EMT E\$LOAD
 L\$ICP:: .WORD L\$ERRTBL
 L\$CCP:: .WORD L\$INIT
 L\$ACP:: .WORD L\$CLEAN
 L\$PRT:: .WORD L\$AUTO
 L\$TEST:: .WORD L\$PROT
 L\$DLY:: .WORD 0
 L\$HIME:: .WORD 0

DISPATCH TABLE

1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156 002122
 002122 000033
 002124
 002124 025274
 002126 025564
 002130 026000
 002132 026230
 002134 026412
 002136 026570
 002140 026772
 002142 027164
 002144 027356
 002146 027530
 002150 027730
 002152 030142
 002154 030410
 002156 030752
 002160 031336
 002162 031546
 002164 031762
 002166 032226
 002170 032476
 002172 032604
 002174 033014
 002176 033250
 002200 033550
 002202 034106
 002204 035132
 002206 036014
 002210 036272
 1157

.SBTTL DISPATCH TABLE

 ; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
 ; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.

DISPATCH 27

.WORD 27
 L\$DISPATCH::
 .WORD T1
 .WORD T2
 .WORD T3
 .WORD T4
 .WORD T5
 .WORD T6
 .WORD T7
 .WORD T8
 .WORD T9
 .WORD T10
 .WORD T11
 .WORD T12
 .WORD T13
 .WORD T14
 .WORD T15
 .WORD T16
 .WORD T17
 .WORD T18
 .WORD T19
 .WORD T20
 .WORD T21
 .WORD T22
 .WORD T23
 .WORD T24
 .WORD T25
 .WORD T26
 .WORD T27

DISPATCH TABLE

```

1165
1166
1167
1168
1169
1170
*****
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183 002212      BGNHW  DFPTBL
      002212      000004
      002214
      002214
1184
1185 002214      160460      .WORD  160460 ;Default CSR Address
1186 002216      000300      .WORD  300    ;Default Vector Address
1187 002220      177777      .WORD  177777 ;Default Active lines bit map
1188 002222      004        .BYTE  4     ;Default BR Level
1189
1190
1191 002224      ENDPHW
      002224
                                     L$HW: .WORD  L10000-L$HW/2
                                     DFPTBL:
                                     L10000:

```

DEFAULT HARDWARE P-TABLE

```

1193
1194 ;*****
1195 ;
1196 ;           VDHA.SWT
1197 ;
1198 ;*****
1199
1200
1201
1202 .SBTTL SOFTWARE P-TABLE
1203
1204 ;**
1205 ; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
1206 ; PROGRAM AS OPERATIONAL PARAMETERS. THESE PARAMETERS ARE
1207 ; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
1208 ; AT RUN TIME.
1209 ; -
1210
1211 002224          BGNSW  SFPTBL
      002224 000002
      002226
      002226
                                L$SW:: .WORD L10001-L$SW/2
                                SFPTBL::
1212
1213 002226 000021
1214 002230 000000
      OPTION:: .WORD 21 ;bit map of program control flags
      NDERPT:: .WORD 0  ;Default number of individual data errors to rpt.
1215
1216 002232          ENDSW
      002232
                                L10001:

```

SOFTWARE P-TABLE

1218
1219

1220
1221
1222
1223
1224
1225

;
;
; VDHA.EQU
;

1226
1227
1237
1238
1239
1240

.SBTTL GLOBAL EQUATES SECTION

1241
1242
1243
1244

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

1245 000010
1246 000377
1247

NUMLNS==10 ;NUMBER OF LINES ON DHV11 IS 8.
MAPLNS==377 ;BIT MAP OF LINES ON DHV11.

1248
1249 000000
1250 000002
1251 000002
1252 000004
1253 000006
1254 000010
1255 000012
1256 000014
1257 000016
1258

***** DEVICE REGISTER OFFSETS FROM THE CSR'S ADDRESS *****
CSRO==0 ;CSR REGISTER OFFSET FROM THE CSR ADDRESS
RBUFO==2 ;RECEIVE REGISTER OFFSET FROM THE CSR ADDRESS
TXCHRO==2 ;TRANSMIT REGISTER OFFSET FROM THE CSR ADDRESS
LPRO==4 ;LINE PARAMETER REGISTER OFFSET FROM THE CSR ADDRESS
STATO==6 ;STATUS REGISTER OFFSET FROM THE CSR ADDRESS
LNCTRO==10 ;LINE CONTROL REGISTER OFFSET FROM THE CSR ADDRESS
TXAD10==12 ;TRANSMIT ADDRESS 1 REGISTER OFFSET FROM THE CSR ADDRESS
TXAD20==14 ;TRANSMIT ADDRESS 2 REGISTER OFFSET FROM THE CSR ADDRESS
TXBFCO==16 ;TRANSMIT COUNT REGISTER OFFSET FROM THE CSR ADDRESS

1259
1260 000020
1261 000030
1262 000100
1263

***** EQUATES USED WITH RESPECT TO THE RX BUFFER *****
RXBETX==16. ;LEVEL OF RX BUFFER AT WHICH TO RE-ENABLE TRANSMISSION.
RXBDTX==24. ;LEVEL OF RX BUFFER AT WHICH TO DISABLE TRANSMISSION.
RXBFUL==64. ;TOTAL CHARACTER CAPACITY OF THE RX BUFFER.

1264
1279 002232

EQUALS

;
; BIT DIFINITIONS

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002

BIT15== 100000
BIT14== 40000
BIT13== 20000
BIT12== 10000
BIT11== 4000
BIT10== 2000
BIT09== 1000
BIT08== 400
BIT07== 200
BIT06== 100
BIT05== 40
BIT04== 20
BIT03== 10
BIT02== 4
BIT01== 2

GLOBAL EQUATES SECTION

```

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
;
;
; BIT POSITION IN SECOND STATUS WORD
000040          EF.START== 32. ; (100000) START COMMAND WAS ISSUED
000037          EF.RESTART== 31. ; (040000) RESTART COMMAND WAS ISSUED
000036          EF.CONTINUE== 30. ; (020000) CONTINUE COMMAND WAS ISSUED
000035          EF.NEW== 29. ; (010000) A NEW PASS HAS BEEN STARTED
000034          EF.PWR== 28. ; (004000) A POWER-FAIL/POWER UP OCCURRED
;
; PRIORITY LEVEL DEFINITIONS
;
000340          PRI07== 340
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

```


GLOBAL EQUATES SECTION

```

1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302 002232 000300
1303 002234 000304
1304 002236 000377
1305 002240 000000
1306 002242 004
1307
1308
1309
1310
1311
1312 002244
1313 002244 160020
1314 002246 160022
1315 002250 160024
1316 002252 160026
1317 002254 160030
1318 002256 160032
1319 002260 160034
1320 002262 160036
1321
1322
1323
1324
1325
1326 002264 137660
1327 002266 177777
1328 002270 000007
1329 002272 177777
1330 002274 166051
1331 002276 000000
1332 002300 077700
1333 002302 000000
1334
1335
1336
1337
1338 002304 006004

```

```

;*****
;
;          VDHA.GDT
;*****
;*****
;
;SBTTL GLOBAL DATA SECTION
;
;+
; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
; IN MORE THAN ONE TEST.
;--
;*****
;
;          Unit Variable Area
;*****
;
;          RXVECA:: .WORD 300      ;RX VECTOR ADDRESS.
;          TXVECA:: .WORD 304      ;TX VECTOR ADDRESS.
;          ACTLNS:: .WORD 377      ;ACTIVE LINE BIT MAP.
;          UNITN::  .WORD 0        ;UNIT NUMBER.
;          BRLEVL:: .BYTE 4        ;INTERRUPT BUS REQUEST LEVEL
;                   .EVEN
;*****
;
;          Device Register Address Table
;*****
;
;          DRADRT::
;          CSRA::  .WORD 160020    ;DHV11-M CSR ADDRESS
TXCHA:: RBUFA:: .WORD 160022    ;DHV11-M RECEIVE/TRANSMIT BUFFER ADDRESS
;          LPRA::  .WORD 160024    ;DHV11-M LINE PARAMETER REGISTER ADDRESS
;          STATA:: .WORD 160026    ;DHV11-M STATUS REGISTER ADDRESS
;          LNCTRA::.WORD 160030    ;DHV11-M LINE CONTROL REGISTER ADDRESS
;          TXAD1A::.WORD 160032    ;DHV11-M TRANSMIT BUFFER 1 REGISTER ADDRESS
;          TXAD2A::.WORD 160034    ;DHV11-M TRANSMIT BUFFER 2 REGISTER ADDRESS
;          TXBFCA::.WORD 160036    ;DHV11-M TRANSMIT BUFFER COUNT REGISTER ADDRESS
;*****
;
;          Bit mask table of un-used DHV device register bits.
;*****
;
;          UNBTTB::.WORD 137660    ;UNUSED BIT MASK FOR THE CSR
;                   .WORD 177777    ;UNUSED BIT MASK FOR THE RBUF/TX REG
;                   .WORD 7         ;UNUSED BIT MASK FOR THE LPR
;                   .WORD 177777    ;UNUSED BIT MASK FOR THE STAT
;                   .WORD 166051    ;UNUSED BIT MASK FOR THE LNCTRL
;                   .WORD 0         ;UNUSED BIT MASK FOR THE TBUFFAD1
;                   .WORD 77700    ;UNUSED BIT MASK FOR THE TBUFFAD2
;                   .WORD 0         ;UNUSED BIT MASK FOR THE TBUFFCT
;*****
;
;          Register Message Address Table
;*****
;
;          RMATBB:: .WORD DROOMG    ;ADDRESS OF "CSR" MESSAGE.

```

GLOBAL DATA SECTION

1339 002306 006010
 1340 002310 006015
 1341 002312 006021
 1342 002314 006026
 1343 002316 006035
 1344 002320 006046
 1345 002322 006057
 1346
 1347
 1348
 1349
 1350 002324 000000
 1351 002326 000001
 1352 002330 000000
 1353 002332 000000
 1354 002334 000000
 1355 002336 000000
 1356 002340 000000
 1357 002342 000000
 1358 002344 000000
 1359 002346 000000
 1360 002350 000000
 1361 002352 000000
 1362
 1363
 1364
 1365
 1366 002354 177546
 1367 002356 000300
 1368 002360 000100
 1369 002362 000074
 1370 002364 000000
 1371 002366 000000
 1372 002370 000170
 1373 002372 000170
 1374 002374 000021
 1375 002376 000062
 1376
 1377
 1378
 1379
 1380 002400 177572
 1381 002402 000000
 1382 002404 000000
 1383 002406 172340
 1384
 1385
 1386
 1387
 1388 002410 000001
 1389 002412 000002
 1390 002414 000004
 1391 002416 000010
 1392 002420 000020
 1393 002422 000040
 1394 002424 000100
 1395 002426 000200

.WORD DR02MG ;ADDRESS OF "RBUF" MESSAGE.
 .WORD DR04MG ;ADDRESS OF "LPR" MESSAGE.
 .WORD DR06MG ;ADDRESS OF "STAT" MESSAGE.
 .WORD DR10MG ;ADDRESS OF "LNCTRL" MESSAGE.
 .WORD DR12MG ;ADDRESS OF "TBUFAD1" MESSAGE.
 .WORD DR14MG ;ADDRESS OF "TBUFAD2" MESSAGE.
 .WORD DR16MG ;ADDRESS OF "TBUFCT" MESSAGE.

 ; Assorted global variables:

BUFPTR:: .WORD 0 ;STORAGE FOR RECEIVE CHARACTER BUFFER POINTER.
 TSTNUM:: .WORD 1 ;STORAGE FOR THE TEST NUMBER.
 IESTAT:: .WORD 0 ;STORAGE FOR THE INTERRUPT ENABLE BIT STATES.
 PASCNT:: .WORD 0 ;STO'G FOR PASS COUNT USED IN ROM VERSION# TST.
 RXINTC:: .WORD 0 ;STORAGE FOR RECEIVER INTERRUPT FLAGS.
 RXINTF:: .WORD 0 ;STORAGE FOR RECEIVER INTERRUPT FLAGS.
 TXINTC:: .WORD 0 ;STORAGE FOR TRANSMIT INTERRUPT COUNT.
 TXINTF:: .WORD 0 ;STORAGE FOR TRANSMIT INTERRUPT FLAGS.
 TP4VEC:: .WORD 0 ;STORAGE FOR THE NORMAL 004 TRAP VECTOR.
 TP4FLG:: .WORD 0 ;FLAGS SET WHEN AN EXPECTED 004 TRAP OCCURS.
 WORD1:: .WORD 0 ;LOCATION FOR PASSING INDIRECT PARAMETERS.
 CTRLCF:: .WORD 0 ;STORAGE FOR THE CONTROL-C FLAG.

 ; Line Time Clock variables and storage.

CLKCSR:: .WORD 177546 ;CSR ADDRESS OF THE LTC.
 CLKBRL:: .WORD PRI06 ;INTERRUPT PRIORITY LEVEL OF THE LTC.
 CLKVEC:: .WORD 100 ;INTERRUPT VECTOR ADDRESS OF THE LTC.
 CLKHRZ:: .WORD 60. ;INTERRUPT FREQUENCY OF THE LTC.
 TIMER1:: .WORD 0 ;HARDWARE CLOCK COUNTER #1.
 TIMER2:: .WORD 0 ;HARDWARE CLOCK COUNTER #2.
 TIMER3:: .WORD 120. ;HARDWARE BREAK COUNTER LOCATION.
 BCOUNT:: .WORD 120. ;BREAK COUNT VALUE IN CLOCK TICKS.
 MSTICK:: .WORD 17. ;NUMBER OF MILLI-SECONDS PER LTC TICK.
 MSLCNT:: .WORD 62 ;LOOP COUNT (USED BY MSLOOP) TO DELAY 1 MS.

 ; Memmory Management Variables and Flags.

MMSR0:: .WORD 177572 ;ADDRESS OF MEM MGT STATUS REGISTER #0.
 MMPRES:: .WORD 0 ;MEM MGT PRESENT FLAG (0 IF MM NOT PRESENT).
 MMENAB:: .WORD 0 ;MEM MGT ENABLED FLAG (0 IF MM NOT ENABLED).
 PAROA:: .WORD 172340 ;ADDRESS OF MEM MGT PAR #0.

 ; Table of words with corresponding bit set for generation of bit maps.

BITTBL:: .WORD 1 ;BIT 0 SET.
 .WORD 2 ;BIT 1 SET.
 .WORD 4 ;BIT 2 SET.
 .WORD 10 ;BIT 3 SET.
 .WORD 20 ;BIT 4 SET.
 .WORD 40 ;BIT 5 SET.
 .WORD 100 ;BIT 6 SET.
 .WORD 200 ;BIT 7 SET.

GLOBAL DATA SECTION

```

1396 002430 000400          .WORD 400          ;BIT 8 SET.
1397 002432 001000          .WORD 1000         ;BIT 9 SET.
1398 002434 002000          .WORD 2000         ;BIT 10 SET.
1399 002436 004000          .WORD 4000         ;BIT 11 SET.
1400 002440 010000          .WORD 10000        ;BIT 12 SET.
1401 002442 020000          .WORD 20000        ;BIT 13 SET.
1402 002444 040000          .WORD 40000        ;BIT 14 SET.
1403 002446 100000          .WORD 100000       ;BIT 15 SET.
1404
1405
1406 ;*****
;*   GPR Save Area Zero.
1407 ;*****
1408 002450          GPRSOB::          ;BASE OF GPR SAVE AREA NUMBER ZERO.
1409 002450 000000          .WORD 0            ;WORD 1, STORAGE FOR R1.
1410 002452 000000          .WORD 0            ;WORD 2, STORAGE FOR R2.
1411 002454 000000          .WORD 0            ;WORD 3, STORAGE FOR R3.
1412 002456 000000          .WORD 0            ;WORD 4, STORAGE FOR R4.
1413 002460 000000          .WORD 0            ;WORD 5, STORAGE FOR R5.
1414
1415 ;*****
1416 ;*   Transmission and Reception Variables, Pointers, and Flags.
1417 ;*****
1418 002462 000000          ERSMRF:: .WORD 0      ;ERROR SUMMARY REPORT FLAGS.
1419 002464          ERCNTB:: .BLKW 16.    ;TABLE OF ERROR COUNTERS.
1420
1421 ;*****
1422 ;   Storage area for the BMP code queue.
1423 ;*****
1424 002524 000000          BMPCQP:: .WORD 0        ;POINTER USED TO ACCESS THE NEXT CELL IN QUE.
1425 002526          BMPCQB:: .BLKW 64.    ;STORAGE FOR 32 CELLS, TEST# PLUS BMP CODE.
1426 002726          BMPCQE::          ;LAST ADDRESS PLUS 2 OF THE BMP CODE QUEUE.
1427 ;*****
1428 ;   General table and buffer area--513 words.
1429 ;*****
1430 002726          BUFBAS::          ;BASE OF MEMORY BUFFER.
1431 002726          ERLTBL:: .BLKW 128.   ;FIRST HALF OF GENERAL TABLE OR BUFFER.
1432 003326          BUFMID:: .BLKW 64.    ;SECOND HALF OF GENERAL TABLE OR BUFFER.
1433 003526          BUF3QT:: .BLKW 64.    ;LAST QUARTER OF THE BUFFER AREA.
1434 003726          BUFEND::          ;END OF GENERAL PURPOSE MEMORY BUFFER.
1435 003726          ENDETBL:: .BLKW 16.    ;BUFFER OVERFLOW SPACE.
1436
1449 003766          ERRTBL
003766
003766 000000          ERRRTBL::          L$ERRTBL::
003770 000000          ERRRTYP:: .WORD 0
003772 000000          ERRNBR:: .WORD 0
003774 000000          ERRMSG:: .WORD 0
1450          ERRBLK:: .WORD 0
1451          .EVEN

```

GPR HANDLING ROUTINES FOR SUBROUTINE CALLS.

```

1453 .SBTTL GPR HANDLING ROUTINES FOR SUBROUTINE CALLS.
1454 ;*****
1455 ;*
1456 ;*   There are 4 routines and macro definitions used for the handling of
1457 ;*   GPR values during subroutine calls within this program. The four
1458 ;*   routines/macro calls have the following names:
1459 ;*
1460 ;*   SAVE - Macro definition used at the beginning of a subroutine to
1461 ;*         save the GPR contents for later restoration.
1462 ;*   PASS - Macro definition used at the end of a subroutine to restore
1463 ;*         the previously saved GPR contents and to leave the contents
1464 ;*         of the specified GPR(s) intact (NOT restored).
1465 ;*   PREG05 - Subroutine which is called from the SAVE and PASS macro
1466 ;*           expansions which actually performs the actions on the GPRs.
1467 ;*
1468 ;*   During a subroutine which uses these GPR save routines the values
1469 ;*   of the GPRs are stored on the stack in the following stack frame:
1470 ;*
1471 ;*       SP      -> RET PC INTO PREG05 ROUTINE.
1472 ;*       SP+2    -> GPR R0 CONTENTS.
1473 ;*       SP+4    -> GPR R1 CONTENTS.
1474 ;*       SP+6    -> GPR R2 CONTENTS.
1475 ;*       SP+8    -> GPR R3 CONTENTS.
1476 ;*       SP+10   -> GPR R4 CONTENTS.
1477 ;*       SP+12   -> GPR R5 CONTENTS
1478 ;*       SP+14   -> RET PC INTO CALLER OF SUB'TNE WHICH CALLED PREG05.
1479 ;*
1480 ;*   Each level of sub'tne calling uses 8 words of stack overhead.
1481 ;*   The SAVE and PASS macros can also be used in "straight line code"
1482 ;*   to save and restore the GPR values. In any case, after the
1483 ;*   issuing of a PASS call the GPRs will be restored to the values
1484 ;*   they had prior to the last SAVE call (except for the excepted,
1485 ;*   or passed intact, GPRs specified as parameters to the PASS call)
1486 ;*   and the SP will also be restored to its condition before the last
1487 ;*   SAVE call. The programmer must be sure that the SP has the same
1488 ;*   value when the PASS macro is called as it had immediately after
1489 ;*   the SAVE macro was called.
;*****

```

GPR FRAME ACCESS EQUATES

.SBTTL GPR FRAME ACCESS EQUATES

```
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505
```

000036
000016
000014
000012
000010
000006
000004
000002

.SBTTL GPR FRAME ACCESS EQUATES
;+++
;Equates that allow access to the stack frame. These are the
;offsets into the stack for registers saved during the PREG05
;routine.
;---
LPCSLT== 36 ;Offset for last return PC.
PCSLOT== 16 ;Offset for return PC.
R5SLOT== 14 ;Offset for R5.
R4SLOT== 12 ;Offset for R4.
R3SLOT== 10 ;Offset for R3.
R2SLOT== 6 ;Offset for R2.
R1SLOT== 4 ;Offset for R1.
ROSLOT== 2 ;Offset for R0.

GLOBAL MACRO DEFINITION

- SAVE -

1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530

```

.SBTTL GLOBAL MACRO DEFINITION - SAVE -
;*****
;* This macro is used at the beginning of a subroutine to save the
;* contents of the GPRs R0 thru R5.
;*
;* INPUTS: SP - Unchanged since subroutine was entered
;* R5SLOT - Offset to stack slot for R5 (Equated to 14 Octal)
;*
;* OUTPUTS: GPR save area on the stack is loaded with the contents of GPRs
;* TOP OF STACK - Loaded with the return address into PREG05
;*
;* CALLING SEQUENCE: SAVE
;*
;* COMMENTS: No arguments are allowed.
;* The PASS macro should be called to restore the GPR values.
;*
;* SUBORDINATE ROUTINES CALLED: PREG05.
;*****
.MACRO SAVE
.LIST
JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
.NLIST
.ENDM SAVE

```

GLOBAL MACRO DEFINITION

- PASS -

1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579

```

.SBTTL GLOBAL MACRO DEFINITION          - PASS -
;*****
;* This macro is used in conjunction with the SAVE macro. It is
;* called at end of a subroutine to pass parameters in GPRs back to the
;* calling routine by altering the GPR save area on the stack and then
;* returning to PREG05 to restore the GPRs to their saved values.
;*
;* INPUTS:      Only allowed ARGUMENTS are "R0" thru "R5".
;*              ROSLOT thru R5SLOT must be equated to their respective GPR save
;*              slot offsets before calling this macro.
;*
;* OUTPUTS:     The GPR values are put in their respective slots on the stack.
;*
;* CALLING SEQUENCE:  PASS  R0,R1,...
;*
;* COMMENTS:    Any combination of GPR arguments may be listed in any order.
;*              For example, the following are legal:
;*              PASS  R1
;*              PASS  R4,R0,R2
;*              The GPRs listed as arguments will be passed intact to the
;*              calling routine, all other GPRs will be restored.
;*              The SP must be at its original value when PASS is called.
;*
;*              The macro call
;*              PASS  R0,R3
;*              expands into the following assembly code:
;*              MOV   R0,ROSLOT(SP)      ;PUT R0 IN STACK SLOT.
;*              MOV   R3,R3SLOT(SP)     ;PUT R3 IN STACK SLOT.
;*              JSR   PC,@(SP)+         ;RETURN TO PREG05 SUBRT.
;*              In this example GPRs R1, R2, R4, and R5 will be restored to
;*              their values contained in the stack frame and R0 and R3
;*              will be left at their values prior to this PASS call.
;*
;* SUBORDINATE ROUTINES CALLED: (PREGRT - Label within PREG05, value on stack.)
;*****
; .MACRO PASS  A,B,C,D,E,F
; .IRP  X,<A,B,C,D,E,F>
; .IF   NB,X
; .LIST
;           MOV   X,X'SLOT(SP)          ;PUT X IN STACK SLOT.
; .NLIST
; .ENDC
; .ENDM
; .LIST
;           JSR   PC,@(SP)+             ;RETURN TO PREG05 SUBRT.
; .NLIST
; .ENDM PASS

```

GLOBAL SUBROUTINE

- PREG05 -

```

1581 .SBTTL GLOBAL SUBROUTINE - PREG05 -
1582 ;*****
1583 ;* Preserve Registers R0 through R5 for subroutine calls.
1584 ;*
1585 ;* INPUTS: The return address back into the calling routine must be in
1586 ;* GPR R5. (i.e.- Macros use "JSR R5,PREG05".)
1587 ;*
1588 ;* OUTPUTS: Registers R0 through R5 are saved on the stack.
1589 ;*
1590 ;*CALLING SEQUENCE: SAVE ;Macro expansion calls PREG05.
1591 ;* [Subroutine code]...
1592 ;* PASS ;Macro expansion recalls PREG05.
1593 ;*
1594 ;*COMMENTS: This routine is re-entrant.
1595 ;*
1596 ;* Parameters may be passed out of a subroutine by modifying the
1597 ;* register save area on the stack. Use the PASS GPRn macro
1598 ;* to return GPR values intact.
1599 ;* Use the RnSLOT offsets from the SP to pass other parameters.
1600 ;* [Example: MOV VALUE,R0SLOT(SP) ]
1601 ;* Make sure the SP is at its original value when you do this.
1602 ;*
1603 ;*SUBORDINATE ROUTINES CALLED: None.
1604 ;*****
1605
1606 003776 PREG05: ;R5 HAS BEEN LOADED ON THE STACK BY THE SUBROUTINE CALL
1607 003776 010446 MOV R4,-(SP) ;SAVE R4
1608 004000 010346 MOV R3,-(SP) ;SAVE R3
1609 004002 010246 MOV R2,-(SP) ;SAVE R2
1610 004004 010146 MOV R1,-(SP) ;SAVE R1
1611 004006 010046 MOV R0,-(SP) ;SAVE R0
1612 004010 010546 MOV R5,-(SP) ;PUSH RETURN PC ON TOP OF STACK
1613 004012 016605 000014 MOV R5SLOT(SP),R5 ;RESTORE R5 TO VALUE IT HAD BEFORE CALLS
1614
1615 004016 004736 JSR PC,@(SP)+ ;Call the subroutine at the return address
1616 ;from the PREG05 call, putting the present
1617 ;PC on the stack as a return address into
1618 ;this (PREG05) routine.
1619
1620 ;+++
1621 ;The following code is executed when the calling routine does a
1622 ;"return" [JSR PC,@(SP)+] using the PC deposited on the stack above.
1623 ;---
1624
1625 004020 012605 PREGRT: MOV (SP)+,R5 ;Put return PC in R5.
1626 004022 012600 MOV (SP)+,R0 ;Restore R0.
1627 004024 012601 MOV (SP)+,R1 ;Restore R1.
1628 004026 012602 MOV (SP)+,R2 ;Restore R2.
1629 004030 012603 MOV (SP)+,R3 ;Restore R3.
1630 004032 012604 MOV (SP)+,R4 ;Restore R4.
1631
1632 004034 000205 RTS R5 ;Return to the subroutine which called PREG05.
1633 ;restoring R5 in the process.

```


GLOBAL TEXT SECTION

```

1635 .SBTTL GLOBAL TEXT SECTION
1637 :*****
1638 :
1639 :           FVTSKL1.P11
1640 :
1641 :*****
1642 :
1643 :
1644 :
1645 :**
1646 : THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
1647 : MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
1648 : MORE THAN ONE TEST.
1649 :--
1650 :
1651 :
1652 : NAMES OF DEVICES SUPPORTED BY PROGRAM
1653 :
1654 :           DEVTYP <DHV11-M>
1655 :
1656 :           L$DVTYP::
1657 :                   .ASCIZ *DHV11-M*
1658 :
1659 :           .EVEN
1660 :
1661 :
1662 : TEST DESCRIPTION
1663 :
1664 :           DESCRIPT      <DHV11-M FUNC TST PART 1>
1665 :
1666 :           L$DESC::
1667 :                   .ASCIZ /DHV11 M FUNC TST PA
1668 :
1669 :           .EVEN
1670 :
1671 :
1672 :
1673 :

```

004036				
004036	104	110	126	
004041	061	061	055	
004044	115	000		
004046				
004046	104	110	126	
004051	061	061	055	
004054	115	040	106	
004057	125	116	103	
004062	040	124	123	
004065	124	040	120	
004070	101	122	124	
004073	040	061	000	

RT 1/

GLOBAL TEXT SECTION

1675
1676

:*****

1677
1678
1679
1680
1681

:
: VDHA.FMT
:
:*****

1682
1683
1684
1685
1686
1687
1698
1699

:
: FORMAT STATEMENTS USED IN PRINT CALLS
:

GLOBAL TEXT SECTION

1708
1709
1710
1711
1712
1713

```
*****
:
:          VDHA.MSG
:
*****
```

1714
1715

.NLIST BIN

.SBTTL GLOBAL MESSAGE AREA

```
1718 : ***** FORMAT STATEMENTS *****
1719 004076 MFUNIT:: .ASCIZ /%N%A TESTING UNIT :%D4%A(D)%N/
1720 004134 EF0503:: .ASCIZ /%T%N/
1721 004141 EF0505:: .ASCIZ /%A %D5%A ILLEGAL INTERRUPTS RECEIVED.%N/
1722 004214 EF1401:: .ASCIZ /%N%A ROM VERSION NUMBERS: PROC_1 = %D2%A(D) PROC_2 = %D2%A(D)%N/
1723 004316 EF1402:: .ASCIZ /%T%A ROM VERSION NUMBER %T%N/
1724 004353 EF1601:: .ASCIZ /%A %T%A ABORTED %N/
1725 004377 EF1602:: .ASCIZ /%A EXPECTED DATA: %06%A (0)%N/
1726 004441 EF1603:: .ASCIZ /%A ACTUAL DATA: %06%A (0)%N/
1727 004503 EF1604:: .ASCIZ /%A BAD BIT(S) IN DEVICE %T%A REGISTER FOR LINE %D2%A (D)%N/
1728 004600 EF3001:: .ASCIZ /%A EXPECTED OR CORRECT VALUE: %03%N/
1729 004647 EF3002:: .ASCIZ /%A ACTUAL OR MEASURED VALUE: %03%N/
1730 004716 EF9001:: .ASCIZ /%A UNEXPECTED %T%A FOUND IN RECEIVE CHAR FIFO:%N/
1731 005000 EF9002:: .ASCIZ /%A CODE IS ASSOCIATED WITH LINE: %D2%A(D)%N/
1732 005057 EF9003:: .ASCIZ /%A CODE IS: %03%A(0)%N/
1733 005113 EF9004:: .ASCIZ /%A %T%A VALUE: %03%A(0)%N/
1734 005150 EF9005:: .ASCIZ /%A %T%A VALUE: NONE%N/
1735 005201 EF9006:: .ASCIZ /%A %T%A %D2%A(D)%N/
1736 005225 EF9010:: .ASCIZ /%A NUMBER OF ERRORS DETECTED ON LINE %D2%A(D) IS %D5%A(D)%N/
1737 005324 EF9016:: .ASCIZ /%A UNEXPECTED %T%A FOR LINE %D2%A(D) IN FIFO AFTER RESET:%N/
1738 005421 EF9017:: .ASCIZ /%A %T%A (WITH ERROR FLAGS) IS %06%A(0)%N/
1739 005475 EF9018:: .ASCII /%A %T%A IN SELFTEST CODE FIFO SLOT FOR LINE %D2/
1740 005555 .ASCIZ /%A(D) AFTER RESET.%N/
1741 005602 EF9019:: .ASCIZ /%A %T%A %06%A(0)%N/
1742 005626 EF9301:: .ASCIZ /%A %T%D2%A(D), BMP CODE REPORTED :%03%A(0)%N/
1743 005704 EF9302:: .ASCIZ /%A OVERFLOW OCCURRED (MORE THAN 31 BMP CODES FOUND IN QUEUE)%N/
1744 : ***** ERROR MESSAGES *****
1745 006004 DR00MG:: .ASCIZ /CSR/
1746 006010 DR02MG:: .ASCIZ /RBUF/
1747 006015 DR04MG:: .ASCIZ /LPR/
1748 006021 DR06MG:: .ASCIZ /STAT/
1749 006026 DR10MG:: .ASCIZ /LNCTRL/
1750 006035 DR12MG:: .ASCIZ /TBUFFAD1/
1751 006046 DR14MG:: .ASCIZ /TBUFFAD2/
1752 006057 DR16MG:: .ASCIZ /TBUFFCT/
1753 006067 EM0103:: .ASCIZ /DEVICE REGISTER ACCESS ERRORS/
1754 006125 EM0201:: .ASCIZ /MASTER RESET (PERFORM SELFTEST) TEST /
1755 006173 EM0202:: .ASCIZ / MASTER RESET BIT DID NOT CLEAR AFTER BOARD RESET./
1756 006257 .ASCIZ / WAITED 5 SECONDS. BIT DEFECTIVE OR FIRMWARE HUNG./
1757 006346 EM0203:: .ASCIZ / MASTER RESET BIT CLEAR IMMEDIATELY AFTER BOARD RESET./
1758 006436 .ASCIZ / BIT DEFECTIVE OR BOARD FIRMWARE ERROR./
1759 006511 EM0204:: .ASCIZ \ MR BIT WENT CLEAR WITHIN 1/2 SECOND OF BOARD RESET.\
1760 006577 .ASCIZ / BIT DEFECTIVE OR SELFTEST WAS (INCORRECTLY) SKIPPED./
1761 006670 EM0301:: .ASCIZ /MASTER RESET (SKIP SELFTEST) TEST /
1762 006733 EM0302:: .ASCIZ / MR BIT CLR WITHIN 10 MILLISECOND AFTER BOARD RESET./
1763 007020 .ASCIZ / BIT DEFECTIVE OR BOARD FIRMWARE ERROR./
1764 007073 EM0303:: .ASCIZ \ MR BIT WENT CLEAR 1/5 TO 5 SECONDS AFTER RESET.\
```

GLOBAL MESSAGE AREA

```

1765 007155 .ASCIZ / SELFTEST DID NOT GET SKIPPED (SHOULD HAVE BEEN SKIPPED)./
1766 007252 EM0401:: .ASCIZ /RBUF REGISTER RX CHARACTER FIELD TEST /
1767 007321 EM0402:: .ASCIZ / IMPROPER CODE FOUND IN RX FIFO AFTER DUT RESET./
1768 007403 .ASCIZ / EXPECTED: SELFTEST CODE, ACTUAL: IMPROPER CODE./
1769 007471 EM0501:: .ASCIZ /RBUF REGISTER ERROR FLAGS FIELD TEST /
1770 007537 EM0502:: .ASCIZ / RX ERROR FLAG(S) FOUND CLEAR ON SELFTEST CODE./
1771 007620 .ASCIZ / EXPECTED: ALL ERROR FLAGS SET, ACTUAL: FLAG(S) CLEAR./
1772 007713 EM0525:: .ASCIZ / RX INTERRUPT(S) RECEIVED WITH RX INTERRUPTS DISABLED./
1773 010003 EM0526:: .ASCIZ / TX INTERRUPT(S) RECEIVED WITH TX INTERRUPTS DISABLED./
1774 010073 EM0601:: .ASCIZ /CSR RX.DATA.AVAIL BIT TEST /
1775 010127 EM0602:: .ASCIZ / RX.DATA.AVAIL BIT FOUND CLEAR AFTER RESET COMPLETION./
1776 010217 .ASCIZ / EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO./
1777 010307 EM0603:: .ASCIZ / RX.DATA.AVAIL BIT COULD NOT BE CLEARED BY PURGING FIFO./
1778 010401 .ASCIZ / 600 CHARS READ FROM FIFO WITHOUT R.D.A BIT CLEARING./
1779 010472 EM0701:: .ASCIZ /RBUF RX.DATA.VALID BIT TEST /
1780 010527 EM0702:: .ASCIZ / RX.DATA.VALID BIT FOUND CLEAR AFTER RESET COMPLETION./
1781 010617 .ASCIZ / EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO./
1782 010707 EM0703:: .ASCIZ / RX.DATA.VALID BIT COULD NOT BE CLEARED BY PURGING FIFO./
1783 011001 .ASCIZ / 600 CHARS READ FROM FIFO WITHOUT R.D.V BIT CLEARING./
1784 011072 EM0801:: .ASCIZ /RBUF RX.LINE.NUMBER FIELD TEST /
1785 011132 EM0802:: .ASCIZ / LINE NUMBER IS WRONG ON A SELFTEST CODE./
1786 011205 EM0901:: .ASCIZ /CHECK FOR BMP CODES TEST/
1787 011236 EM0902:: .ASCIZ /UNEXPECTED BMP CODES FOUND./
1788 011272 EM1001:: .ASCIZ /DIAGNOSTIC FAIL (SKP SELFTEST) TEST/
1789 011336 EM1002:: .ASCIZ / SKIP SELF-TEST TOOK TOO LONG TO COMPLETE, > 50 MS./
1790 011423 EM1003:: .ASCIZ / SKIP SELF-TEST COMPLETED TOO SOON, < 10 MS./
1791 011501 EM1101:: .ASCIZ /SKIP SELF-TEST TEST/
1792 011525 EM1201:: .ASCIZ /SELF-TEST TEST/
1793 011544 EM1202:: .ASCIZ / SELF-TEST TOOK TOO LONG TO COMPLETE, > 3 SECONDS./
1794 011630 EM1203:: .ASCIZ \ SELF-TEST COMPLETED TOO SOON, < 1,2 SECOND.\
1795 011706 EM1204:: .ASCIZ / SELF-TEST DID NOT EXECUTE/
1796 011742 EM1205:: .ASCIZ / DIAG FAIL BIT BAD/
1797 011766 EM1301:: .ASCIZ /FAIL SELF-TEST TEST/
1798 012012 EM1302:: .ASCIZ / SELF-TEST ERROR REPORTING BAD/
1799 012051 EM1401:: .ASCIZ /ROM VERSION_NUMBER TEST/
1800 012101 EM1402:: .ASCIZ / FIFO EMPTY, ONE OR MORE ROM VERSION_NUMBERS MISSING/
1801 012167 EM1403:: .ASCIZ / ROM VERSION_NUMBER FOUND OUT OF SEQUENCE/
1802 012242 EM1404:: .ASCIZ / ONE OR MORE ROM VERSION_NUMBERS MISSING/
1803 012314 EM1405:: .ASCIZ / PROC_1/
1804 012327 EM1406:: .ASCIZ / PROC_2/
1805 012342 EM1407:: .ASCIZ /NOT FOUND/
1806 012354 EM1408:: .ASCIZ /FOUND/
1807 012362 EM1601:: .ASCIZ /TIMEOUT OCCURRED WAITING FOR MASTER RESET TO CLEAR/
1808 012445 EM1604:: .ASCIZ \DEVICE REGISTER WORD READ/WRITE TEST \
1809 012513 EM1701:: .ASCIZ \DEVICE REGISTER WORD READ/MODIFY/WRITE TEST \
1810 012570 EM1801:: .ASCIZ \DEVICE REGISTER BYTE READ/WRITE TEST \
1811 012636 EM1901:: .ASCIZ \DEVICE REGISTER BYTE READ/MODIFY/WRITE TEST \
1812 012713 EM2001:: .ASCIZ /DEVICE STAT REGISTER ID BIT TEST /
1813 012755 EM2002:: .ASCIZ /ID BIT BAD. EXPECTED: CLEAR, ACTUAL: SET./
1814 013030 EM2101:: .ASCIZ \NO TX_DATA_VALID/NO TX_ACTION TEST\
1815 013073 EM2102:: .ASCIZ / TX_ACTION FOUND AFTER INVALID DATA WORD WRITTEN TO LINE: /
1816 013167 EM2201:: .ASCIZ \TX_DATA_VALID/TX_ACTION TEST\
1817 013224 EM2202:: .ASCIZ / NO TX_ACTION FOUND AFTER VALID DATA WORD TX'D ON LINE: /
1818 013316 EM2203:: .ASCIZ / INCORRECT LINE NUMBER FOUND WITH TX_ACT AFTER DATA TX'D ON LINE : /
1819 013423 EM2301:: .ASCIZ /TX_ENABLE (INACTIVE) BIT TEST/
1820 013461 EM2302:: .ASCIZ / TX_ENABLE BIT BAD ON LINE: /
1821 013517 EM2401:: .ASCIZ /TX_ENABLE (ACTIVE) BIT TEST/

```

GLOBAL MESSAGE AREA

```

1822 013553 EM2601:: .ASCIZ /RECEIVE INTERRUPT TEST /
1823 013603 EM2602:: .ASCIZ / NO RX INT GENERATED (DATA_VALID SET, RX INTS ENABLED)./
1824 013674 EM2603:: .ASCIZ / NO RX INT GENERATED (NO CODES IN FIFO AFTER RESET)./
1825 013762 EM2604:: .ASCIZ / NO RX INT GENERATED (RX_DATA_AVAIL CLR, RX INTS ENABLED)./
1826 014056 EM2605:: .ASCIZ / RX INTERRUPT GENERATED WITH RX_DATA_AVAIL CLEAR./
1827 014141 EM2606:: .ASCIZ /TRANSMIT INTERRUPT TEST ERROR:/
1828 014200 EM2607:: .ASCIZ / TX_ACTION SET REPEATEDLY AFTER BOARD RESET, NO DATA SENT./
1829 014274 EM2608:: .ASCIZ / TX_ACTION STUCK SET AFTER BOARD RESET./
1830 014345 EM2609:: .ASCIZ / TX INTERRUPT GENERATED WITH TX_ACTION CLEAR./
1831 014424 EM2610:: .ASCIZ / NO TX INTERRUPT WITH TX_ACTION SET AND TX INTS ENABLED./
1832 014516 EM2611:: .ASCIZ / TX ACTION NOT SET AFTER CHARS SENT ON ALL LINES./
1833 014601 EM3001:: .ASCIZ /INTERRUPT BR LEVEL TEST /
1834 014632 EM3002:: .ASCIZ / NO RX_DATA_AVAIL FROM SELFTEST CODES IN FIFO AFTER RESET./
1835 014726 EM3003:: .ASCIZ / TX INTERRUPT GENERATED AT WRONG BR LEVEL:/
1836 015002 EM3004:: .ASCIZ / RX INTERRUPT GENERATED AT WRONG BR LEVEL:/
1837 015056 EM3005:: .ASCIZ / TX INTERRUPT GIVEN PRECEDENCE OVER SIMULTANEOUS RX INT./
1838 015150 EM3101:: .ASCIZ /DIAGNOSTIC FIELD (BMP) TEST/
1839 015204 EM3102:: .ASCIZ / DIAGNOSTIC FIELD BAD ON LINE: /
1840 015245 EM9009:: .ASCIZ /EXPECTED OR CORRECT/
1841 015271 EM9010:: .ASCIZ /ACTUAL OR MEASURED /
1842 015315 EM9014:: .ASCIZ /SUMMARY REPORTS FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:/
1843 015411 EM9017:: .ASCII / FIFO WILL NOT PURGE (DATA_VALID STUCK SET),/
1844 015466 .ASCIZ / REMAINDER OF TEST SKIPPED./
1845 015522 EM9018:: .ASCIZ /NO CODE/
1846 015532 EM9019:: .ASCIZ /NON-SELFTEST/
1847 015547 EM9020:: .ASCIZ /SELFTEST ERROR CODE/
1848 015573 EM9022:: .ASCIZ /DATA CHARACTER/
1849 015612 EM9023:: .ASCIZ /MODEM STATUS CODE/
1850 015634 EM9024:: .ASCIZ /SELFTEST CODE/
1851 015652 EM9026:: .ASCIZ / LPR CONTENTS: /
1852 015676 EM9301:: .ASCIZ /BMP CODE REPORT/
1853 015716 EM9302:: .ASCIZ /BMP CODE FOUND IN TEST /
1854 015746 EM9303:: .ASCIZ /THE LAST BMP CODE WAS FOUND IN TEST /
1855 016013 EM9304:: .ASCIZ /UNEXPECTED BMP CODES FOUND DURING THIS PASS/
1856 .EVEN
1857 .LIST BIN

```

GLOBAL MESSAGE AREA

1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875

```
*****  
: FVTSKL2.P11  
:*****
```

.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.  
;--
```

GLOBAL ERROR REPORTING ROUTINE

- ER0101 -

```

1877 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0101 -
1878 ;*****
1879 ;* This is an error reporting subroutine which prints additional error
1880 ;* information if an error is detected in TEST 1 (Register Address
1881 ;* Access Test). This subroutine reports the type of access (Read or
1882 ;* Write or both) which caused a bus time out trap (004 trap).
1883 ;* A message indicating that the DHV may be at the wrong Q bus address
1884 ;* is also printed.
1885 ;*
1886 ;* INPUTS: R5 - Error flag word.
1887 ;* If bit 0 is set, a read error occurred.
1888 ;* If bit 1 is set, a write error occurred.
1889 ;*
1890 ;* OUTPUTS: Messages are printed at the operator console.
1891 ;*
1892 ;* CALLING SEQUENCE: Include the label "ER0101" as the message pointer
1893 ;* parameter in the DRS error report macro call.
1894 ;*
1895 ;* COMMENTS:
1896 ;*
1897 ;* SUBORDINATE ROUTINES USED: None.
1898 ;*****
1899
1900 016070 BGNMSG ER0101
1901 016070 SAVE ;SAVE THE GPR CONTENTS. ER0101::
1902 016070 004567 165702 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
1903 016074 032705 000001 BIT #BIT0,R5 ;TEST FOR READ ERROR.
1904 016100 001410 BEQ 2$ ;SKIP READ ERROR MSG IF NO READ ERROR.
1905 016102 PRINTB #MSG1 ;PRINT READ ERROR MESSAGE.
1906 016102 012746 016174 MOV #MSG1,-(SP)
1907 016106 012746 000001 MOV #1,-(SP)
1908 016112 010600 MOV SP,R0
1909 016114 104414 TRAP C$PNTB
1910 016116 062706 000004 ADD #4,SP
1911 016122 032705 000002 2$: BIT #BIT1,R5 ;TEST FOR WRITE ERROR.
1912 016126 001410 BEQ 4$ ;SKIP WRITE ERROR MSG IF NO WRITE ERROR.
1913 016130 PRINTB #MSG2 ;PRINT WRITE ERROR MESSAGE.
1914 016130 012746 016252 MOV #MSG2,-(SP)
1915 016134 012746 000001 MOV #1,-(SP)
1916 016140 010600 MOV SP,R0
1917 016142 104414 TRAP C$PNTB
1918 016144 062706 000004 ADD #4,SP
1919 016150 4$: PRINTX #MSG3 ;SUGGEST THAT DHV MAY BE AT WRONG ADDRESS.
1920 016150 012746 016331 MOV #MSG3,-(SP)
1921 016154 012746 000001 MOV #1,-(SP)
1922 016160 010600 MOV SP,R0
1923 016162 104415 TRAP C$PNTX
1924 016164 062706 000004 ADD #4,SP
1925 016170 PASS ;RESTORE THE GPR CONTENTS.
1926 016170 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
1927 016172 ENDMSG
1928 016172 104423 L10002: TRAP C$MSG
1929 016174 045 101 102 MSG1:: .ASCIZ /*ABUS TIME OUT TRAP CAUSED BY READ ATTEMPT.*N/

```

GLOBAL ERROR REPORTING ROUTINE

- ER0101 -

	016177	125	123	040	
	016202	124	111	115	
	016205	105	055	117	
	016210	125	124	040	
	016213	124	122	101	
	016216	120	040	103	
	016221	101	125	123	
	016224	105	104	040	
	016227	102	131	040	
	016232	122	105	101	
	016235	104	040	101	
	016240	124	124	105	
	016243	115	120	124	
	016246	056	045	116	
	016251	000			
1914	016252	045	101	102	MSG2:: .ASCIZ /*ABUS TIME OUT TRAP CAUSED BY WRITE ATTEMPT.*N/
	016255	125	123	040	
	016260	124	111	115	
	016263	105	055	117	
	016266	125	124	040	
	016271	124	122	101	
	016274	120	040	103	
	016277	101	125	123	
	016302	105	104	040	
	016305	102	131	040	
	016310	127	122	111	
	016313	124	105	040	
	016316	101	124	124	
	016321	105	115	120	
	016324	124	056	045	
	016327	116	000		
1915	016331	045	101	104	MSG3:: .ASCIZ /*ADHV MAY BE AT THE WRONG Q-BUS ADDRESS.*N*N/
	016334	110	126	040	
	016337	115	101	131	
	016342	040	102	105	
	016345	040	101	124	
	016350	040	124	110	
	016353	105	040	127	
	016356	122	117	116	
	016361	107	040	121	
	016364	055	102	125	
	016367	123	040	101	
	016372	104	104	122	
	016375	105	123	123	
	016400	056	045	116	
	016403	045	116	000	
1916					
1917					.EVEN

GLOBAL ERROR REPORTING ROUTINE

- ER0201 -

```

1919 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0201 -
1920 ;*****
1921 ;* This is an error reporting subroutine which prints 2 contiguous
1922 ;* ASCII error messages. The address of the first message is passed
1923 ;* as an input parameter and the address of the second is found by
1924 ;* searching for the end of the first message.
1925 ;*
1926 ;* INPUTS: R1 - Address of the first message to print.
1927 ;*
1928 ;* OUTPUTS: A messages is printed at the operator console.
1929 ;*
1930 ;* CALLING SEQUENCE: Load the address of the first message in R1.
1931 ;* Include the label "ER0201" as the message pointer
1932 ;* parameter in the Diag Super error report macro call.
1933 ;*
1934 ;* COMMENTS: The message is printed as Basic error information.
1935 ;* The second message should follow the first one in the program
1936 ;* memory. Each message should be defined using .ASCIZ
1937 ;*
1938 ;* SUBORDINATE ROUTINES USED: None.
1939 ;*****
1940
1941 016406 BGNMSG ER0201
1942 016406 ER0201::
016406 SAVE ;SAVE THE GPR CONTENTS.
016406 004567 165364 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
1943
1944 016412 010102 MOV R1,R2
1945 016414 105722 2$: TSTB (R2)+ ;CHECK FOR A ZERO BYTE FND OF MESSAGE).
1946 016416 001376 BNE 2$ ;LOOP UNTIL NEXT MESSAGE IS FOUND.
1947
1948 016420 PRINTB #EF0503,R1 ;PRINT THE FIRST MESSAGE.
016420 010146 MOV R1,-(SP)
016422 012746 004134 MOV #EF0503,-(SP)
016426 012746 000002 MOV #2,-(SP)
016432 010600 MOV SP,R0
016434 104414 TRAP C$PNTB
016436 062706 000006 ADD #6,SP
1949 016442 PRINTB #EF0503,R2 ;PRINT THE SECOND MESSAGE.
016442 010246 MOV R2,-(SP)
016444 012746 004134 MOV #EF0503,-(SP)
016450 012746 000002 MOV #2,(SP)
016454 010600 MOV SP,R0
016456 104414 TRAP C$PNTB
016460 062706 000006 ADD #6,SP
1950
1951 016464 PASS ;RESTORE THE GPR CONTENTS.
016464 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
1952
1953 016466 ENDMMSG
016466 L10003:
016466 104423 TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER0503 -

```

1955 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0503 -
1956 ;*****
1957 ;* This is an error reporting subroutine which prints an additional error
1958 ;* message whose address is passed as an input parameter.
1959 ;*
1960 ;* INPUTS: R1 - Address of the message to print.
1961 ;*
1962 ;* OUTPUTS: A messages is printed at the operator console.
1963 ;*
1964 ;* CALLING SEQUENCE: Load the address of the message in R1.
1965 ;* Include the label "ER0503" as the message pointer
1966 ;* parameter in the Diag Super error report macro call.
1967 ;*
1968 ;* COMMENTS: The message is printed as Basic error information.
1969 ;*
1970 ;* SUBORDINATE ROUTINES USED: None.
1971 ;*****
1972
1973 016470 BGNMSG ER0503
1974 016470 ER0503::
1975 016470 PRINTB #EF0503,R1 ;PRINT THE MESSAGE.
1976 016470 010146 MOV R1,-(SP)
1977 016472 012746 004134 MOV #EF0503,-(SP)
016476 012746 000002 MOV #2,-(SP)
016502 010600 MOV SP,R0
016504 104414 TRAP C$PNTB
016506 062706 000006 ADD #6,SP
1976 016512 ENDMSG
1977 016512 L10004:
016512 104423 TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER0504 -

```

1979 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0504 -
1980 ;*****
1981 ;* This is an error reporting subroutine which prints additional error
1982 ;* messages when illegal interrupts are received.
1983 ;*
1984 ;* INPUTS R1 Address of the message to print.
1985 ;* R2 - Number of illegal interrupts received.
1986 ;*
1987 ;* OUTPUTS: Messages are printed at the operator console.
1988 ;*
1989 ;* CALLING SEQUENCE: Load the address of the message in R1.
1990 ;* Load the number of illegal ints in R2.
1991 ;* Include the label "ER0504" as the message pointer
1992 ;* parameter in the Diag Super error report macro call.
1993 ;*
1994 ;* COMMENTS:
1995 ;*
1996 ;* SUBORDINATE ROUTINES USED: None.
1997 ;*****
1998
1999 016514 BGNMSG ER0504
016514 ER0504::
2000
2001 016514 PRINTB #EF0503,R1 ;PRINT THE FIRST LINE OF THE MESSAGE.
016514 010146 MOV R1, (SP)
016516 012746 004134 MOV #EF0503, (SP)
016522 012746 000002 MOV #2, (SP)
016526 010600 MOV SP,R0
016530 104414 TRAP C$PNTB
016532 062706 000006 ADD #6,SP
2002 016536 PRINTX #EF0505,R2 ;PRINT THE NUMBER OF INTS RECEIVED.
016536 010246 MOV R2, -(SP)
016540 012746 004141 MOV #EF0505, (SP)
016544 012746 000002 MOV #2, -(SP)
016550 010600 MOV SP,R0
016552 104415 TRAP C$PNTX
016554 062706 000006 ADD #6,SP
2003
2004 016560 ENDMSG
016560 L10005:
016560 104423 TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER1401

```

2006 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER1401 -
2007 ;*****
2008 ;* This is an error reporting subroutine which prints additional error
2009 ;* information if an error is detected in the Rom version test.
2010 ;* This subroutine analyses the input parameters which contain the
2011 ;* ROM version numbers for PROC_1 and PROC_2 and reports the appropriate
2012 ;* error message to the operator.
2013 ;*
2014 ;* INPUTS: R1 - Contains the address of the first message to be reported.
2015 ;* R3 - Contains the ROM version number of PROC_1.
2016 ;* R4 - Contains the ROM version number of PROC_2.
2017 ;*
2018 ;* OUTPUTS: Basic and extended error messages are reported at the
2019 ;* operators console.
2020 ;*
2021 ;* CALLING SEQUENCE: Include the label "ER1401" as the message pointer
2022 ;* parameter in the DRS error report macro call.
2023 ;*
2024 ;* COMMENTS:
2025 ;*
2026 ;* SUBORDINATE ROUTINES USED: None.
2027 ;*****
2028
2029 016562 BGNMSG ER1401
2029 016562 ER1401::
2030
2031 016562 PRINTB #EF0503,R1 ;REPORT THE ERROR MESSAGE PASSED IN.
016562 010146 MOV R1,-(SP)
016564 012746 004134 MOV #EF0503,-(SP)
016570 012746 000002 MOV #2,-(SP)
016574 010600 MOV SP,R0
016576 104414 TRAP C$PNTB
016600 062706 000006 ADD #6,SP
2032
2033 ;+
2034 ; Determine which Rom version number(s) are missing.
2035 ;-
2036
2037 016604 012705 000143 MOV #99.,R5 ;GET INVALID ROM NUMBER.
2038 016610 012701 012314 MOV #EM1405,R1 ;SELECT PROC_1 MESSAGE.
2039 016614 012702 012342 MOV #EM1407,R2 ;SELECT THE "NOT FOUND" MESSAGE.
2040 016620 120305 CMPB R3,R5 ;CHECK PROC_1 ROM VERSION N' 1BER.
2041 016622 001402 BEQ 2$ ;GO REPORT PROC_1 CODE NOT FOUND.
2042 016624 012702 012354 MOV #EM1408,R2 ;SELECT "FOUND" MESSAGE.
2043 016630 004767 000026 2$: JSR PC,50$ ;GO REPORT MESSAGE.
2044
2045 016634 012701 012327 MOV #EM1406,R1 ;SELECT PROC_2 MESSAGE.
2046 016640 012702 012342 MOV #EM1407,R2 ;SELECT THE "NOT FOUND" MESSAGE.
2047 016644 120405 CMPB R4,R5 ;CHECK PROC_2 ROM VERSION NUMBER.
2048 016646 001402 BEQ 4$ ;GO REPORT PROC_2 CODE NOT FOUND.
2049 016650 012702 012354 MOV #EM1408,R2 ;SELECT "FOUND" MESSAGE.
2050 016654 004767 000002 4$: JSR PC,50$ ;GO REPORT THE MESSAGE.
2051 016660 000413 BR 60$ ;EXIT.
2052
2053 016662 50$: PRINTX #EF1402,R1,R2 ;REPORT THE MESSAGE.
016662 010246 MOV R2,-(SP)
016664 010146 MOV R1,(SP)

```

GLOBAL ERROR REPORTING ROUTINE

- ER1401 -

	016666	012746	004316
	016672	012746	000003
	016676	010600	
	016700	104415	
	016702	062706	000010
2054	016706	000207	
2055	016710		
	016710		
	016710	104423	

60‡: RTS PC
 ENDMSG

;RETURN.

MOV #EF1402, -(SP)
MOV #3, -(SP)
MOV SP, R0
TRAP C‡PNTX
ADD #10, SP

L10006:
 TRAP C‡MSG

GLOBAL ERROR REPORTING ROUTINE

- ER1601 -

2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087

016712
016712
016712 016304 002304
016716
016716 010546
016720 010446
016722 012746 004503
016726 012746 000003
016732 010600
016734 104414
016736 062706 000010
016742
016742 010246
016744 012746 004377
016750 012746 000002
016754 010600
016756 104415
016760 062706 000006
016764
016764 010146
016766 012746 004441
016772 012746 000002
016776 010600
017000 104415
017002 062706 000006
017006
017006
017006 104423

```

.SBTTL GLOBAL ERROR REPORTING ROUTINE - ER1601 -
;*****
;* This an error reporting subroutine which prints additional error
;* information if an error is detected in one of + device register
;* access tests.
;* This subroutine reports the actual and expected in the device
;* register(s) which is(are) in faulty.
;*
;* INPUTS: R1 - Actual data (unused bits set to 0).
;* R2 - Expected data (unused bits set to 0).
;* R3 - Offset (in bytes) to the register being tested.
;* R5 - Line number of register being tested.
;* RMATBB - Label at base of register message address table.
;*
;* OUTPUTS: Messages are printed at the operators console.
;*
;* CALLING SEQUENCE: Include the label "ER1601" as the message pointer
;* parameter in the DRS error report macro call.
;*
;* COMMENTS:
;*
;* SUBORDINATE ROUTINES CALLED: NONE
;*****
          BGNMSG ER1601
                                ER1601::
2082 016712 016304 002304          MOV    RMATBB(R3),R4    ;FETCH ADDRESS OF REGISTER NAME MESSAGE.
2084 016716                                PRINTB #EF1604,R4,R5    ;REPORT BASIC MESSAGE (REG NAME AND LINE #).
                                MOV     R5,-(SP)
                                MOV     R4,-(SP)
                                MOV     #EF1604,-(SP)
                                MOV     #3,-(SP)
                                MOV     SP,R0
                                TRAP    C$PNTB
                                ADD     #10,SP
2085 016742                                PRINTX #EF1602,R2      ;PRINT THE EXPECTED DATA.
                                MOV     R2,-(SP)
                                MOV     #EF1602,-(SP)
                                MOV     #2,-(SP)
                                MOV     SP,R0
                                TRAP    C$PNTX
                                ADD     #6,SP
2086 016764                                PRINTX #EF1603,R1      ;PRINT THE ACTUAL DATA.
                                MOV     R1,-(SP)
                                MOV     #EF1603,-(SP)
                                MOV     #2,-(SP)
                                MOV     SP,R0
                                TRAP    C$PNTX
                                ADD     #6,SP
2087 017006                                ENDMSG
                                L10007:
                                TRAP    C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER1603 -

```

2089 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER1603 -
2090 ;*****
2091 ;* This error reporting routine is used to print out a basic error
2092 ;* message, along with a message informing the operator which test is
2093 ;* about to be aborted.
2094 ;*
2095 ;* INPUTS: R1 - Contains the address of the message to be printed.
2096 ;* ERRMSG - Contains the address of the message that indicates
2097 ;* the test that is being performed, eg DMA, BREAK etc.
2098 ;*
2099 ;* OUTPUTS: Messages are printed at the operators console.
2100 ;* "testname TEST ABORTED"
2101 ;*
2102 ;* CALLING SEQUENCE: Include the label "ER1603" as the message pointer
2103 ;* parameter in the DRS error report macro call.
2104 ;*
2105 ;* COMMENTS:
2106 ;*
2107 ;*
2108 ;* SUBORDINATE ROUTINES CALLED: None.
2109 ;*****
2110 017010 BGNMSG ER1603
2111 017010 ER1603::
017010 004567 164762 SAVE JSR R5,PREG05 ;SAVE THE CONTENTS OF THE GPRS.
;CALL REGISTER SAVE SUBRT.
2112 017014 PRINTB #EF0503,R1 ;PRINT BASIC MESSAGE ON OPERATORS CONSOLE.
2113 017014 010146 MOV R1,-(SP)
017016 012746 004134 MOV #EF0503,(SP)
017022 012746 000002 MOV #2,-(SP)
017026 010600 MOV SP,R0
017030 104414 TRAP C$PNTB
017032 062706 000006 ADD #6,SP
2114 017036 016702 164730 MOV ERRMSG,R2 ;GET THE "TEST MESSAGE".
2115 017042 PRINTB #EF1601,R2 ;PRINT "TEST ABORTED" MESSAGE.
2116 017042 010246 MOV R2,-(SP)
017044 012746 004353 MOV #EF1601,-(SP)
017050 012746 000002 MOV #2,(SP)
017054 010600 MOV SP,R0
017056 104414 TRAP C$PNTB
017060 062706 000006 ADD #6,SP
2117 017064 PASS ;RESTORE THE CONTENTS OF THE GPRS.
2118 017064 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2119 017066 ENDMSG
017066 104423 L10010: TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER3001 -

2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140

```

.SBTTL GLOBAL ERROR REPORTING ROUTINE - ER3001 -
;*****
;* This is an error reporting subroutine which is intended for use in the
;* Interrupt BR Level Test. It reports additional information when an
;* interrupt has occurred at the wrong BR level.
;*
;* INPUTS: R1 - Address of message to print first.
;* R4 - BR level at which the int request occurred.
;* R5 - Expected or correct BR level for the DUT.
;*
;* OUTPUTS: A messages is printed at the operator console.
;*
;* CALLING SEQUENCE: Include the label "ER3001" as the message pointer
;* parameter in the Diag Super error report macro call.
;*
;* COMMENTS: The message is printed as Basic and Extended error information.
;*
;* SUBORDINATE ROUTINES USED: None.
;*****

```

2141 017070
017070

BGNMSG ER3001

ER3001::

2142

2143 017070
017070 010146
017072 012746 004134
017076 012746 000002
017102 010600
017104 104414
017106 062706 000006

PRINTB #EF0503,R1 ;PRINT THE FIRST LINE OF THE MESSAGE.

```

MOV R1,-(SP)
MOV #EF0503,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTB
ADD #6,SP

```

2144 017112
017112 010546
017114 012746 004600
017120 012746 000002
017124 010600
017126 104415
017130 062706 000006

PRINTX #EF3001,R5 ;REPORT EXPECTED BR LEVEL.

```

MOV R5,(SP)
MOV #EF3001,-(SP)
MOV #2,(SP)
MOV SP,R0
TRAP C$PNTX
ADD #6,SP

```

2145 017134
017134 010446
017136 012746 004647
017142 012746 000002
017146 010600
017150 104415
017152 062706 000006

PRINTX #EF3002,R4 ;REPORT ACTUAL BR LEVEL.

```

MOV R4,-(SP)
MOV #EF3002,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTX
ADD #6,SP

```

2146
2147 017156
017156
017156 104423

ENDMSG

L10011:

TRAP C\$MSG

GLOBAL ERROR REPORTING ROUTINE

- ER9004 -

2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169

```

.SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9004 -
;*****
;* This is an error reporting subroutine which reports error summaries
;* for lines which have exceeded the specified maximum number of
;* individual reception errors.
;*
;* INPUTS: R1 - Address of message to print first.
;* ERCNTB - Label at base of line error counters table.
;* ERSMRF - "Report error summary for line" flags.
;*
;* OUTPUTS: A message is printed at the operator console.
;*
;* CALLING SEQUENCE: Include the label "ER9004" as the message pointer
;* parameter in the Diag Super error report macro call.
;*
;* COMMENTS: The message is printed as Basic and Extended error information.
;* The contents of GPR's R2, R3, R4, and R5 are destroyed.
;*
;* SUBORDINATE ROUTINES USED: None.
;*****

```

2170 017160
017160

BGNMSG ER9004

ER9004::

2171
2172

017160 012746 015315
017164 012746 004134
017170 012746 000002
017174 010600
017176 104414
017200 062706 000006
2173 017204 005002
2174 017206 016703 163250
2175 017212 005004
2176 017214 000241
2177 017216 006003
2178 017220 103013
2179 017222

```

PRINTB #EF0503,#EM9014 ;REPORT THE SECONDARY ERROR MESSAGE.
MOV #EM9014,(SP)
MOV #EF0503,(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C$PNTB
ADD #6,SP
CLR R2 ;CLEAR THE LINE COUNTER.
MOV ERSMRF,R3 ;GET THE ERROR SUMMARY FLAGS.
CLR R4 ;CLEAR "LINE COUNTER TIMES 2" OFFSET.
2$: CLC ;CLEAR THE CARRY FOR THE FOLLOWING ROTATE.
ROR R3 ;SHIFT ANOTHER ERROR SUMMARY FLAG INTO CARRY.
BCC 4$ ;SKIP PRINTING MESSAGE IF FLAG FOR LINE CLEAR.
PRINTX #EF9010,R2,ERCNTB(R4)

```

017222 016446 002464
017226 010246
017230 012746 005225
017234 012746 000003
017240 010600
017242 104415
017244 062706 000010

```

MOV ERCNTB(R4),(SP)
MOV R2,-(SP)
MOV #EF9010,(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C$PNTX
ADD #10,SP

```

2180 017250 012405
2181 017252 005202
2182 017254 005703
2183 017256 001356
2184

```

4$: MOV (R4)+,R5 ;INCREMENT THE LINE OFFSET BY 2.
INC R2 ;INCREMT THE LINE COUNTER.
TST R3 ;CHECK THE ERROR SUMMARY FLAGS.
BNE 2$ ;IF MORE FLAGS SET, LOOP TO DO OTHER LINES.

```

2185 017260
017260
017260 104423

ENDMSG

L10012:

TRAP C\$MSG

GLOBAL ERROR REPORTING ROUTINE

- ER9007 -

```

2187 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9007 -
2188 ;*****
2189 ;* This is an error reporting subroutine which is used to report that
2190 ;* something other than a selftest code was found in a selftest code
2191 ;* FIFO slot during the removal of the selftest codes from the FIFO.
2192 ;* This routine is used by the RSTRPT routine.
2193 ;*
2194 ;* INPUTS: R1 - Address of error message qualifier string.
2195 ;* R2 - Incorrect code as read from the selftest code FIFO slot.
2196 ;* R3 - Line number associated with the selftest FIFO slot.
2197 ;*
2198 ;* OUTPUTS: A message is printed at the operator console.
2199 ;*
2200 ;* CALLING SEQUENCE: Include the label "ER9007" as the message pointer
2201 ;* parameter in the Diag Super error report macro call.
2202 ;*
2203 ;* COMMENTS: The message is printed as Basic and Extended error information.
2204 ;*
2205 ;* SUBORDINATE ROUTINES USED: None.
2206 ;*****
2207
2208 017262 BGNMSG ER9007
2209 017262 ER9007::
2210 017262 042703 177760 BIC #177760,R3 ;REMOVE ALL BUT LINE # BITS FROM LINE # WORD.
2211 017266 PRINTB #EF9018,R1,R3 ;REPORT SECONDARY ERROR MESSAGE.
2212 017266 010346 MOV R3,(SP)
2213 017270 010146 MOV R1,(SP)
2214 017272 012746 005475 MOV #EF9018,(SP)
2215 017276 012746 000003 MOV #3,-(SP)
2216 017302 010600 MOV SP,R0
2217 017304 104414 TRAP C$PNTB
2218 017306 062706 000010 ADD #10,SP
2219 017312 PRINTX #EF9017,R1,R2 ;REPORT THE ACTUAL INCORRECT CODE.
2220 017312 010246 MOV R2,-(SP)
2221 017314 010146 MOV R1,(SP)
2222 017316 012746 005421 MOV #EF9017,-(SP)
2223 017322 012746 000003 MOV #3,(SP)
2224 017326 010600 MOV SP,R0
2225 017330 104415 TRAP C$PNTX
2226 017332 062706 000010 ADD #10,SP
2227
2228 017336 ENDMSG
2229 017336 L10013:
2230 017336 104423 TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER9008 -

2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235

2236
2237
2238
2239
2240
2241
2242
2243
2244

2245

2246
2247

017340
017340

010203
000303
042703 177760

010346
010146
012746 005324
012746 000003
010600
104414
062706 000010

010246
010146
012746 005421
012746 000003
010600
104415
062706 000010

017420
017420
017420 104423

```
.SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9008 -
;*****
;* This is an error reporting subroutine which is used to report that
;* an unexpected code or character has been found in the DUT receive
;* character FIFO.
;*
;* INPUTS: R1 - Address of partial error message string.
;* R2 - Incorrect code as read from the selftest code FIFO slot.
;*
;* OUTPUTS: A message is printed at the operator console.
;*
;* CALLING SEQUENCE: Include the label "ER9008" as the message pointer
;* parameter in the Diag Super error report macro call.
;*
;* COMMENTS: The message is printed as Basic and Extended error information.
;*
;* SUBORDINATE ROUTINES USED: None.
;*****
```

BGNMSG ER9008

ER9008::

```
;+
; Extract the line number from the incorrect code or character which was read
; from the selftest code FIFO slot.
;-
```

```
MOV R2,R3
SWAB R3
BIC #177760,R3 ;CALCULATE LINE NUMBER OF CODE.
PRINTB #EF9016,R1,R3 ;REPORT TYPE OF INCORRECT CODE FOUND.
MOV R3,-(SP)
MOV R1,-(SP)
MOV #EF9016,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C$PNTB
ADD #10,SP
PRINTX #EF9017,R1,R2 ;REPORT THE ACTUAL INCORRECT CODE.
MOV R2,(SP)
MOV R1,-(SP)
MOV #EF9017,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C$PNTX
ADD #10,SP
```

ENDMSG

L10014: TRAP C\$MSG

GLOBAL ERROR REPORTING ROUTINE

- ER9101 -

```

2249 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9101 -
2250 ;*****
2251 ;* This is a general error reporting subroutine which reports a message
2252 ;* which takes a single, 2 digit decimal argument after the end of an
2253 ;* ASCII message.
2254 ;*
2255 ;* INPUTS: R1 - Value to be printed after msg as 2 decimal digits.
2256 ;* R2 - Address of message to print first.
2257 ;*
2258 ;* OUTPUTS: A messages is printed at the operator console.
2259 ;*
2260 ;* CALLING SEQUENCE: Include the label "ER9101" as the message pointer
2261 ;* parameter in the Diag Super error report macro call.
2262 ;*
2263 ;* COMMENTS: The message is printed as Basic error information.
2264 ;*
2265 ;* SUBORDINATE ROUTINES USED: None.
2266 ;*****
2267
2268 017422 BGNMSG ER9101
2269 017422 ER9101::
2270 017422 PRINTB #EF9006,R2,R1 ;REPORT THE STRING FOLLOWED BY THE NUMBER.
2271 017422 010146 MOV R1,-(SP)
2272 017422 010246 MOV R2,-(SP)
017426 012746 005201 MOV #EF9006,(SP)
017432 012746 000003 MOV #3,-(SP)
017436 010600 MOV SP,R0
017440 104414 TRAP C$PNTB
017442 062706 000010 ADD #10,SP
2271 017446
2272 017446 ENDMSG
017446 104423 L10015: TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER9301 -

```

2274 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9301 -
2275 ;*****
2276 ;* This is an error reporting subroutine which prints any BMP codes
2277 ;* that are found in the BMP code queue, together with the the number of
2278 ;* the test that was executing at the time the BMP code was logged.
2279 ;*
2280 ;* INPUTS: R1 The address of the first message to be reported.
2281 ;* R2 - The address of the next empty cell in the queue.
2282 ;*
2283 ;* OUTPUTS: The test number followed by the BMP code are printed at the
2284 ;* operator console.
2285 ;*
2286 ;* CALLING SEQUENCE: Include the label "ER9301" as the message pointer
2287 ;* parameter in the Diag Super error report macro call.
2288 ;*
2289 ;* COMMENTS: The message is printed as Basic error information.
2290 ;*
2291 ;* SUBORDINATE ROUTINES USED: None.
2292 ;*****
2293
2294 017450 BGNMSG ER9301
2295 017450 ER9301::
017450 SAVE ;SAVE THE GPRS ON THE STACK.
017450 004567 164322 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2296
2297 017454 PRINTB #EF0503,R1 ;REPORT UNEXPECTED BMP CODES FOUND.
017454 010146 MOV R1,-(SP)
017456 012746 004134 MOV #EF0503,-(SP)
017462 012746 000002 MOV #2,-(SP)
017466 010600 MOV SP,R0
017470 104414 TRAP C$PNTB
017472 062706 000006 ADD #6,SP
2298 017476 012703 002526 MOV #BMPCQB,R3 ;GET THE START ADDRESS OF THE BMP CODE QUEUE.
2299 017502 012705 015716 MOV #EM9302,R5 ;GET THE MESSAGE TO BE REPORTED.
2300 017506 012301 2$: MOV (R3)+,R1 ;GET THE NUMBER OF THE TEST THAT WAS EXECUTING.
2301 017510 012304 MOV (R3)+,R4 ;GET BMP CODE THAT WAS REPORTED OFF THE QUEUE.
2302 017512 004767 000056 JSR PC,50$ ;GO REPORT THE BMP CODE.
2303 017516 020302 CMP R3,R2 ;CHECK IF ALL CODES HAVE BEEN REPORTED.
2304 017520 103772 BLO 2$ ;IF IT IS NOT THE LAST BMP CODE THEN LOOP.
2305 ;+
2306 ; Check if overflow has occurred.
2307 ; The conditions for overflow are: the pointer contains the address of the
2308 ; last cell in the queue, and a bmp code has already been written into that
2309 ; cell.
2310 ;-
2311 017522 020227 002722 CMP R2,#BMPCQE-4 ;CHECK IF THE POINTER IS AT THE LAST LOCATION.
2312 017526 001036 BNE 60$ ;EXIT IF NOT AT THE LAST LOCATION.
2313 017530 005762 000002 TST 2(R2) ;CHECK FOR A BMP CODE IN THE LAST CELL
2314 017534 001433 BEQ 60$ ;EXIT IF NO OVERFLOW HAS OCCURED, CELL EMPTY.
2315 017536 012301 MOV (R3)+,R1 ;GET THE TEST NUMBER OFF THE QUEUE.
2316 017540 011304 MOV (R3),R4 ;GET THE BMP CODE OFF THE QUEUE.
2317 017542 012705 015746 MOV #EM9303,R5 ;SELECT THE MESSAGE TO BE REPORTED.
2318 017546 PRINTX #EF9302 ;REPORT OVERFLOW CONDITION.
017546 012746 005704 MOV #EF9302,(SP)
017552 012746 000001 MOV #1,-(SP)
017556 010600 MOV SP,R0
017560 104415 TRAP C$PNTX

```

GLOBAL ERROR REPORTING ROUTINE

- ER9301 -

```

017562 062706 000004
2319 017566 004767 000002      JSR    PC,50$      ;REPORT THE LAST BMP CODE PLACED ON THE QUEUE.
2320 017572 000414              BR      60$        ;EXIT.
2321
2322 017574              50$: PRINTX #EF9301,R5,R1,R4 ;PRINT THE MESSAGE.
017574 010446              MOV    R4,-(SP)
017576 010146              MOV    R1,-(SP)
017600 010546              MOV    R5,-(SP)
017602 012746 005626      MOV    #EF9301,-(SP)
017606 012746 000004      MOV    #4,-(SP)
017612 010600              MOV    SP,R0
017614 104415              TRAP  C$PNTX
017616 062706 000012      ADD   #12,SP
2323 017622 000207      RTS    PC          ;RETURN.
2324 017624              60$: PASS         ;RESTORE THE GPR CONTENTS.
017624 004736              JSR    PC,@(SP)+  ;RETURN TO PREG05 SUBRT.
2325
2326 017626              ENDMSG
017626
017626 104423              L10016: TRAP  C$MSG

```

GLOBAL SUBROUTINES SECTION

```
2328 .SBTTL GLOBAL SUBROUTINES SECTION
2330 ;*****
2331 ;
2332 ;           FVTSKL3.P11
2333 ;
2334 ;*****
2336
2337
2338 ;++
2339 ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
2340 ; THAT ARE USED IN MORE THAN ONE TEST.
2341 ;--
```

GLOBAL SUBROUTINE

- ALTFLD -

```

2343 .SBTTL GLOBAL SUBROUTINE - ALTFLD -
2344 ;* *****
2345 ;* - Alter Device Register Fields Routine -
2346 ;* This subroutine alters the specified field of the specified device
2347 ;* register for the specified lines. This routine can be used to set
2348 ;* or clear bits within selected fields of selected registers.
2349 ;* Use examples: Set RX.BAUD.RATE fields on lines 3 and 6.
2350 ;* Clear TX.DMA bits on all lines.
2351 ;*
2352 ;* INPUTS: R1 - Address of the registers to alter.
2353 ;* R2 - Bit fields set to desired states.
2354 ;* R3 - Bit map of lines for which to alter register.
2355 ;* R4 - Mask of bits to alter (1 indicates change bit).
2356 ;* CSRA - Contains the address of the device CSR
2357 ;* IESTAT - Saved states of the interrupt enable bits.
2358 ;*
2359 ;* OUTPUTS: DEVICE REGISTERS - Specified register fields altered.
2360 ;* CSR IND.ADR.REG field - Destroyed.
2361 ;*
2362 ;* CALLING SEQUENCE: JSR PC,ALTFLD
2363 ;*
2364 ;* COMMENTS: This routine reads the specified registers for all lines
2365 ;* with numbers lower than the highest specified line.
2366 ;* This routine does not read the CSR.
2367 ;*
2368 ;* SUBROUTINES CALLED: None.
2369 ;*-- *****
2370
2371 017630 004567 164142 ALTFLD:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
; JSP R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2372
2373 ;*
2374 ;* Set up to loop for each line:
2375 ;* Prepare the word to be ORed into the register contents.
2376 ;* Set up the word to write into the IND.ADR.REG field of the CSR.
2377 ;*
2378 017634 010400 MOV R4,R0 ;CALCULATE THE NEW CONTENTS OF THE
2379 017636 005100 COM R0 ; REGISTER FIELDS WHICH ARE TO BE
2380 017640 040002 BIC R0,R2 ; ALTERED BY THIS ROUTINE.
2381 017642 016705 162462 MOV IESTAT,R5 ;SET UP TO WRITE IND.ADR.REG FIELD TO 0.
2382 ;*
2383 ;* Loop once for each line, altering the specified field in the specified
2384 ;* register if the line has been selected for altering.
2385 ;* Exit the loop if no more lines to alter, or if we have altered the max
2386 ;* allowable number of lines (as specified by NUMLNS).
2387 ;*--
2388 017646 000241 CLC ;PREPARE FOR ROTATE, "TST R5" DOES THIS BELOW.
2389 017650 006003 2$: ROR R3 ;GET THE LINE SELECT BIT FOR THIS LINE.
2390 017652 103006 BCC 4$ ;SKIP SETUP IF LINE IS NOT SELECTED.
2391 017654 010577 162364 MOV R5,@CSRA ;SET OUT CSR IND.ADR.REG FIELD TO THIS LINE.
2392 017660 011100 MOV (R1),R0 ;GET THE PRESENT CONTENTS OF THE REG TO ALTER.
2393 017662 040400 BIC R4,R0 ;CLEAR THE BIT FIELDS WE ARE TO ALTER.
2394 017664 050200 BIS R2,R0 ;OR IN THE NEW STATES OF THE FIELDS.
2395 017666 010011 MOV R0,(R1) ;WRITE THE NEW REGISTER CONTENTS TO THE REG.
2396 017670 005205 4$: INC R5 ;SET LINE NUMBER TO THE NEXT LINE.
2397 017672 005703 TST R3 ;CHECK FOR UNHANDLED LINES, CLEAR CARRY FLAG.
2398 017674 001365 BNE 2$ ;LOOP IF SELECTED LINE(S) IS NOT HANDLED.

```


GLOBAL SUBROUTINE

- ALTFLD -

```
2399
2400 017676          60$: PASS          ;RESTORE GPRS.
      017676 004736          PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2401 017700 000207          RTS   PC    JSR   ;RETURN TO CALLING ROUTNE.
```

GLOBAL SUBROUTINE

- CALMSL -

```

2403 .SBTTL GLOBAL SUBROUTINE - CALMSL -
2404 ;** *****
2405 ;* - Calibrate Milli Second Loop count subroutine -
2406 ;* This subroutine calibrates the timing loop which is used in the MSLOOP
2407 ;* routine. This subroutine calculates a value for the MSLCNT variable
2408 ;* which is the number of software loops which takes 1 ms to execute in
2409 ;* the MSLOOP routine. This routine calibrates the count by using the
2410 ;* Line Time Clock (LTC), so if no LTC is available the default value for
2411 ;* the delay count must be used.
2412 ;*
2413 ;*
2414 ;* INPUTS: MSLCNT - Default 1 ms delay loop count value, or
2415 ;* value from previous calibration.
2416 ;* MSTICK - Number of MS per LTC clock tick.
2417 ;* TIMER1 - Timer counter changed by LTC interrupt service rtn.
2418 ;* CLKHRZ - Number of LTC clicks per second (50 or 60).
2419 ;*
2420 ;* OUTPUTS: CARRY - Set if LTC is available, and new calibration performed.
2421 ;* MSLCNT - New 1 ms delay loop count value if LTC available, or
2422 ;* unchanged if no LTC is available.
2423 ;*
2424 ;* CALLING SEQUENCE: JSR PC,CALMSL
2425 ;*
2426 ;* COMMENTS:
2427 ;*
2428 ;* SUBORDINATE ROUTINES CALLED: UNSDIV,OOPS.
2429 ;-- *****
2430
2431 017702 CALMSL:: SAVE JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
017702 004567 164070 ; R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2432 017706 005067 000210 CLR 62$ ;CLEAR THE 2ND TIME FLAG.
2433 ;*
2434 ; Synchronize with the LTC.
2435 ;-
2436 017712 012705 000001 2$: MOV #1,R5 ;SET OUTER LOOP COUNTER TO 1 LOOP.
2437 ;INCREASE THE VALUE LOADED INTO THIS COUNTER IF THE <*&
2438 ;FOLLOWING LOOP FAILS ON FUTURE, FASTER PROCESSORS. <*&
2439 017716 005000 CLR R0 ;CLEAR THE WAIT FOR CLOCK INT COUNTER.
2440 017720 012767 000001 162436 MOV #1,TIMER1 ;SET UP COUNT OF 1 TO SYNCH WITH LTC.
2441 017726 005767 162432 4$: TST TIMER1 ;CHECK FOR COUNTER HAVING GONE TO ZERO.
2442 017732 001410 BEQ 6$ ;JUMP OUT OF LOOP IF LTC HAS INTERRUPTED.
2443 017734 005200 INC R0 ;COUNT THIS ITERATION OF THE INNER LOOP.
2444 017736 001373 BNE 4$ ;LOOP IF COUNTER HAS NOT TURNED OVER.
2445 017740 005305 DEC R5 ;DECREMENT THE INNER LOOP COUNTER.
2446 017742 003371 BGT 4$ ;LOOP IF OUTER LOOP COUNT NOT UP.
2447 ;*
2448 ; If we got no LTC interrupt, indicate that there is no LTC available.
2449 ; LTC must be flakey, or not really an LTC at all.
2450 ;-
2451 017744 005067 162412 CLR CLKHRZ ;CLEAR LTC FREQUENCY WORD TO INDICATE NO LTC.
2452 017750 000241 CLC ;INDICATE FAILURE FOR RETURN.
2453 017752 000461 BR 60$ ;BYPASS THE FOLLOWING CALIBRATION PROCEDURES.
2454 ;*
2455 ; We are now synchronized with the LTC.
2456 ; Set up for the calibration loop.
2457 ;-
2458 017754 012704 002364 6$: MOV #TIMER1,R4 ;WILL TEST TIMER1 IN THE LOOP BELOW.

```

GLOBAL SUBROUTINE

- CALMSL -

```

2459 017760 005001          CLR    R1          ;CLEAR THE OUTER LOOP COUNTER.
2460 017762 005002          CLR    R2          ;INDICATE TO CHECK ALL BITS OF TIMER1.
2461 017764 005003          CLR    R3          ;INDICATE TO CHECK FOR TIMER1 CLEAR.
2462 017766 012714 000001    MOV    #1,(R4)      ;LOAD TIMER1 WITH COUNT OF 1.
2463
2464 017772 016705 162400    8$:   MOV    MSLCNT,R5    ;LOAD MS LOOP COUNT.
2465 017776 011400    10$:  MOV    (R4),R0        ;GET THE TIMER1 VALUE.
2466 020000 010067 000120    MOV    R0,64$        ;SAVE WORD (LIKE IN THE REAL LOOP).
2467 020004 040200          BIC    R2,R0         ;LEAVE ALL THE BITS.
2468 020006 020003          CMP    R0,R3         ;COMPARE AGAINST ZERO.
2469 020010 000261          SEC          ;SET CARRY IN CASE OF SUCCESS.
2470 020012 001406          BEQ    12$          ;EXIT LOOP IF TIMER1 HAS CLEARED.
2471 020014 005305          DFC    R5           ;COUNT DOWN THE INSIDE MS LOOP COUNT.
2472 020016 001367          BNE    10$          ;LOOP IF MS NOT UP.
2473 020020 005301          DEC    R1           ;DECREMENT THE MS TIME COUNT.
2474 020022 001363          BNE    8$           ;KEEP LOOPING.
2475 020024 004767 000440    JSR    PC,OOPS      ;WE OVERFLOWED, SOMETHING IS WRONG, ABORT.
2476
2477          ;*
2478          ; We have now have loop count information for one clock tick.
2479          ; We have negative of number of outer loops in R1, each is MSLCNT inner loops.
2480          ; We have the portion of the last outer loop not executed, in R5.
2481          ; Now we calculate the total number of inner loops executed.
2482 020030 005401    12$:  NEG    R1           ;GET NUMBER OF OUTER LOOPS.
2483 020032 016702 162340    MOV    MSLCNT,R2    ;GET THE NUMBER OF INNER LOOPS PER OUTER LOOP.
2484 020036 010203          MOV    R2,R3        ;COPY NUMBER OF LOOPS FOR MULTIPLY.
2485 020040 160502          SUB    R5,R2        ;CALC # OF INNER LOOPS DONE IN LAST OUTER LOOP
2486 020042 010204          MOV    R2,R4        ; AND ADD TO ACCUMULATOR LSWORD.
2487 020044 005005          CLR    R5           ;CLEAR ACCUMULATOR MSWORD.
2488 020046 005301    14$:  DEC    R1           ;CHECK R1 FOR 0 CONDITION
2489 020050 100403          BMI    16$          ; SKIP MULTIPLICATION IF ZERO
2490 020052 060304          ADD    R3,R4        ;MULTIPLY NUMBER OF INNER
2491 020054 005505          ADC    R5           ; LOOPS PER OUTER LOOP BY
2492 020056 000773          BR    14$          ;NUMBER OF OUTER LOOPS PERFORMED.
2493
2494          ;*
2495          ; Divide the total number of inner loops by the number of MS per LTC tick.
2496 020060 016701 162310    16$:  MOV    MSTICK,R1    ;# OF MS PER LTC TICK IS DIVISOR.
2497 020064 010403          MOV    R4,R3        ;LSWORD OF LOOP COUNT IS LSWORD OF DIVIDEND.
2498 020066 010502          MOV    R5,R2        ;MSWORD OF LOOP COUNT IS MSWORD OF DIVIDEND.
2499 020070 004767 002756    JSR    PC,UNSDIV    ;DIVIDE NUMBER OF LOOPS BY MS PER LTC TICK.
2500 020074 103402          BCS    18$          ;BYPASS OOPS IF WE'RE OK.
2501 020076 004767 000366    JSR    PC,OOPS      ;CLOCK ROUTINES ARE NOT LONG ENOUGH, OR BUG.
2502 020102 010167 162270    18$:  MOV    R1,MSLCNT    ;SET NEW VALUE FOR MS LOOP COUNT.
2503 020106 005167 000010    COM    62$          ;SET THE 2ND ITERATION FLAGS IF 1ST ITERATION.
2504 020112 001277          BNE    2$           ;BRANCH IF ONLY ONE ITERATION DONE.
2505 020114 000261          SEC          ;SET THE SUCCESS FLAG FOR EXIT.
2506
2507 020116          60$:  PASS          ;RESTORE GPRS,
2508 020120 004736          RTS    PC          JSR    PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2509          ; CARRY - SUCCESS FLAG. SET IF SUCCESS.
2510 020122 000000    62$:  .WORD 0          ;2ND CALIBRATION ITERATION FLAGS.
2511 020124 000000    64$:  .WORD 0          ;DUMMY WORD FOR STORAGE OF THE READ WORD.

```

GLOBAL SUBROUTINE

CKTRAP -

```

2513 .SBTTL GLOBAL SUBROUTINE - CKTRAP -
2514 ;*****
2515 ;* Check Trap Routine -
2516 ;* This subroutine is used to check for a bus time-out trap (004 trap)
2517 ;* which is caused by an access to a non-existent memory or I/O location.
2518 ;* If the trap does not occur, this routine returns a success indication.
2519 ;*
2520 ;* INPUTS: R0 - Source address for move.
2521 ;* R1 - Destination address for move.
2522 ;* (R0) - Source for the move.
2523 ;*
2524 ;* OUTPUTS: (R1) - Written to the contents of (R0).
2525 ;* Carry flag - Set on return if no 004 trap detected.
2526 ;* TP4FLG - Nonzero if trap occurred, cleared otherwise.
2527 ;*
2528 ;* CALLING SEQUENCE: JSR PC,CKTRAP
2529 ;*
2530 ;* COMMENTS: If this subroutine causes a trap, either the address which
2531 ;* is labeled ADRPTR will be the trap PC address on the stack.
2532 ;*
2533 ;* SUBORDINATE ROUTINES CALLED: None.
2534 ;*****
2535
2536 020126 CKTRAP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020126 004567 163644 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2537 020132 005067 162210 CLR TP4FLG ;CLEAR THE 004 TRAP FLAGS.
2538 020136 011011 MOV (R0),(R1) ;PERFORM THE MOVE IN QUESTION.
2539 020140 005767 162202 ADRPTR:: TST TP4FLG ;CHECK FOR OCCURENCE OF TRAP.
2540 020144 000261 SEC ;INDICATE SUCCESS.
2541 020146 001401 BEQ 60$ ;EXIT WITH SUCCESS IF TRAP DID NOT OCCUR.
2542 020150 000241 CLC ;INDICATE FAILURE.
2543 020152 004736 60$: PASS ;RESTORE GPRS.
020152 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2544 020154 000207 RTS PC

```

GLOBAL SUBROUTINE

- CLNRST -

```

2546 .SBTTL GLOBAL SUBROUTINE CLNRST -
2547 ;*****
2548 ;* - Clean Reset of the Device Under Test -
2549 ;* This subroutine is used to reset the DUT to a known state.
2550 ;* The DUT's self-test is skipped, and the fifo is purged of any error
2551 ;* codes, etc.
2552 ;* If the reset does not successfully complete, then the carry bit is
2553 ;* passed back to the calling routine (clear).
2554 ;*
2555 ;* INPUTS: CSRA - Contains the address of the CSR
2556 ;* TXBFCA - Contains address of DUT DMA Buffer Count register.
2557 ;* ERRNBR - Error number for possible error report.
2558 ;* ERRTBL- ERRTYP,ERNBR,and ERRMSG set up correctly.
2559 ;*
2560 ;* OUTPUTS: The DUT performs its reset function into a known state.
2561 ;* CARRY - Clear indicates the test is to be aborted.
2562 ;* ERRBLK - value may be destroyed.
2563 ;* IESTAT - TX and RX interrupt flags are cleared.
2564 ;* TX and RX interrupt enable bits in the DUT's CSR are cleared.
2565 ;*
2566 ;* CALLING SEQUENCE: JSR PC,CLNRST
2567 ;*
2568 ;* COMMENTS: This subroutine can report errors with numbers ERRNBR.
2569 ;* This routine does not destroy the value of ERRNBR.
2570 ;*
2571 ;* SUBORDINATE ROUTINES CALLED: DELAY,MSLGET,PUFIFO,RESETT.
2572 ;*****
2573
2574 020156 CLNRST:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020156 004567 163614 JSP R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2575
2576 ;+
2577 ; Reset the DUT.
2578 ; This routine reports errors with numbers from ERRNBR thru ERRNBR+2.
2579 ;-
020162 004767 001270 JSR PC,RESETT ;RESET THE DUT TO A KNOWN STATE.
2580 020166 103002 BCC 60$ ;EXIT ROUTINE WITH ABORT TEST INDICATOR.
2581 ;+
2582 ; Purge the FIFO of error codes, save any BMP codes found.
2583 ;-
2584 020170 004767 000522 JSR PC,PUFIFO ;PURGE THE FIFO.
2585
2586 020174 60$: ;EXIT THE TEST USING RESETT OR PUFIFO STATUS.
2587 020174 PASS ;RESTORE GPRS, PASS THE FOLLOWING INTACT:
020174 004736 JSP PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2588 ;CARRY BIT:IF CLEAR, THEN ABORT THE TEST.
2589 020176 000207 RTS PC

```

GLOBAL SUBROUTINE

- CLR16W -

2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613

020200
020200 004567 163572
020204 012701 000020
020210 005020
020212 005301
020214 001375
020216
020216 004736
020220 000207

```

.SBTTL GLOBAL SUBROUTINE - CLR16W
;+ *****
;* - Clear Sixteen Words Routine -
;* This subroutine clears 16 words starting with the specified word.
;*
;* INPUTS: R0 - Address of the first word to clear.
;*
;* OUTPUTS: (R0) to (R0+15) - 16 words of memory are cleared to 0.
;*
;* CALLING SEQUENCE: JSR PC,CLR16W
;*
;* COMMENTS:
;*
;* SUBORDINATE ROUTINES CALLED: None.
;-- *****

CLR16W:: SAVE
                JSR    R5,PREG05 ;SAVE CONTENTS OF GPRS R0 THRU R5.
                MOV    #16.,R1 ;CALL REGISTER SAVE SUBRT.
                CLR    (R0)+ ;SET THE LOOP COUNTER TO 16.
2$:              ;CLEAR A WORD OF MEMORY.
                DEC    R1 ;COUNT THIS LOOP.
                BNE    2$ ;LOOP IF NOT 16 WORD CLEARED.
60$:            PASS ;RESTORE GPRS.
                JSR    PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
                RTS    PC

```

GLOBAL SUBROUTINE

- CNTERR -

```

2615 .SBTTL GLOBAL SUBROUTINE - CNTERR -
2616 ;** *****
2617 ;* - Count Error Routine -
2618 ;* This subroutine is used to count a "data" error on the specified
2619 ;* line. It checks whether error summary reporting is active, or should
2620 ;* be made active on this line, and activates it if necessary.
2621 ;*
2622 ;* INPUTS: R5 - Line number of line under consideration.
2623 ;* ERCNTB - Label at base of error counters table.
2624 ;* ERSMRF - Error summary flags (bit set if line in summary mode).
2625 ;* NDERPT - Number of individual data errors to report on a line.
2626 ;*
2627 ;* OUTPUTS: CARRY - Set if line is in error summary mode.
2628 ;* ERCNT - Error counter incremented for specified line.
2629 ;* ERSMRF Bit set if line should be in summary mode.
2630 ;*
2631 ;* CALLING SEQUENCE: JSR PC,CNTERR
2632 ;*
2633 ;* COMMENTS:
2634 ;*
2635 ;* SUBORDINATE ROUTINES CALLED: None.
2636 ;-- *****
2637
2638 020222 CNTERR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020222 004567 163550 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2639
2640 ; Count the error on the counter for the specified line.
2641 ;
2642 020226 006305 ASL R5 ;FORM WORD OFFSET FROM LINE NUMBER.
2643 020230 016501 002464 MOV ERCNTB(R5),R1 ;GET THE PRESENT ERROR COUNT FOR THIS LINE.
2644 020234 005201 INC R1 ;COUNT ERROR.
2645 020236 103402 BCS 2$ ;OVERFLOW? YES, DON'T UPDATE COUNTER IN TABLE.
2646 020240 010165 002464 MOV R1,ERCNTB(R5) ;UPDATE ERROR COUNTER TABLE ENTRY.
2647 020244 005767 161760 2$: TST NDERPT
2648 020250 001411 BEQ 60$ ;SUMMARYS DISABLED? YES, EXIT WITH CARRY 0.
2649 020252 020167 161752 CMP R1,NDERPT ;NO, CHECK FOR ENOUGH ERRORS FOR SUMMARY USE.
2650 020256 101002 BHI 4$ ;ENOUGH ERRORS TO USE SUMMARY? YES, GO HANDLE.
2651 020260 000241 CLC ;INDICATE NOT TO USE SUMMARY REPORT YET.
2652 020262 000404 BR 60$ ;EXIT WITH CARRY 0.
2653 020264 056567 002410 162170 4$: BIS BITTBL(R5),ERSMRF ;SET THE ERROR SUMMARY FLAG FOR LINE.
2654 020272 00C261 SEC ;INDICATE TO USE SUMMARY REPORT.
2655 020274 004736 60$: PASS ;RESTORE GPRS.
020274 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2656 020276 000207 RTS PC

```

GLOBAL SUBROUTINE

- DELAY -

```

2658 .SBTTL GLOBAL SUBROUTINE - DELAY -
2659 ;*****
2660 ;* - DELAY SUBROUTINE -
2661 ;* This subroutine is used to delay a variable number of milli-seconds.
2662 ;*
2663 ;* INPUTS: R4 - Contains the number of ms to delay.
2664 ;* MSLCNT.
2665 ;*
2666 ;* OUTPUTS: None.
2667 ;*
2668 ;* CALLING SEQUENCE: JSR PC,DELAY
2669 ;*
2670 ;* COMMENTS: If no hardware clock interrupts are occurring, control Cs will
2671 ;* not be honored for the duration of the delay.
2672 ;*
2673 ;* SUBORDINATE ROUTINES CALLED: None.
2674 ;*****
2675
2676 020300 DELAY:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020300 004567 163472 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2677 020304 010401 MOV R4,R1 ;PASS NUMBER OF MS DELAY AS TIME-OUT VALUE.
2678 020306 012702 177777 MOV #1,R2 ;TELL MSLOOP ROUTINE TO CHECK ALL BITS.
2679 020312 005003 CLR R3 ;TELL MSLOOP RTN TO CHECK FOR ALL BITS CLEAR.
2680 020314 012704 020336 MOV #62$,R4 ;TELL MSLOOP TO CHECK DUMMY NON-ZERO WORD.
2681 020320 004767 000130 JSR PC,MSLOOP ;DELAY THE REQUESTED # OF MS.
2682 020324 103002 BCC 60$ ;EXIT ROUTINE IF WE TIMED OUT.]
2683 020326 004767 000136 JSR PC,OOPS ;IF NO TIME-OUT, BAD PROGRAM OR HOST MACHINE.
2684 020332 60$: PASS ;RESTORE GPRS.
020332 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2685 020334 000207 RTS PC
2686
2687 020336 177777 62$: .WORD -1 ;DUMMY, NON ZERO WORD.

```


GLOBAL SUBROUTINE

- MSLGET -

```

2689 .SBTTL GLOBAL SUBROUTINE - MSLGET -
2690 ;*****
2691 ;* - Milli Seconds Loop which returns read word and remaining time -
2692 ;* This subroutine is a general purpose test loop subroutine. It is used
2693 ;* to verify that a certain action occurs before a time-out period. The
2694 ;* calling routine passes in which bits should be set and cleared for the
2695 ;* desired condition and the time-out value in milli-seconds.
2696 ;* This routine checks for the desired condition upon entrance into the
2697 ;* routine and then once each milli-second there after.
2698 ;* Upon return, the last word which was read to check for the condition
2699 ;* is returned by this subroutine.
2700 ;*
2701 ;* INPUTS: R1 - Time-out value in milli-seconds (up to 64K ms).
2702 ;* R2 - Bit map of bits to test (1 indicates to test the bit).
2703 ;* R3 - Desired states of the indicated fields in R2.
2704 ;* R4 - Address of the word to test.
2705 ;* MSLCNT - Milli second software loop count.
2706 ;*
2707 ;* OUTPUTS: R0 - The last word which was read to check for the condition.
2708 ;* R1 - Remaining number of ms in time-out time.
2709 ;* CARRY - Success flag (set if condition is met before time out).
2710 ;*
2711 ;* CALLING SEQUENCE: JSR PC,MSLGET
2712 ;*
2713 ;* COMMENTS: This routine works with or without a hardware clock, but the
2714 ;* calibration is only guaranteed when a line clock is available
2715 ;* on the system.
2716 ;* This routine can be used as a delay routine, by specifying the
2717 ;* desired delay as the time-out and specifying a condition to
2718 ;* look for which will not be met during the delay.
2719 ;* If a time-out value of 0 is specified, this routine checks for
2720 ;* the desired condition before returning. It indicates success
2721 ;* if the condition is met, failure otherwise.
2722 ;*
2723 ;*
2724 ;* SUBORDINATE ROUTINES CALLED: None.
2725 ;*****
2726
2727 020340 004567 163432 MSLGET:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
; JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2728 ;+
2729 ; Set up mask for removing unused bits in the test word, and clear unused
2730 ; bits in the desired state word to allow direct comparison.
2731 ;-
2732 020344 005102 COM R2 ;GET MASK OF UNUSED BITS.
2733 020346 040203 BIC R2,R3 ;MASK OUT UNUSED BITS IN DESIRED STATE WORD.
2734 ;+
2735 ; Handle the test and exit if we have a 0 time-out value.
2736 ;-
2737 020350 005701 TST R1 ;TEST THE TIME-OUT VALUE FOR ZERO.
2738 020352 001011 BNE 2$ ;IF NON-ZERO TIME-OUT, GO LOOP AND TEST.
2739 020354 011400 MOV (R4),R0 ;GET THE WORD TO TEST BEFORE EXITING.
2740 020356 010067 000070 MOV R0,62$ ;SAVE VALUE SO WE CAN RETURN IT.
2741 020362 040200 BIC R2,R0 ;MASK OUT UNTESTED BITS OF WORD.
2742 020364 020003 CMP R0,R3 ;COMPARE AGAINST DESIRED STATE WORD.
2743 020366 000261 SEC ;INDICATE SUCCESS IN CASE WORDS ARE EQUAL.
2744 020370 001420 BEQ 6$ ;EXIT WITH SUCCESS IF WORDS ARE EQUAL.

```

GLOBAL SUBROUTINE

- MSLGET -

```

2745 020372 000241          CLC          ;INDICATE FAILURE (TIME-OUT).
2746 020374 000416          BR          6$          ;EXIT WITH FAILURE, WORDS AREN'T EQUAL.
2747                          ;+
2748                          ; Non-zero time-out value. Loop, waiting for condition or time-out.
2749                          ;-
2750 020376 016705 161774  2$:      MOV      MSLCNT,R5      ;LOAD MS LOOP COUNT.
2751 020402 011400          4$:      MOV      (R4),R0      ;GET THE WORD TO TEST.
2752 020404 010067 000042  MOV      R0,62$      ;SAVE WORD IN CASE THIS IS THE LAST.
2753 020410 040200          BIC      R2,R0      ;MASK OUT UNTESTED BITS OF WORD.
2754 020412 020003          CMP      R0,R3      ;COMPARE AGAINST DESIRED STATE WORD.
2755 020414 000261          SEC          ;SET CARRY IN CASE OF SUCCESS.
2756 020416 001405          BEQ      6$          ;EXIT WITH SUCCESS IF WORDS ARE EQUAL.
2757 020420 005305          DEC      R5          ;COUNT DOWN THE INSIDE MS LOOP COUNT.
2758 020422 001367          BNE      4$          ;LOOP IF MS NOT UP.
2759 020424 005301          DEC      R1          ;DECREMENT THE MS TIME COUNT.
2760 020426 001363          BNE      2$          ;IF TIME NOT UP, LOOP TO COUNT ANOTHER MS.
2761 020430 000241          CLC          ;CLEAR CARRY, WE TIMED-OUT.
2762                          ;+
2763                          ; Have either found condition, or timed-out (possibly from 0 time-out value).
2764                          ; Restore the last contents read from the test word. Exit routine.
2765                          ;-
2766 020432 016700 000014  6$:      MOV      62$,R0      ;PASS OUT THE LAST READ WORD.
2767 020436 010066 000002  60$:     PASS      R0,R1      ;RESTORE GPRS, EXCEPT THE FOLLOWING:
                                MOV      R0,R0SLOT(SP)      ;PUT R0 IN STACK SLOT.
                                MOV      R1,R1SLOT(SP)      ;PUT R1 IN STACK SLOT.
                                JSR      PC,@(SP)+          ;RETURN TO PREG05 SUBRT.
2768                          ;R0 - LAST READ WORD CHECKED FOR CONDITION.
2769                          ;R1 - REMAINING TIME (0 IF TIME OUT OCCURED).
2770 020450 000207          RTS      PC          ;CARRY - SET IF SUCCESS, CLEAR IF TIME-OUT.
2771                          ;+
2772                          ; Local storage.
2773                          ;-
2774 020452 000000          62$:     .WORD 0          ;STORAGE FOR THE LAST READ WORD.

```

GLOBAL SUBROUTINE

- MSLOOP -

```

2776 .SBTTL GLOBAL SUBROUTINE - MSLOOP -
2777 ;*****
2778 ;* - Test Loop subroutine -
2779 ;* This subroutine is a general purpose test loop subroutine. It is used
2780 ;* to verify that a certain action occurs before a time-out period. The
2781 ;* calling routine passes in which bits should be set and cleared for the
2782 ;* desired condition and the time-out value in milli-seconds.
2783 ;* This routine checks for the desired condition upon entrance into the
2784 ;* routine and then once each milli-second thereafter.
2785 ;*
2786 ;* INPUTS: R1 - Time-out value in milli-seconds (up to 64K ms).
2787 ;* R2 - Bit map of bits to test (1 indicates to test the bit).
2788 ;* R3 - Desired states of the indicated fields in R2.
2789 ;* R4 - Address of the word to test.
2790 ;* MSLCNT - Milli second software loop count.
2791 ;*
2792 ;* OUTPUTS: CARRY - Success flag (set if condition is met before time-out).
2793 ;*
2794 ;* CALLING SEQUENCE: JSR PC,MSLOOP
2795 ;*
2796 ;* COMMENTS: This routine works with or without a hardware clock, but the
2797 ;* calibration is only guaranteed when a line clock is available
2798 ;* on the system.
2799 ;* This routine can be used as a delay routine, by specifying the
2800 ;* desired delay as the time-out and specifying a condition to
2801 ;* look for which will not be met during the delay.
2802 ;* If a time-out value of 0 is specified, this routine checks for
2803 ;* the desired condition before returning. It indicates success
2804 ;* if the condition is met, failure otherwise.
2805 ;*
2806 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
2807 ;*****
2808
2809 020454 MSLOOP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020454 004567 163316 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2810
2811 ;*
2812 ;* Calling the MSLGET routine from the MSLOOP routine isolates the caller of
2813 ;* MSLOOP from the returned test word and remaining time-out values.
2814 ;*
2815 020460 004767 177654 JSR PC,MSLGET ;CALL THE MULTI-PURPOSE MS LOOP AND SEARCH RTN.
2816
2817 020464 60$: PASS ;RESTORE GPRS,
020464 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2818 020466 000207 RTS PC ;CARRY SET IF SUCCESS, CLEAR IF TIME-OUT.

```

GLOBAL SUBROUTINE

- OOPS -

```

2820 .SBTTL GLOBAL SUBROUTINE - OOPS -
2821 ;* *****
2822 ;* - Program abort subroutine -
2823 ;* This subroutine is used to abort the program when a fatal error is
2824 ;* detected in the program or the host system hardware. An error message
2825 ;* is printed giving some information about the nature of the abort.
2826 ;*
2827 ;* INPUTS: R1 - Error code giving reason for abort.
2828 ;*
2829 ;* OUTPUTS: An error message is printed.
2830 ;* A list of return PC values for all subroutine calls is printed.
2831 ;*
2832 ;* CALLING SEQUENCE: JSR PC,OOPS
2833 ;*
2834 ;* COMMENTS:
2835 ;*
2836 ;* SUBORDINATE ROUTINES CALLED: None.
2837 ;*
2838 ;* *****
2839 OOPS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
2840 020470 004567 163302 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2841 ; REPORT "HOST COMPUTER HARDWARE OR SOFTWARE BUG ENCOUNTERED." ERROR.
2842 ERRSF 101,EM0101
2843 ; REPORT "PROGRAM HUNG, WAITING FOR A CONTROL-C."
2844 PRINTF #EM0102
2845 020504 012746 020620 MOV #EM0102,-(SP)
2846 020510 012746 000001 MOV #1,-(SP)
2847 020514 010600 MOV SP,R0
2848 020516 104417 TRAP C$PNTF
2849 020520 062706 000004 ADD #4,SP
2850 2$: BREAK ;LOOK FOR OPERATOR CONTROL-C INPUT.
2851 TRAP C$BRK
2852 60$: BR 2$ ;INFINITE LOOP.
2853 ;DON'T NEED THIS, BUT SOMEBODY MAY CHANGE THIS
2854 020530 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2855 020532 000207 RTS PC ; ROUTINE IN THE FUTURE, SO BE CONSISTANT.
2856
2857 EM0101:: .ASCIZ /HOST COMPUTER HARDWARE OR SOFTWARE BUG ENCOUNTERED./
2858
2859 020534 110 117 123
2860 020537 124 040 103
2861 020542 117 115 120
2862 020545 125 124 105
2863 020550 122 040 110
2864 020553 101 122 104
2865 020556 127 101 122
2866 020561 105 040 117
2867 020564 122 040 123
2868 020567 117 106 124
2869 020572 127 101 122
2870 020575 105 040 102
2871 020600 125 107 140
2872 020603 105 116 103
2873 020606 117 125 116
2874 020611 124 105 122

```

GLOBAL SUBROUTINE

- OOPS -

	020614	105	104	056	
	020617	000			
2850	020620	045	116	045	EM0102:: .ASCIZ /*N*APROGRAM HUNG, WAITING FOR A CONTROL-C. <*****N*/
	020623	101	120	122	
	020626	117	107	122	
	020631	101	115	040	
	020634	110	125	116	
	020637	107	054	040	
	020642	127	101	111	
	020645	124	111	116	
	020650	107	040	106	
	020653	117	122	040	
	020656	101	040	103	
	020661	117	116	124	
	020664	122	117	114	
	020667	055	103	056	
	020672	040	074	052	
	020675	052	052	052	
	020700	052	052	052	
	020703	052	052	052	
	020706	052	052	052	
	020711	045	116	045	
2851	020714	116	000		.EVEN

GLOBAL SUBROUTINE

PUFIFO -

```

2853 .SBTTL GLOBAL SUBROUTINE - PUFIFO -
2854 ;*****
2855 ;* - PURGE THE FIFO
2856 ;* This routine tries to remove all the characters from the FIFO.
2857 ;* Any BMP codes that are found are saved on the BMP code queue.
2858 ;*
2859 ;* INPUTS: RBUFA- Contains the address of the Receiver.
2860 ;*
2861 ;*
2862 ;* OUTPUTS: Carry bit - Indicates the state of the fifo, set:= purged.
2863 ;* BMPCQ - The contents of the BMP code queue may be updated.
2864 ;*
2865 ;* CALLING SEQUENCE: JSR PC,PUFIFO
2866 ;*
2867 ;* COMMENTS:
2868 ;*
2869 ;* SUBORDINATE ROUTINES CALLED: SAVBMP.
2870 ;*****
2871
2872 020716 PUFIFO::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020716 004567 163054 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2873 020722 012701 001000 MOV #512.,R1 ;SET MAXIMUM TRY COUNT OF 512.
2874 020726 016704 161314 MOV RBUFA,R4 ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
2875
2876 020732 011402 2$: MOV (R4),R2 ;GET THE CONTENTS OF THE RECEIVER BUFFER REG.
2877 020734 100016 BPL 6$ ;EXIT IF THE FIFO IS EMPTY, DATA_VALID CLR.
2878 ;+
2879 ; Check if the read character is actually a BMP code.
2880 ; If it is, then save it on the BMP code queue to be reported later.
2881 ;-
2882 020736 012700 070000 MOV #70000,R0 ;GENERATE A BIT MAP OF CHAR ERROR BITS
2883 020742 040200 BIC R2,R0 ; WHICH ARE NOT SET FOR CHAR.
2884 020744 001006 BNE 4$ ;THROW CHAR AWAY IF NOT BMP OR SELFTEST CODE.
2885 ;+
2886 ; Check if the read data is modem status , BMP or Selftest?.
2887 ;-
2888 020746 012700 000300 MOV #300,R0 ; CHECK IF BMP OR SELFTEST?.
2889 020752 040200 BIC R2,R0 ;TRY TO CLEAR BMP FLAGS IN THE READ DATA.
2890 020754 001002 BNE 4$ ;IF IT IS MODEM OR SELFTEST CODE THROW IT AWAY.
2891 020756 004767 001306 JSR PC,SAVBMP ;SAVE BMP CODE ON THE QUEUE.
2892
2893 020762 005301 4$: DEC R1 ;DECREMENT THE TRY COUNT.
2894 020764 001362 BNE 2$ ;LOOP TO TRY AGAIN.
2895 020766 000241 CLC ;CLEAR CARRY,TO INDICATE FIFO NOT PURGED.
2896 020770 000401 BR 60$ ;EXIT WITH CARRY CLEAR.
2897 020772 000261 6$: SEC ;SET CARRY, TO INDICATE FIFO PURGED.
2898
2899 020774 004736 60$: PASS ;RESTORE GPRS,
020774 JSR PC,@(SP). ;RETURN TO PREG05 SUBRT.
2900 ;CARRY BIT, SET INDICATES FIFO PURGED.
2901 020776 000207 RTS PC

```

GLOBAL SUBROUTINE

- RDPDR -

```

2903 .SBTTL GLOBAL SUBROUTINE - RDPDR -
2904 ;*****
2905 ;* - Read and verify Data Pattern from Device Registers Routine -
2906 ;* This routine reads and verifies the rotated data pattern which has
2907 ;* been written by the WDPDR subroutine.
2908 ;* Each active line's register's contents is read and compared with the
2909 ;* written data.
2910 ;* After the unused and Read Only (RO) bits are masked out, any errors are
2911 ;* reported from this routine.
2912 ;* This routine will take into account the type of write operation which
2913 ;* was performed by the WDPDR subroutine.
2914 ;*
2915 ;* INPUTS: R2 - Used to pass in the data pattern to be rotated & verified.
2916 ;* R3 - Byte indicator (- => lo byte, + => hi byte, 0 => both).
2917 ;* R4 - Operation type indicator (- => BIC, + => BIS, 0 => MOV).
2918 ;* ACTLNS - Bit map of active lines on the device under test.
2919 ;* CSRA - Contains the CSR address of the Device under test.
2920 ;* DRADRT - Base address of device register address table.
2921 ;* ERCNTB - Label at base of error counters table for lines.
2922 ;* ERRMSG - Set up with the proper error message for this test.
2923 ;* ERRNBR - Set up with the proper error number.
2924 ;* LPRO - Equated to LPR reg offset from device CSR address.
2925 ;* NUMLNS - Number of lines on the device under test.
2926 ;* NDERPT - Number of individual data errors to report on a line.
2927 ;* XBFCO - Equated to TBUFFCT reg offset from device CSR address.
2928 ;* NBTTB - Base address of the unused bit table.
2929 ;*
2930 ;* OUTPUTS: Error messages may be printed at the operator's console.
2931 ;* ERCNT - Error counters table is updated for line under test.
2932 ;* ERRBLK - Contents destroyed.
2933 ;* ERSRFR - Error summary flags bit set if line in summary mode.
2934 ;* UUT CSR - All bits cleared, except IND.ADR.REG field destroyed.
2935 ;*
2936 ;* CALLING SEQUENCE: JSR PC,RDPDR
2937 ;*
2938 ;* COMMENTS: For byte accesses, only the specified byte is verified.
2939 ;*
2940 ;* SUBORDINATE ROUTINES CALLED: ER1601,ROLDAP.
2941 ;-- *****
2942
2943 021000 RDPDR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
2944 021000 004567 162772 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2945 021004 012767 016712 162762 MOV #ER1601,ERRBLK ;SET UP THE ADDRESS OF THE ERROR REPORT RTN.
2946 ;+
2947 ; Determine whether register data should be inverted from data pattern.
2948 ;-
2948 021012 005704 TST R4 ;CHECK THE OPERAND TYPE INDICATOR.
2949 021014 100001 BPL 2$ ;BIC WRITE PERFORMED? NO, USE STANDARD DATA.
2950 021016 005102 COM R2 ;YES, INVERT THE DATA PATTERN.
2951 ;+
2952 ; Set up outer loop.
2953 ;-
2954 021020 005005 2$: CLR R5 ;CLEAR LINE COUNTER TO SELECT LINE 0.
2955 ;+
2956 ; The outer loop follows. Each pass through this loop reads and compares data
2957 ; from all of the device registers for a particular line if the line is active
2958 ;-

```

GLOBAL SUBROUTINE

- RDPDR -

```

2959 021022 010267 000170      4$:  MOV    R2,70$           ;SAVE THE OUTER LOOP DATA PATTERN.
2960 021026 010577 161212      MOV    R5,@CSRA        ;SET CSR IND.ADR.REG FIELD TO THIS LINE.
2961 021032 010500              MOV    R5,R0
2962 021034 006300              ASL    R0
2963 021036 036067 002410 161172  BIT    BITBL(R0),ACTLNS
2964 021044 001452              BEQ    16$             ;IS THE LINE ACTIVE? NO, SKIP THE LINE.
2965 021046 012703 000004      MOV    @LPRO,R3        ;YES, INITIALIZE REGISTER OFFSET FOR LPR.
2966
2967      ;*
2968      ; The inner loop follows. Each pass through this loop reads and compares
2969      ; data from a device register.
2970 021052 010204      6$:  MOV    R2,R4           ;SAVE THE INNER LOOP DATA PATTERN.
2971 021054 046302 002264      BIC    UNBITB(R3),R2   ;REMOVE UNUSED BITS FROM EXPECTED DATA.
2972 021060 016300 002244      MOV    DRADRT(R3),R0
2973 021064 005766 000010      TST    R3SLOT(SP)     ;CHECK THE ACCESS TYPE INDICATOR.
2974 021070 001002              BNE    8$             ;BYTE ACCESS? YES, GO PERFORM BYTE READ.
2975 021072 011001              MOV    (R0),R1        ;NO, PERFORM WORD READ OF DEVICE REGISTER.
2976 021074 000416              BR     12$
2977 021076 100410      8$:  BMI    10$           ;LOW BYTE ACCESS? YES, GO DO LOW BYTE READ.
2978 021100 005200              INC    R0              ;HIGH BYTE ACCESS. FORM HIGH BYTE ADDRESS.
2979 021102 111001              MOVB   (R0),R1        ;READ THE HI BYTE OF THE DUT REGISTER.
2980 021104 000301              SWAB   R1             ;PUT HI BYTE BACK INTO THE HI BYTE.
2981 021106 042701 000377      BIC    #377,R1        ;REMOVE THE UNUSED BYTE IN ACTUAL DATA.
2982 021112 042702 000377      BIC    #377,R2        ;REMOVE THE UNUSED BYTE IN EXPECTED DATA.
2983 021116 000405              BR     12$
2984 021120 111001      10$: MOVB   (R0),R1        ;READ THE LOW BYTE OF THE DUT REGISTER.
2985 021122 042701 177400      BIC    #177400,R1     ;REMOVE THE UNUSED BYTE.
2986 021126 042702 177400      BIC    #177400,R2     ;FORM EXPECTED LOW BYTE FOR COMPARISON.
2987
2988 021132 046301 002264      12$: BIC    UNBITB(R3),R1   ;REMOVE UNUSED BITS FROM ACTUAL DATA.
2989 021136 020102              CMP    R1,R2          ;COMPARE ACTUAL AND EXPECTED DATA.
2990 021140 001404              BEQ    14$           ;ACTUAL = EXPECTED? YES, SKIP ERROR.
2991 021142 004767 177054      JSR    PC,CNTERR      ;NO, COUNT THE ERROR, CHECK FOR ERROR SUMMARY.
2992 021146 103401              BCS    14$           ;USE ERROR SUMMARY? YES, SKIP ERROR.
2993
2994      ;No, report "BAD BIT(S) IN DEVICE xxxxx REGISTER FOR LINE nn (D)."  

2994 021150              ERROR
2994 021150 104460              TRAP   C$ERROR
2995 021152 010402      14$: MOV    R4,R2           ;RESTORE THE INNER LOOP DATA PATTERN.
2996 021154 004767 000410      JSR    PC,ROLDAP      ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
2997 021160 062703 000002      ADD    #2,R3          ;SET REGISTER OFFSET TO THE NEXT REGISTER.
2998 021164 020327 000016      CMP    R3,#TXBFCO    ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
2999 021170 003730      BLE    6$            ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
3000
3001      ;*
3002      ; Back into the outer loop. Now set up for next line. Loop if not done.
3003 021172 016702 000020      16$: MOV    70$,R2       ;SET UP TO ROTATE THE DATA PATTERN.
3004 021176 004767 000366      JSR    PC,ROLDAP      ;ROTATE THE DATA PATTERN.
3005 021202 005205              INC    R5             ;COUNT THIS LINE
3006 021204 020527 000010      CMP    R5,#NUMLNS    ;COMPARE LINE COUNT WITH NUMBER OF LINES.
3007 021210 002704              BLT    4$            ;LOOP IF SOME LINES NOT DONE.
3008
3009 021212      60$: PASS           ;RESTORE GPRS.
3009 021212 004736              JSR    PC,@(SP)+      ;RETURN TO PREG05 SUBRT.
3010 021214 000207      RTS    PC
3011
3012 021216 000000      70$: .WORD 0          ;STORAGE FOR DATA PATTERN OUTSIDE INNER LOOP.

```


GLOBAL SUBROUTINE

- REGTST -

```

3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040 021220
      021220 004567 162552
3041
3042
3043
3044 021224 012705 000020
3045 021230 012702 167410
3046 021234 032704 000001
3047 021240 001001
3048 021242 005004
3049 021244
3050
3051
3052
3053 021244 010400
3054 021246 004767 001142
3055 021252 016701 162512
3056 021256 010004
3057 021260 005404
3058 021262 005002
3059 021264 026627 000012 000002
3060 021272 001401
3061 021274 005102
3062 021276 005003
3063 021300 005000
3064 021302 026627 000012 177776
3065 021310 001001
3066 021312 005100
3067 021314 004767 001074
3068
3069

```

```

.SBTTL GLOBAL SUBROUTINE - REGTST -
;* *****
;* - Registers Test Subroutine
;* Subroutine to test the Device Under Test (DUT) registers. The used
;* bits of the registers are either all cleared or all set and then the
;* data pattern is written and verified using either word or byte
;* accesses in read/write or read/modify/write mode.
;*
;* INPUTS: R3 Byte indicator (- => low, + => high, 0 => both bytes).
;* R4 - Access mode (-1 => set then BIC, 1 => clear then BIS,
;* (-2 => set then MOV, +2 clear then MOV).
;* ERRNBR - Set up with initial error number.
;*
;* OUTPUTS: GPRS0 - GPR save area 0 is destroyed.
;* Device Under Test registers are written.
;* Error messages may be printed at the operators console.
;*
;* CALLING SEQUENCE: JSR PC,REGTST
;*
;* COMMENTS: This routine loop 16 times writing the same data pattern
;* rotated left once each iteration.
;* This routine can report errors INITIAL ERRNBR thru INITIAL+2.
;*
;* SUBORDINATE ROUTINES CALLED: RDPDR,ROLDAP,SWAPO,WDPDR
;* *****
REGTST:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
          JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
;*
;* Set up the GPRs for the writing of the data pattern.
;*
;*
          MOV #16.,R5 ;SET UP LOOP COUNTER TO COUNT 16 ITERATIONS.
          MOV #167410,R2 ;INITIALIZE THE DATA PATTERN.
          BIT #BIT0,R4 ;TEST FOR R/W ACCESS.
          BNE 2$ ;R/M/W ACCESS? YES, R4 IS ALL SET UP.
          CLR R4 ;NO, INDICATE R/W ACCESS.
2$:
;*
;* Set up the GPRs for the clearing or setting of all the used bits.
;*
;*
          MOV R4,R0 ;PASS OPERATION TYPE INDICATOR AROUND SWAPO.
          JSR PC,SWAPO ;GET ALTERNATE GPR SET IN R1 THRU R5.
          MOV ERRNBR,R1 ;SAVE THE INITIAL ERROR NUMBER.
          MOV R0,R4
          NEG R4 ;SET UP OP TYPE FOR CLEARING OR SETTING.
          CLR R2 ;SET UP CLEAR WRITE PATTERN.
          CMP R4,SLOT(SP),#2 ;TEST FOR CLEAR THEN MOV TEST SEQUENCE.
          BEQ 4$ ;CLEAR THEN MOV? YES, LEAVE WRITE PAT CLEAR.
          COM R2 ;NO, SET ALL BITS OF WRITE PATTERN.
          CLR R3 ;INDICATE THAT WORD ACCESSES SHOULD BE USED.
          CLR R0 ;SET ALTERNATE BYTE EXPECTED DATA PAT TO CLEAR.
          CMP R4,SLOT(SP),#-2 ;TEST FOR SET THEN MOV TEST SEQUENCE.
          BNE 6$ ;SET THEN MOV? YES, LEAVE ALT BYTE PAT CLEAR.
          COM R0 ;NO, SET ALT BYTE EXPECTED DATA PAT TO ALL 1 S.
          JSR PC,SWAPO ;RESTORE SWAPPED GPR VALUES TO R1 THRU R5.
6$:
;*
;* Start of data pattern loop.

```

GLOBAL SUBROUTINE

- REGTST -

```

3070
3071 021320      8$:
3072
3073      ;+
3074      ; Set or clear all the used bits of the device registers for all lines.
3075      ; Verify that all the bits were set or cleared correctly.
3076 021320 004767 001070      JSR    PC,SWAPO      ;GET ALTERNATE GPRS FOR SETTING INTIAL STATES.
3077 021324 004767 002006      JSR    PC,WDPDR      ;GO CLEAR ALL USED REGISTER BITS, ALL LINES.
3078 021330 010167 162434      MOV    R1,ERRNBR     ;SET UP ERROR NUMBER TO INITIAL ERRNBR.
3079 021334 004767 177440      JSR    PC,RDPDR      ;VERIFY ALL USED REGISTER BITS, ALL LINES.
3080 021340 004767 001050      JSR    PC,SWAPO      ;RESTORE MAIN GPRS CONTENTS.
3081
3082      ;+
3083      ; Write data patterns, all lower byte used bits, all registers, all lines.
3084      ; Verify that the data pattern was written correctly.
3085 021344 004767 001766      JSR    PC,WDPDR      ;WRITE DATA PATTERN TO DEVICE REGISTERS.
3086 021350 C05267 162414      INC    ERRNBR        ;SET ERROR NUMBER TO INITIAL+1.
3087 021354 004767 177420      JSR    PC,RDPDR      ;VERIFY DATA PATTERN IN ALTERRED BYTE(S).
3088 021360 005703
3089 021362 001411      TST    R3            ;CHECK THE BYTE INDICATOR.
3090      BEQ    10$      ;WORD ACCESS? YES, SKIP SECOND BYTE CHECK.
3091      ;+
3092      ; Check that the alternate (unmodified) byte is clear or set as expected.
3093 021364 010201      MOV    R2,R1        ;SAVE THE DATA PATTERN.
3094 021366 010002      MOV    R0,R2        ;GET THE ALTERNATE BYTE EXPECTED DATA.
3095 021370 005403      NEG    R3            ;INDICATE THAT OTHER BYTE IS TO BE CHECKED.
3096 021372 005267 162372      INC    ERRNBR        ;SET ERROR NUMBER TO INITIAL+2.
3097 021376 004767 177376      JSR    PC,RDPDR      ;VERIFY DATA PATS IN OTHER BYTES OF REGISTERS.
3098 021402 005403      NEG    R3            ;RESTORE BYTE INDICATOR.
3099 021404 010102      MOV    R1,R2        ;RESTORE DATA PATTERN.
3100
3101      ;+
3102      ; Pepare the next data pattern and loop if not done.
3103 021406 004767 000156      10$: JSR    PC,ROLDAP     ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
3104 021412 005305      DEC    R5            ;COUNT THIS ITERATION OF THE LOOP.
3105 021414 003341      BGT    8$            ;ALL PATTERNS DONE? NO, LOOP.
3106      ;YES, RESTORE ERROR NUMBER AND EXIT.
3107 021416 016767 161026 162344 60$: MOV    GPRSOB,ERRNBR ;GET THE ERROR NUMBR FROM GPR SWAP STORAGE.
3108 021424      PASS      ;RESTORE GPRS.
3109 021426 000207      RTS    PC          JSR    PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- REPSMR -

```

3111 .SBTTL GLOBAL SUBROUTINE - REPSMR -
3112 ;* *****
3113 ;* - Report Error Summary Routine -
3114 ;* This subroutine reports an error summary for those lines which have
3115 ;* exceeded the number of individual errors to report for a single line
3116 ;* in a single test. This parameter can be specified by the operator if
3117 ;* he/she answers the Software Parameter Questions.
3118 ;*
3119 ;* INPUTS: ERCNTB - Label at base of line error counters table.
3120 ;* ERRMSG - Address of primary error message.
3121 ;* ERRNBR - Error number of errors in this routine.
3122 ;* ERSMRF - "Report error summary for line" flags.
3123 ;*
3124 ;* OUTPUTS: ERRBLK - Address of error reporting routine (Destroyed).
3125 ;* Summary messages may be printed at the operator console.
3126 ;*
3127 ;* CALLING SEQUENCE: JSR PC,REPSMR
3128 ;*
3129 ;* COMMENTS: If no lines have exceeded the maximum number of individual
3130 ;* errors to report, no messages are printed by this routine.
3131 ;* Error summaries in this routine are reported as errors.
3132 ;* The contents of ERRBLK are destroyed.
3133 ;*
3134 ;* SUBORDINATE ROUTINES CALLED:
3135 ;-- *****
3136
3137 021430 REPSMR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3138 021430 004567 162342 TST ERSMRF JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3139 021440 001404 BEQ 60$ ;CHECK THE "PRINT LINE ERROR SUMMARY" FLAGS.
3140 ;* ;EXIT WITHOUT ACTION IF NO SUMMARY FLAGS SET.
3141 ;*
3142 ;* We have some error summaries to report.
3143 021442 012767 017160 162324 MOV #ER9004,ERRBLK ;SELECT ERROR REPORTING ROUTINE.
3144 ;*
3145 ;* Report
3146 ;* "ERROR SUMMARY REPORT FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:"
3147 ;*
3148 021450 ERROR TRAP C$ERROR
3149 021450 104460
3150 021452 60$: PASS ;RESTORE GPRS.
3151 021452 004736 RTS PC JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3151 021454 000207

```

GLOBAL SUBROUTINE

- RESETT -

```

3153 .SBTTL GLOBAL SUBROUTINE - RESETT -
3154 ;*****
3155 ;* - Reset Device Under Test -
3156 ;* This subroutine is used to reset the DUT to a known state.
3157 ;* If reset does not successfully complete, ie. time-out occurs, then
3158 ;* an abort test error message is reported.
3159 ;*
3160 ;* INPUTS: CSRA - Contains the address of the CSR
3161 ;* TXBFCA - Contains address of DUT DMA Buffer Count register.
3162 ;* ERRTBL- ERRTYP,ERNBR,and ERRMSG set up correctly.
3163 ;*
3164 ;* OUTPUTS: The DUT performs its reset function into a known state.
3165 ;* CARRY Clear indicates the test is to be aborted.
3166 ;* ERRBLK - value may be destroyed.
3167 ;* IESTAT - TX and RX interrupt flags are cleared.
3168 ;* TX and RX interrupt enable bits in the DUT's CSR are cleared.
3169 ;*
3170 ;* CALLING SEQUENCE: JSR PC,RESETT
3171 ;*
3172 ;* COMMENTS: This subroutine can report errors with numbers initial ERRNBR
3173 ;* This routine does not destroy the value of ERRNBR.
3174 ;*
3175 ;* SUBORDINATE ROUTINES CALLED: DELAY,MSLGET.
3176 ;*****
3177
3178 021456 RESETT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
021456 004567 162314 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3179 021462 012702 000040 MOV #BIT05,R2 ;SET BIT MASK OF MASTER RESET BIT.
3180
3181 ;+
3182 ; Test the state of the master reset bit in the CSR.
3183 ; If MR is set then wait for self-test to complete.
3184 ; If time-out occurs, report the error and pass out abort test indicator.
3185 ;-
021466 016704 160552 MOV CSRA,R4 ;GET THE ADDRESS OF THE DUT'S CSR.
021472 030214 BIT R2,(R4) ;CHECK STATE OF MASTER RESET BIT.
3187 021474 001406 BEQ 2$ ;DON'T DELAY IF MR IS ALREADY CLEAR.
3188 021476 005003 CLR R3 ;SET UP DESIRED STATE OF MASTER RESET BIT.
3189 021500 012701 004704 MOV #2500.,R1 ;PASS TIME OUT VALUE OF 2.5 SECONDS.
3190 021504 004767 176630 JSR PC,MSLGET ;WAIT FOR SELF-TEST TO COMPLETE, MR CLEAR.
3191 021510 103012 BCC 4$ ;GO REPORT ERROR IF TIMEOUT OCCURRED.
3192
3193 ;+
3194 ; Set Master Reset bit in CSR. Clear TX and RX enable bits, etc.
3195 ; Skip the selftest.
3196 ; Time-out of 2.5 secs, just in case the self-test executes.
3197 ;-
021512 010277 160526 2$: MOV R2,@CSRA ;SET MASTER RESET BIT, DISABLE TX AND RX INTS.
021516 004767 000614 JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST.
3200
3201 ;+
3202 ; Set Self-test time-out of 2.5 seconds, and wait for M.R to clear.
3203 ; If Time-out occurs, then report the fatal error and pass out the abort
3204 ; test indicator.
3205 ;-
021522 005003 CLR R3 ;SET UP DESIRED STATE OF MASTER RESET BIT.
021524 012701 004704 MOV #2500.,R1 ;PASS TIME OUT VALUE OF 2.5 SECONDS.
021530 004767 176604 JSR PC,MSLGET ;WAIT FOR SELF-TEST TO COMPLETE, MR CLEAR.
021534 103410 BCS 6$ ;SKIP ERROR REPORT IF MR CLEARED IN TIME.

```

GLOBAL SUBROUTINE

- RESETT -

```

3209
3210 ; Set up error message to report "fatal error found during reset, test aborted".
3211 ; Indicate test is to be aborted by clearing the carry bit.
3212
3213 021536 012701 012362 4$: MOV #EM1601,R1 ;PASS ERROR MESSAGE TO REPORT.
3214 021542 012767 017010 162224 MOV #ER1603,ERRBLK ;PASS ADDRESS OF ERROR HANDLING ROUTINE.
3215 ;REPORT ERROR "TIME OUT OCCURRED WAITING FOR MASTER RESET TO CLEAR"
3216 ; "TEST ABORTED"
3217 021550 ERROR ; >>>>> ERROR <<<<<<
      021550 104460 TRAP C$ERROR
3218 021552 000241 CLC ;INDICATE TEST IS TO BE ABORTED.
3219 021554 000403 BR 60$ ;EXIT THIS SUBROUTINE, ABORT TEST INDICATOR.
3220
3221 ; Clear TX and RX Interrupt enable status flags in IESTAT.
3222 ; Exit with continue test indicator set (ie, carry set).
3223
3224 021556 005067 160546 6$: CLR IESTAT ;CLEAR TX AND RX INTERRUPT STATUS FLAGS.
3225 021562 000261 SEC ;INDICATE SUCCESS, CONTINUE TEST.
3226
3227 021564 004736 60$: PASS ;RESTORE GPRS, PASS THE FOLLOWING INTACT:
      021564 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3228 ;CARRY BIT:IF CLEAR,INDICATES ABORT TEST.
3229 021566 000207 RTS PC
3230

```

GLOBAL SUBROUTINE

- ROLDAP -

```

3232 .SBTTL GLOBAL SUBROUTINE ROLDAP
3233 ;*****
3234 ;* - ROTate Left Data Pattern
3235 ;* This routine rotates the passed input data pattern left,without going
3236 ;* through the carry.The carry is initially set or cleared depending
3237 ;* upon the state of the MSB of the data pattern,before a ROL instruction
3238 ;* is executed.
3239 ;*
3240 ;* INPUTS: R2 Contains the data pattern to be rotated
3241 ;*
3242 ;* OUTPUTS: R2 Contains the rotated data pattern
3243 ;*
3244 ;* CALLING SEQUENCE: JSR PC,ROLDAP
3245 ;*
3246 ;* COMMENTS:
3247 ;*
3248 ;* SUBORDINATE ROUTINES CALLED: NONE
3249 ;*****
3250
3251 021570 ROLDAP::SAVE JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
021570 004567 162202 R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3252 021574 010202 MOV R2,R2 ;SET PROCESSOR STATUS CODES
3253 021576 005702 TST R2 ;CHECK MSB
3254 021600 100402 BMI 2$ ;BRANCH IF SET
3255 021602 000241 CLC ;CLEAR CARRY BIT IF MSB CLEAR
3256 021604 000401 BR 4$ ;
3257 021606 000261 2$: SEC ;SET CARRY IF MSB SET
3258 021610 006102 4$: ROL R2 ;ROTATE DATA PATTERN LEFT
3259 021612 60$: PASS R2 ;RESTORE GPRS,EXCEPT
021612 010266 000006 MOV R2,R2SLOT(SP) ;PUT R2 IN STACK SLOT.
021616 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
;R2 - CONTAINS THE ROTATED DATA PATTERN
3260
3261 021620 000207 RTS PC

```

GLOBAL SUBROUTINE

- RSTRPT -

```

3263 .SBTTL GLOBAL SUBROUTINE - RSTRPT
3264 ;+ *****
3265 ;* - Report any Reset Errors Routine -
3266 ;* This routine determines if any error codes are among the diagnostic
3267 ;* codes reported placed in the DUT received character FIFO by the
3268 ;* Self-test. If any non BMP error codes are found, or if other errors
3269 ;* are encountered, appropriate errors are reported. Any BMP codes that
3270 ;* are found, are placed on the BMP code queue to be reported later.
3271 ;* This routine also purges the DUT FIFO looking for any characters
3272 ;* or modem status codes. If any are found, errors are reported.
3273 ;*
3274 ;* INPUTS: ERRMSG - Address of the primary error message.
3275 ;*          ERRNBR - Error number of first error reported by this routine.
3276 ;*          NUMLNS - Equated to the number of line on the DUT.
3277 ;*          RBUFA - Contains address of the DUT receiver FIFO.
3278 ;*
3279 ;* OUTPUTS: CARRY - Success flag (set if FIFO cleared successfully).
3280 ;*          ERRBLK - Address of the error report routine (Destroyed).
3281 ;*          Error messages can be printed at the operators console.
3282 ;*
3283 ;* CALLING SEQUENCE: JSR PC,RSTRPT
3284 ;*
3285 ;* COMMENTS: This subroutine can report errors with numbers Initial ERRNBR
3286 ;*          thru Initial ERRNBR+4.
3287 ;*          This routine does not destroy the value of ERRNBR.
3288 ;*
3289 ;* SUBORDINATE ROUTINES CALLED: ER0503,ER9007,ER9008,SAVBMP.
3290 ;-- *****
3291
3292 021622 004567 162150 RSTRPT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3293 ;+ ; R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3294 ; Read correct number (number of line on DUT) of chars from the FIFO.
3295 ; Verify that each char is a selftest success code.
3296 ;-
3297 021626 005003 ; CLEAR THE CODE COUNTER.
3298 021630 016705 162134 ; SAVE ERRNBR FOR RESTORATION LATER.
3299 021634 017702 160406 2$: MOV @RBUFA,R2 ;READ A CHAR FROM THE DUT FIFO.
3300 021640 100412 BMI 4$ ;SKIP ERROR IF DATA.VALID SET FOR CHAR.
3301 ;+
3302 ; We expect a selftest code, but this FIFO slot is empty.
3303 ;-
3304 021642 010567 162122 ; RESTORE ERROR NUMBER TO INITIAL VALUE.
3305 021646 012701 015522 ; PASS ERROR MESSAGE INFO TO ER9007 ROUTINE.
3306 021652 012767 017262 162114 ; SELECT PROPER ERROR REPORT ROUTINE.
3307 ;+
3308 ; Report error with number Initial ERRNRB.
3309 ; "NO SELFTEST CODE IN SELFTEST CODE FIFO SLOT FOR LINE nn AFTER RESET."
3310 ;
3311 021660 104460 ERROR ; >>>> ERROR <<<<<.
3312 ;+ ; TRAP C$ERROR
3313 ; Indicate "success" (because FIFO is purged), and exit this routine.
3314 ;-
3315 021662 000261 SEC ;SET SUCCESS FLAG.
3316 021664 000545 BR 60$ ;EXIT ROUTINE.
3317 ;+

```

GLOBAL SUBROUTINE

RSTRPT

```

3318 ; Determine if this is not a selftest code.
3319 ;
3320 021666 012700 070001 4$: MOV #70001,R0 ;GENERATE BIT MAP OF ANY CLEAR ERROR BITS OR
3321 021672 040200 BIC R2,R0 ; BIT 0 WHICH ARE CLEAR.
3322 021674 001033 BNE 8$ ;GO TO REPORT ERROR IF THIS IS NOT A TEST CODE.
3323 ;
3324 ; We have a test code (either BMP or selftest code).
3325 ; Determine what type of code we have.
3326 ;
3327 021676 032702 000200 BIT #BIT7,R2 ;TEST ROM VERSION CODE INDICATOR BIT.
3328 021702 001443 BEQ 10$ ;SKIP ERRORS IF SELFTEST ROM VERSION CODE.
3329 021704 120227 000203 CMPB R2,#203 ;CHECK IF SKIP SELF TEST CODE.
3330 021710 001440 BEQ 10$ ;SKIP ERROR REPORT IF SKIP SELF TEST CODE FOUND
3331 021712 120227 000201 CMPB R2,#201 ;CHECK IF NULL CODE PRESENT.
3332 021716 001435 BEQ 10$ ;SKIP ERROR REPORT IF SELF TEST NULL CODE.
3333 021720 012700 000300 MOV #300,R0 ;TEST CODE TYPE BITS FOR BOTH CODE
3334 021724 040200 BIC R2,R0 ; TYPE BITS SET (INDICATING BMP CODE).
3335 021726 001003 BNE 6$ ;IF IT IS NOT A BMP CODE GO REPORT ERROR.
3336 021730 004767 000334 JSR PC,SAVBMP ;SAVE THE BMP CODE ON THE QUEUE.
3337 021734 000426 BR 10$ ;GO GET THE NEXT CHARACTER FROM THE FIFO.
3338 ;
3339 ; We have a selftest error code.
3340 ;
3341 021736 010567 162026 6$: MOV R5,ERRNBR ;RESTORE ERROR NUMBER TO INITIAL VALUE.
3342 021742 005267 162022 INC ERRNBR ;CALCULATE INITIAL ERROR NUMBER PLUS 1.
3343 021746 012701 015547 MOV #EM9020,R1 ;PASS ERROR MESSAGE INFO TO ER9008 ROUTINE.
3344 021752 012767 017340 162014 MOV #ER9008,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3345 ;
3346 ; Report error with number Initial ERRNRB + 1.
3347 ; "UNEXPECTED SELFTEST ERROR CODE FOR LINE nn IN FIFO AFTER RESET:"
3348 ;
3349 021760 ERROR ; >>>> ERROR <<<<<.
021760 104460 TRAP C$ERROR
3350 021762 000413 BR 10$ ;GO TO END OF LOOP.
3351 ;
3352 ; We have a non selftest code (either BMP code or data char).
3353 ;
3354 021764 010567 162000 8$: MOV R5,ERRNBR ;RESTORE ERROR NUMBER TO INITIAL VALUE.
3355 021770 062767 000002 161772 ADD #2,ERRNBR ;CALCULATE INITIAL ERROR NUMBER PLUS 2.
3356 021776 012701 015532 MOV #EM9019,R1 ;PASS ERROR MESSAGE INFO TO ER9007 ROUTINE.
3357 022002 012767 017262 161764 MOV #ER9007,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3358 ;
3359 ; Report error with number Initial ERRNRB + 2.
3360 ; "NON-SELFTEST CODE IN SELFTEST CODE FIFO SLOT FOR LINE nn AFTER RESET."
3361 ;
3362 022010 ERROR ; >>>> ERROR <<<<<.
022010 104460 TRAP C$ERROR
3363 ;
3364 ; End of loop, loop if not all chars have been read from the FIFO.
3365 ;
3366 022012 005203 10$: INC R3 ;SET CODE COUNTER FOR NEXT ITERATION OF LOOP.
3367 022014 020327 000010 CMP R3,#NUMLNS ;TEST FOR ALL CODES READ.
3368 022020 002705 BLT 2$ ;LOOP IF NOT CHARS READ FROM FIFO.
3369 ;
3370 ; Purge the FIFO until DATA.VALID is clear or until too many chars are read.
3371 ;
3372 022022 012704 000022 MOV #18.,R4 ;INITIALIZE THE CHARACTER COUNTER.

```


GLOBAL SUBROUTINE

- RSTRPT -

```

3373 022026 010567 161736      MOV    R5,ERRNBR      ;GET INITIAL VALUE OF THE ERROR NUMBER.
3374 022032 062767 000003 161730  ADD    #3,ERRNBR      ;CALCULATE ERROR NUMBER OF NEXT ERROR.
3375 022040 012767 017340 161726  MOV    #ER9008,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3376 022046 017702 160174      MOV    @RBUFA,R2      ;READ A CHARACTER FROM THE DUT FIFO.
3377 022052 000261      SEC                      ;INDICATE SUCCESS IN CASE DATA.VALID IS CLEAR.
3378 022054 100051      BPL    60$            ;EXIT ROUTINE WITH SUCCESS IF DATA.VALID CLEAR.
3379
3380      ;+
3381      ; We have a character.
3382      ; Determine if character is a data character.
3383 022056 012700 070000      MOV    #70000,R0      ;TEST BITS 12 THRU 14 OF THE
3384 022062 040200      BIC    R2,R0          ; CODE READ FROM THE DUT FIFO.
3385 022064 001403      BEQ    14$            ;SKIP THIS ERROR IF CODE IS NOT A DATA CHAR.
3386
3387      ;+
3388      ; We have an unexpected data character: set up and go to report error.
3389 022066 012701 015573      MOV    #EM9022,R1     ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
3390 022072 000423      BR     22$            ;GO TO REPORT THIS ERROR.
3391
3392      ;+
3393      ; We have an unexpected code.
3394      ; Determine if the code is a modem status code.
3395 022074 032702 000001 14$:  BIT    #BIT0,R2      ;TEST MODEM STATUS INDICATOR BIT OF CODE.
3396 022100 001003      BNE    16$            ;SKIP THIS ERROR IF NOT MODEM STATUS CODE.
3397
3398      ;+
3399      ; We have a modem status code: set up and go to report error.
3400 022102 012701 015612      MOV    #EM9023,R1     ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
3401 022106 000415      BR     22$            ;GO TO REPORT THIS ERROR.
3402
3403      ;+
3404      ; We have an onboard test code.
3405      ; Determine if this code is a BMP code.
3406 022110 032702 000200 16$:  BIT    #BIT7,R2      ;TEST THE ROM VERSION BIT OF THE CODE.
3407 022114 001404      BEQ    18$            ;GOTO SET UP FOR SELFTEST CODE IF ROM VERSION.
3408 022116 012700 000300      MOV    #300,R0        ;
3409 022122 040200      BIC    R2,R0          ;TEST THE ERROR TYPE BITS OF THE CODE.
3410 022124 001403      BEQ    20$            ;SKIP THIS ERROR IF BMP CODE.
3411
3412      ;+
3413      ; We have a selftest code: set up and go to report error.
3414 022126 012701 015634 18$:  MOV    #EM9024,R1     ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
3415 022132 000403      BR     22$            ;GO TO REPORT THIS ERROR.
3416
3417      ;+
3418      ; We have a BMP code: save it on the queue.
3419 022134 004767 000130 20$:  JSR    PC,SAVBMP      ;SAVE THE BMP CODE ON THE QUEUE.
3420 022140 000401      BR     24$            ;
3421
3422      ;+
3423      ; Report the error with error number of Initial ERRNBR + 3.
3424      ; "UNEXPECTED xxx xxxx FOR LINE nn IN FIFO AFTER RESET:"
3425 022142 104460 22$:  ERROR      ;          >>>> ERROR <<<<<<
3426      ;+
3427      ; End of loop.
3428      ; Count the character we just received, and check for too many received.

```

GLOBAL SUBROUTINE

- RSTRPT

```

3429
3430 022144 005304      24$:  DEC   R4           ;COUNT THIS CHARACTER.
3431 022146 001337      BNE   12$           ;LOOP IF NOT TOO MANY CHARACTERS PURGED.
3432
3433      ;+
3434      ; We read too many valid characters while trying to purge the FIFO.
3435      ; Report error and exit without success.
3436      ; "FIFO WILL NOT PURGE (DATA.VALID STUCK SET), REMAINDER OF TEST SKIPPED."
3437 022150 012701 015411      MOV   #EM9017,R1      ;SELECT PROPER ERROR MESSAGE.
3438 022154 010567 161610      MOV   R5,ERRNBR      ;GET INITIAL ERROR NUMBER.
3439 022160 062767 000004 161602  ADD   #,ERRNBR      ;CALCULATE INITIAL ERRNBR + 4.
3440 022166 012767 016470 161600  MOV   #ER0503,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3441      ;PRINT ERROR REPORT.
3442 022174      ERROR      ;
3443 022176 104460      >>>> ERROR <<<<<.
3444      CLC           TRAP   C$ERROR
3445 022200      60$:  PASS      ;RESTORE GPRS,
3446 022202 004736      RTS   PC   JSR   PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
                                ; CARRY - SUCCESS FLAG (SET IF FIFO IS PURGED).

```

GLOBAL SUBROUTINE

- RXIE0 -

```

3448 .SBTTL GLOBAL SUBROUTINE - RXIE0 -
3449 ;** *****
3450 ;* - RECEIVER INTERRUPT DISABLE -
3451 ;* This routine is used to disable receiver interrupts in the DHV11.
3452 ;*
3453 ;* INPUTS: NONE.
3454 ;*
3455 ;* OUTPUTS: The RX.INT.ENBL bit is cleared in the DUT CSR.
3456 ;* IESTST -contains the updated status of the TX and RX interrupt
3457 ;* enable bits.
3458 ;*
3459 ;* CALLING SEQUENCE: JSR PC,RXIE0
3460 ;*
3461 ;* COMMENTS: The contents of the indirect address register field in
3462 ;* the DUT CSR are destroyed.
3463 ;*
3464 ;* SUBORDINATE ROUTINES CALLED: NONE.
3465 ;-- *****
3466 022204 010046 RXIE0:: MOV RO,-(SP) ;SAVE CONTENTS OF RO ON THE STACK.
3467 022206 104440 GETPRI -(SP) ;SAVE PROCESSOR PRIORITY ON STACK.
022210 010046 TRAP C$GPRI
3468 022212 SETPRI #PRI07 ;IGNORE ANY INTERRUPT THAT MAY BE GENERATED.
022212 012700 000340 MOV RO,-(SP)
022216 104441 MOV #PRI07,RO
3469 022220 042767 137777 160102 BIC #137777,IESTAT ;CLEAR RX.INT.ENBL BIT IN IESTAT.
3470 022226 016777 160076 160010 MOV IESTAT,@CSRA ;DISABLE RX INTERRUPTS.
3471 022234 SETPRI (SP)+ ;ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
022234 012600 MOV (SP)+,RO
022236 104441 TRAP C$SPRI
3472 022240 012600 MOV (SP)+,RO
3473 022242 000207 RTS PC ;RESTORE RO.

```

GLOBAL SUBROUTINE

- RXIE1 -

```

3475 .SBTTL GLOBAL SUBROUTINE - RXIE1 -
3476 ;** *****
3477 ;* - RECEIVER INTERRUPT ENABLE -
3478 ;* This routine is used to enable receiver interrupts in the DHV11.
3479 ;*
3480 ;* INPUTS: NONE.
3481 ;*
3482 ;* OUTPUTS: The RX.INT.ENBL bit is set in the DUT CSR.
3483 ;* IESTST -contains the updated status of the TX and RX interrupt
3484 ;* enable bits.
3485 ;*
3486 ;* CALLING SEQUENCE: JSR PC,RXIE1
3487 ;*
3488 ;* COMMENTS: The contents of the indirect address register field in
3489 ;* the DUT CSR are destroyed.
3490 ;*
3491 ;* SUBORDINATE ROUTINES CALLED: NONE.
3492 ;-- *****
3493
3494 022244 052767 000100 160056 RXIE1:: BIS #BIT06,IESTAT ;SET RX.INT.ENBL BIT IN IESTAT.
3495 022252 042767 137677 160050 BIC #137677,IESTAT ;CLEAR ALL OTHER BITS, EXCEPT TX AND RX I.E.
3496 022260 016777 160044 157756 MOV IESTAT,@CSRA ;ENABLE RX INTERRUPTS.
3497 022266 000207 RTS PC

```

GLOBAL SUBROUTINE

- SAVBMP -

3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534

022270 004567 161502
022274 016704 160224
022300 116724 160022
022304 005204
022306 042702 177400
022312 010224
022314 020427 002726
022320 103402
022322 162704 000004
022326 010467 160172
022332
022332 004736
022334 000207

```
.SBTTL GLOBAL SUBROUTINE - SAVBMP -
;* *****
;* - Save BMP codes Routine -
;* This routine saves the parameter passed in, onto the BMP code queue
;* together with the number of the currently executing test.
;*
;* INPUTS: R2 - Contains the BMP code that is to be placed on the queue.
;* BMPCQP - Contains address of next location in the bmp queue.
;* BMPCQB - Label at base of the BMP code queue.
;* BMPCQE - Label of next location after the end of the BMP queue.
;* TSTNUM - Contains the number of the current test.
;*
;* OUTPUTS: BMPCQP - Incremented by 4.
;* The contents of the BMP code queue are updated.
;*
;* CALLING SEQUENCE: JSR PC,SAVBMP
;*
;* COMMENTS: If the overflow occurs then the last location will be
;* overwritten by any subsequent attempts to update the queue.
;*
;* SUBORDINATE ROUTINES CALLED: None.
;-- *****
```

```
SAVBMP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
JSR ;GET THE POINTER TO THE NEXT LOCATION IN QUEUE.
MOV BMPCQP,R4 ;SAVE THE CURRENT TEST NUMBER ON THE QUEUE.
MOVB TSTNUM,(R4) ;INCREMENT THE POINTER TO GIVE AN EVEN ADDRESS.
INC R4 ;CLEAR THE UNWANTED BITS FROM THE BMP CODE.
BIC #177400,R2 ;SAVE THE BMP CODE ON THE QUEUE.
MOV R2,(R4) ;CHECK IF OVERFLOW WILL OCCUR THE NEXT TIME.
CMP R4,#BMPCQE ;GO SAVE THE POINTER IF WE WILL NOT OVERFLOW.
BLO 2$ ;RESET THE POINTER TO THE LAST LOCATION IN QUE.
SUB #4,R4 ;SAVE THE POINTER.
2$: MOV R4,BMPCQP
60$: PASS ;RESTORE GPRS.
JSR PC,@(SP) ;RETURN TO PREG05 SUBRT.
RTS PC
```

GLOBAL SUBROUTINE

- SKPSTS -

```

3536 .SBTTL GLOBAL SUBROUTINE - SKPSTS -
3537 ;* *****
3538 ;* - Skip Selftest Routine -
3539 ;* This subroutine is used to skip the selftest after a DUT reset has been
3540 ;* initiated. It must be entered immediately after setting the DUT Master
3541 ;* Reset routine or after the execution of a bus reset (because of timing
3542 ;* considerations).
3543 ;*
3544 ;* INPUTS: CSRA - Contains address of the DUT CSR.
3545 ;* TXBFCA - Contains address of DUT DMA Buffer Count register.
3546 ;*
3547 ;* OUTPUTS: Skip selftest codes are written to the DUT registers.
3548 ;*
3549 ;* CALLING SEQUENCE: JSR PC,SKPSTS
3550 ;*
3551 ;* COMMENTS:
3552 ;*
3553 ;* SUBORDINATE ROUTINES CALLED: DELAY.
3554 ;* - *****
3555
3556 022336 SKPSTS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
022336 004567 161434 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3557 022342 012704 000012 MOV #10.,R4 ;PASS DELAY VALUE OF 10 MILLI-SECONDS.
3558 022346 004767 175726 JSR PC,DELAY ;DELAY FOR 10 MILLI-SECONDS.
3559 ;*
3560 ; Write skip self-test code (52525) to all the indexed DUT Registers.
3561 ; -
3562 022352 012701 000050 MOV #NUMLNS!BIT05,R1 ;FORM IND.ADR.REG FIELD (PLUS M.R. BIT) WORD.
3563 ;THE ABOVE INCLUSION OF THE M.R. BIT IS NECESSARY BECAUSE OF THE
3564 ; LACK OF A M.R. BIT WRITE LOCK-OUT ON THE DHV11-M.
3565 022356 012703 052525 MOV #52525,R3 ;INITIALISE THE SKIP SELF-TEST CODE.
3566 022362 005301 4$: DEC R1 ;SELECT THE NEXT SET OF DEVICE REGISTERS.
3567 022364 016704 157654 MOV CSRA,R4 ;GET THE ADDRESS OF THE CSR OF THE DUT.
3568 022370 010124 MOV R1,(R4)+ ;SELECT A BANK OF DUT REGISTERS.
3569 022372 010324 6$: MOV R3,(R4)+ ;WRITE THE CODE TO A DUT REGISTER.
3570 022374 020467 157662 CMP R4,TXBFCA ;COMPARE POINTER WITH LAST REGISTER ADDRESS.
3571 022400 103774 BLO 6$ ;LOOP IF NOT ALL REGS DONE IN THIS BANK.
3572 022402 032701 000017 BIT #17,R1 ;TEST FOR IND.ADR.REG FIELD DECREMENTED TO 0.
3573 022406 001365 BNE 4$ ;LOOP UNTIL ALL REGISTERS CONTAIN THE CODE.
3574
3575 022410 60$: PASS ;RESTORE GPRS.
022410 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3576 022412 000207 RTS PC

```

GLOBAL SUBROUTINE

- SWAPO -

3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629

022414 010046

022416 010146
022420 010246
022422 010346
022424 010446
022426 010546

022430 012700 002450
022434 012001
022436 012002
022440 012003
022442 012004
022444 012005

022446 012640
022450 012640
022452 012640
022454 012640
022456 012640

022460 012600

022462 000207

```

.SBTTL GLOBAL SUBROUTINE - SWAPO -
;+ *****
;* - Swap GPRs With GPR Set 0 Routine -
;* This subroutine swaps the present contents of GPRs R1 thru R5 with
;* the contents of the number zero GPR save area. The contents of R0
;* are not altered by this subroutine.
;*
;* INPUTS: GPR contents R1 thru R5.
;* GPRS0B - Label at base of GPR save area number zero.
;*
;* OUTPUTS: R1 thru R5 contain the previous contents of GPR save area
;* zero words 1 thru 5 respectively.
;* GPRS0 - GPR save area 0 words 1 thru 5, contain previous
;* contents of GPRs R1 thru R5 respectively.
;*
;* CALLING SEQUENCE: JSR PC,SWAPO
;*
;* COMMENTS: The state of the CARRY flag is not altered by this routine.
;*
;* SUBORDINATE ROUTINES CALLED: None.
;-- *****
SWAPO:: MOV R0,-(SP) ;SAVE THE CONTENTS OF R0.
;+
; Load the stack from the GPRs.
;-
MOV R1,-(SP) ;SAVE THE CONTENTS OF R1.
MOV R2,-(SP) ;SAVE THE CONTENTS OF R2.
MOV R3,-(SP) ;SAVE THE CONTENTS OF R3.
MOV R4,-(SP) ;SAVE THE CONTENTS OF R4.
MOV R5,-(SP) ;SAVE THE CONTENTS OF R5.
;+
; Load the GPRs from the GPR save area 0.
;-
MOV #GPRS0B,R0 ;GET THE BASE ADDRESS OF GPR SAVE AREA 0.
MOV (R0)+,R1 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 1.
MOV (R0)+,R2 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 2.
MOV (R0)+,R3 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 3.
MOV (R0)+,R4 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 4.
MOV (R0)+,R5 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 5.
;+
; Load the GPR save area 0 from the stack.
;-
MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 5 WITH SAVED R5.
MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 4 WITH SAVED R4.
MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 3 WITH SAVED R3.
MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 2 WITH SAVED R2.
MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 1 WITH SAVED R1.
MOV (SP)+,R0 ;RESTORE THE INITIAL VALUE OF R0.
RTS PC

```

GLOBAL SUBROUTINE

- TSABRT -

```

3631 .SBTTL GLOBAL SUBROUTINE - TSABRT -
3632 ;** *****
3633 ;* - TEST ABORT ROUTINE -
3634 ;* This subroutine is used when a non-test related error has been found
3635 ;* during the execution of the current test.
3636 ;* It is used to inform the operator that the current test has been
3637 ;* aborted.
3638 ;*
3639 ;* INPUTS: ERRMSG - Contains the name of the current test.
3640 ;* ERRNBR - Contains the correct error number.
3641 ;* The remainder of the ERRBL is correctly initialised.
3642 ;*
3643 ;* OUTPUTS: Messages are reported to the operator.
3644 ;*
3645 ;* CALLING SEQUENCE: JSR PC,TSABRT
3646 ;*
3647 ;* COMMENTS:
3648 ;*
3649 ;* SUBORDINATE ROUTINES CALLED: ER1603.
3650 ;-- *****
3651
3652 022464 TSABRT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
022464 004567 161306 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3653 022470 012701 022506 MOV #2$,R1 ;PASS ADDRESS OF FIRST MESSAGE TO BE REPORTED.
3654 022474 012767 017010 161272 MOV #ER1603,ERRBLK ;SET UP THE ERROR REPORTING ROUTINE.
3655 022502 ERROR ; >>>> ERROR <<<<<.
022502 104460 ; TRAP C$ERROR
3656 022504 000432 BR 60$ ;
3657 022506 040 116 117 2$: .ASCIZ / NON-RELATED TEST ERROR FOUND DURING TEST EXECUTION/
022511 116 055 122
022514 105 114 101
022517 124 105 104
022522 040 124 105
022525 123 124 040
022530 105 122 122
022533 117 122 040
022536 106 117 125
022541 116 104 040
022544 104 125 122
022547 111 116 107
022552 040 124 105
022555 123 124 040
022560 105 130 105
022563 103 125 124
022566 111 117 116
022571 000
3658 .EVEN
3659 022572 60$: PASS ;RESTORE GPRS.
022572 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3660 022574 000207 RTS PC

```


GLOBAL SUBROUTINE

- TXDSBL -

```

3662 .SBTTL GLOBAL SUBROUTINE - TXDSBL -
3663 ;* *****
3664 ;* - Transmitter Disable -
3665 ;* This subroutine is used to disable transmission on selected lines by,
3666 ;* clearing the associated TX.ENABLE bit on the DUT.
3667 ;*
3668 ;* INPUTS: R5 - Bit's set correspond to lines on which to clear TX.ENABLE.
3669 ;* CSRA - Contains the address of the DUT CSR.
3670 ;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
3671 ;* NUMLNS - Equated to be the maximum number of lines available.
3672 ;* TXAD2A Contains the address of the TBUFFAD2 register.
3673 ;*
3674 ;* OUTPUTS: R5 - Bit's set indicate the initial states of all TX.ENABLE bits.
3675 ;* TBUFFAD2 - The state of the TX.ENABLE bit may be altered.
3676 ;* The contents of the I. ADD.REG field in the CSR are destroyed.
3677 ;*
3678 ;* CALLING SEQUENCE: JSR PC,TXDSBL
3679 ;*
3680 ;* COMMENTS:
3681 ;*
3682 ;* SUBORDINATE ROUTINES CALLED: NONE.
3683 ;*
3684 ;* *****
3685 TXDSBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3686 022576 004567 161174 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3687 022602 010500 MOV R5,R0 ;COPY BIT MAP OF LINES TO DISABLE TRANSMISSION.
3688 022604 012701 000001 MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
3689 022610 016702 157444 MOV TXAD2A,R2 ;GET THE ADDRESS OF THE TBUFFAD2 REGISTER.
3690 022614 005202 INC R2 ;GET THE ADDRESS OF THE MSBYTE OF TBUFFAD2 REG.
3691 022616 012703 000010 MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER PLUS ONE.
3692 022622 016704 157502 MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
3693 022626 005005 CLR R5 ;LOG POSSIBLE TX DISABLED ON ALL LINES.
3694 ;*
3695 ; Select every line in turn, and log the state of each TX.ENABLE bit.
3696 022630 010477 157410 2$: MOV R4,@CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
3697 022634 105712 TSTB (R2) ;CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.
3698 022636 100001 BPL 4$ ;SKIP NEXT INSTRUCTION IF TX.ENABLE CLEAR.
3699 022640 050105 BIS R1,R5 ;LOG TX ENABLE BIT SET FOR SELECTED LINE.
3700 ;*
3701 ; Clear TX.ENABLE on lines that have a corresponding bit set in the tx disable
3702 ; line bit map.
3703 ;*
3704 022642 030100 4$: BIT R1,R0 ;CHECK STATE OF DISABLE LINE BIT MAP.
3705 022644 001402 BEQ 6$ ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
3706 022646 142712 000200 BICB #BIT7,(R2) ;CLEAR TX.ENABLE BIT ON SELECTED LINE.
3707 022652 005204 6$: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
3708 022654 006301 ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
3709 022656 005303 DEC R3 ;DECREMENT LINE NUMBER.
3710 022660 001363 BNE 2$ ;LOOP TO CHECK NEXT LINE.
3711 ;*
3712 022662 60$: PASS R5 ;RESTORE GPRS,EXCEPT
3713 022662 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
3714 022666 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
;R5 - PREVIOUS STATES OF ALL TX.ENABLE BITS.
3713
3714 022670 000207 RTS PC

```

GLOBAL SUBROUTINE

- TXENBL -

```

3716 .SBTTL GLOBAL SUBROUTINE - TXENBL -
3717 ;+ *****
3718 ;* - Transmitter Enable -
3719 ;* This subroutine is used to enable transmission on selected lines by
3720 ;* setting the associated TX.ENABLE bit on the DUT.
3721 ;*
3722 ;* INPUTS: R5 - Bit's set correspond to lines on which to set TX.ENABLE.
3723 ;* CSRA - Contains the address of the DUT CSR.
3724 ;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
3725 ;* NUMLNS - Equated to be the maximum number of lines available.
3726 ;* TXAD2A - Contains the address of the TBUFFAD2 register.
3727 ;*
3728 ;* OUTPUTS: R5 - Bit's set indicate previously disabled lines.
3729 ;* TBUFFAD2 - The state of the TX.ENABLE bit may be altered.
3730 ;* The contents of the IND.ADD.REG field in the CSR are destroyed.
3731 ;*
3732 ;* CALLING SEQUENCE: JSR PC,TXENBL
3733 ;*
3734 ;* COMMENTS:
3735 ;*
3736 ;* SUBORDINATE ROUTINES CALLED: NONE.
3737 ; *****
3738 ;
3739 022672 TXENBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
022672 004567 161100 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3740 022676 010500 MOV R5,R0 ;COPY BIT MAP OF LINES TO ENABLE.
3741 022700 012701 000001 MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
3742 022704 016702 157350 MOV TXAD2A,R2 ;GET THE ADDRESS OF THE TBUFFAD2 REGISTER.
3743 022710 005202 INC R2 ;GET THE ADDRESS OF THE MSBYTE OF TBUFFAD2 REG.
3744 022712 012703 000010 MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER.
3745 022716 016704 157406 MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
3746 022722 005005 CLR R5 ;CLEAR TX.ENABLE BIT LOG OF DISABLED LINES.
3747 ;+
3748 ; Select every line in turn,and log any TX.ENABLE bit that is clear.
3749 ;-
3750 022724 010477 157314 2$: MOV R4,@CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
3751 022730 105712 TSTB (R2) ;CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.
3752 022732 100401 BMI 4$ ;SKIP NEXT INSTRUCTION IF TX.ENABLE SET.
3753 022734 050105 BIS R1,R5 ;LOG TX ENABLE BIT CLEAR FOR SELECTED LINE.
3754 ;+
3755 ; Set TX.ENABLE on lines that have a corresponding bit set in the tx enable
3756 ; line bit map.
3757 ;-
3758 022736 030100 4$: BIT R1,R0 ;CHECK STATE OF TX.ENABLE LINE BIT MAP.
3759 022740 001402 BEQ 6$ ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
3760 022742 152712 000200 BISB #BIT7,(R2) ;ENABLE TRANSMISSION ON SELECTED LINE.
3761 022746 005204 6$: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
3762 022750 006301 ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
3763 022752 005303 DEC R3 ;DECREMENT LINE NUMBER.
3764 022754 001363 BNE 2$ ;LOOP TO CHECK NEXT LINE.
3765 ;-
3766 022756 60$: PASS R5 ;RESTORE GPRS,EXCEPT
022756 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
022762 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3767 ;R5 - LINE BIT MAP CORRESPONDING TO THE
3768 ; PREVIOUS LINES THAT WERE DISABLED.
3769 022764 000207 RTS PC

```

GLOBAL SUBROUTINE

- TXIE0 -

```

3771 .SBTTL GLOBAL SUBROUTINE - TXIE0 -
3772 ;** *****
3773 ;* - TRANSMITTER INTERRUPT DISABLE -
3774 ;* This routine is used to disable transmitter interrupts in the DHV11.
3775 ;*
3776 ;* INPUTS: NONE.
3777 ;*
3778 ;* OUTPUTS: The TX.INT.ENBL bit is cleared in the DUT CSR.
3779 ;* IESTST -contains the updated status of the TX and RX interrupt
3780 ;* enable bits.
3781 ;*
3782 ;* CALLING SEQUENCE: JSR PC,TXIE0
3783 ;*
3784 ;* COMMENTS: The contents of the indirect address register field in
3785 ;* the DUT CSR are destroyed.
3786 ;*
3787 ;* SUBORDINATE ROUTINES CALLED: NONE.
3788 ;-- *****
3789 022766 010046 TXIE0:: MOV RO,-(SP) ;SAVE CONTENTS OF RO ON THE STACK.
3790 022770 104440 GETPRI -(SP) ;SAVE CURRENT PROCESSOR PRIORITY ON THE STACK.
022772 010046 TRAP C$GPRI
3791 022774 SETPRI #PRI07 ;IGNORE ANY INTERRUPTS THAT MAY BE GENERATED.
022774 012700 000340 MOV RO,-(SP)
023000 104441 TRAP C$GPRI
3792 023002 042767 177677 157320 BIC #177677,IESTAT ;CLEAR TX.INT.ENBL BIT IN IESTAT.
3793 023010 016777 157314 157226 MOV IESTAT,@CSRA ;DISABLE TX INTERRUPTS.
3794 023016 SETPRI (SP)+ ;ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
023016 012600 MOV (SP)+,RO
023020 104441 TRAP C$SPRI
3795 023022 012600 MOV (SP)+,RO ;RESTORE RO.
3796 023024 000207 RTS PC

```

GLOBAL SUBROUTINE

- TXIE1 -

```

3798 .SBTTL GLOBAL SUBROUTINE - TXIE1 -
3799 ;* *****
3800 ;* - TRANSMITTER INTERRUPT ENABLE -
3801 ;* This routine is used to enable transmitter interrupts in the DHV11.
3802 ;*
3803 ;* INPUTS: NONE.
3804 ;*
3805 ;* OUTPUTS: The TX.INT.ENBL bit is set in the DUT CSR.
3806 ;* IESTST contains the updated status of the TX and RX interrupt
3807 ;* enable bits.
3808 ;*
3809 ;* CALLING SEQUENCE: JSR PC,TXIE1
3810 ;*
3811 ;* COMMENTS: The contents of the indirect address register field in
3812 ;* the DUT CSR are destroyed.
3813 ;*
3814 ;* SUBORDINATE ROUTINES CALLED: NONE.
3815 ;* *****
3816
3817 023026 052767 040000 157274 TXIE1:: BIS #BIT14,IESTAT ;SET TX.INT.ENBL BIT IN IESTAT.
3818 023034 042767 137677 157266 BIC #137677,IESTAT ;CLEAR ALL BITS EXCEPT TX RX I.E BITS.
3819 023042 016777 157262 157174 MOV IESTAT,@CSRA ;DISABLE TX INTERRUPTS.
3820 023050 000207 RTS PC

```

GLOBAL SUBROUTINE

- UNSDIV -

```

3822 .SBTTL GLOBAL SUBROUTINE UNSDIV
3823 ;* *****
3824 ;* - Unsigned Divide Routine -
3825 ;* This subroutine is used to divide a 32 bit unsigned dividend by a
3826 ;* 16 bit unsigned divisor giving a 16 bit quotient. All numbers are
3827 ;* considered to be unsigned. A success flag is not set on return if
3828 ;* the quotient was too big to be contained in 16 bits.
3829 ;*
3830 ;* INPUTS: R1 - The divisor, unsigned, 16 bits.
3831 ;* R2 - Most significant word of the dividend, unsigned, 16 bits.
3832 ;* R3 - Least significant word of the dividend, unsigned, 16 bits.
3833 ;*
3834 ;* OUTPUTS: R1 - Quotient, unsigned, 16 bits (17777 if overflow).
3835 ;* CARRY - Success flag, set if complete quotient fits in 16 bits.
3836 ;*
3837 ;* CALLING SEQUENCE: JSR PC,UNSDIV
3838 ;*
3839 ;* COMMENTS: If the divisor is 0 the quotient is returned as all ones
3840 ;* (17777) and the carry is clear regardless of the dividend.
3841 ;*
3842 ;* SUBORDINATE ROUTINES CALLED: None.
3843 ;*
3844 ;* *****
3845 023052 UNSDIV:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023052 004567 160720 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3846 ;+
3847 ; Check for quotient greater than 16 bits condition.
3848 ;-
3849 023056 010204 MOV R2,R4 ;GET MSW OF DIVIDEND FOR SUBTRACT.
3850 023060 160104 SUB R1,R4 ;SUBTRACT DIVISOR FROM MSW OF DIVIDEND.
3851 023062 103403 BCS 2$ ;IF IT DIDN'T GO, WE HAVE QUOTIENT < 16 BITS.
3852 023064 012701 177777 MOV #1,R1 ;SET QUOTIENT TO ALL ONES (177777),
3853 023070 000442 BR 60$ ;EXIT WITH CARRY CLEAR.
3854 ;+
3855 ; Set up counters and various working GPRs.
3856 ;-
3857 023072 005004 2$: CLR R4 ;CLEAR THE LSW OF THE DIVISOR.
3858 023074 000241 CLC ;CLEAR CARRY FOR THE SHIFT OF THE DIVISOR.
3859 023076 006001 ROR R1 ; DIVISOR BY
3860 023100 006004 ROR R4 ; 2(UNSIGNED)
3861 023102 012700 000020 MOV #16.,R0 ;SET UP INITIAL SHIFT COUNT TO 16.
3862 ;+
3863 ; The subtract and shift loop.
3864 ;-
3865 023106 010246 4$: MOV R2,-(SP) ;SAVE MSWORD OF DIVIDEND.
3866 023110 010346 MOV R3,-(SP) ;SAVE LSWORD OF DIVIDEND.
3867 023112 160403 SUB R4,R3 ;LSWORD DIVIDEND - LSWORD OF DIVISOR.
3868 023114 005602 SBC R2 ;MSWORD DIVIDEND - BORROW
3869 023116 103402 BCS 6$ ;IF BORROW FROM BORROW SUBTRACT, IT DIDN'T GO.
3870 023120 160102 SUB R1,R2 ;MSWORD DIVIDEND - MSWORD OF DIVISOR.
3871 023122 103003 BCC 8$ ;IF NO BORROW, IT WENT, CARRY IS CLEAR.
3872 ;+
3873 ; It didn't go, so we shift a 1 into the quotient (complemented later).
3874 ; Carry is set.
3875 ;-
3876 023124 012603 6$: MOV (SP)+,R3 ;RESTORE LSWORD OF DIVIDEND.
3877 023126 012602 MOV (SP)+,R2 ;RESTORE MSWORD OF DIVIDEND.

```

GLOBAL SUBROUTINE

- UNSDIV -

```

3878 023130 000401      BR      10$      ;GOTO SHIFT 1 INTO THE QUOTIENT.
3879
3880                    ;+
3881                    ; It went, so we restore the stack and shift a 0 into quotient (will be
3882                    ; complemented later).  Carry is clear.
3883 023132 012626      8$:  MOV      (SP)+,(SP)+      ;POP THE SAVED DIVIDEND OFF OF THE STACK.
3884
3885                    ;+
3886                    ; Shift the result of the subtract attempt into the quotient shift reg.
3887 023134 006105      10$:  ROL      R5              ;SHIFT NEXT BIT INTO THE INVERTED QUOTIENT.
3888 023136 000241      CLC                    ;DIVIDE THE
3889 023140 006001      ROR      R1              ; DEVISOR BY
3890 023142 006004      ROR      R4              ; 2 (UNSIGNED).
3891 023144 005300      DEC      R0              ;COUNT THIS SHIFT AND SUBTRACT.
3892 023146 001357      BNE      4$              ;LOOP FOR ANOTHER SHIFT & SUB IF NOT DONE.
3893 023150 005105      COM      R5              ;GET QUOTIENT FROM INVERTED QUOTIENT.
3894
3895                    ;+
3896                    ; Now we either round up or leave quotient alone.
3897 023152 000241      CLC                    ;CLEAR THE CARRY FOR THE SHIFT OF THE DIVIDEND.
3898 023154 006103      ROL      R3              ;MULTIPLY LSWORD OF DIVIDEND BY 2, MSWORD IS 0.
3899 023156 103402      BCS      12$            ;IF CARRY FROM SHIFT, ROUND UP.
3900 023160 160403      SUB      R4,R3          ;SUBTRACT DIVISOR FROM DIVIDEND.
3901 023162 103403      BCS      14$            ;IF BORROW, DON'T ROUND UP.
3902
3903                    ;+
3904                    ; Round up, extra subtract went.
3905 023164 005205      12$:  INC      R5              ;INCREMENT THE QUOTIENT BY ONE.
3906 023166 001001      BNE      14$            ;IF NO OVERFLOW, WE LEAVE THE ROUND UP.
3907 023170 005305      DEC      R5              ;DON'T LET ROUNDING CAUSE OVERFLOW.
3908
3909                    ;+
3910                    ; All done, pass quotient and exit.
3911 023172 010501      14$:  MOV      R5,R1          ;PASS QUOTIENT BACK IN R1.
3912 023174 000261      SEC                    ;INDICATE NO OVERFLOW.
3913
3914 023176                60$:  PASS      R1              ;RESTORE GPRS, LEAVE THE FOLLOWING INTACT:
023176 010166 000004      MOV      R1,R1SLOT(SP)    ;PUT R1 IN STACK SLOT.
023202 004736                JSR      PC,@(SP)+      ;RETURN TO PREG05 SUBRT.
3915
3916 023204 000207      RTS      PC              ;R1 - 16 BIT, UNSIGNED QUOTIENT,
;CARRY - SET INDICATES NO OVERFLOW (SUCCESS).

```

GLOBAL SUBROUTINE

- WAIBIC -

```

3918 .SBTTL GLOBAL SUBROUTINE - WAIBIC -
3919 ;** *****
3920 ;* - Wait For Bit Clear Routine -
3921 ;* This subroutine waits for the specified bit to become clear. If the
3922 ;* specified bit goes to a clear state within the specified time-out
3923 ;* period a success indication is returned by this routine.
3924 ;* The last value which is read looking for the condition is returned to
3925 ;* allow the use of this routine to look for destructive read conditions.
3926 ;*
3927 ;* INPUTS: R1 - Time-out value and bit number indication:
3928 ;* Bits 15 thru 12 - Number of bit to test (range 0 thru 15).
3929 ;* Bits 11 thru 0 - Time-out value in milli-seconds (4095 max).
3930 ;* R2 - Address of word containing the bit to test.
3931 ;* MSLCNT.
3932 ;*
3933 ;* OUTPUTS: R2 The last word which was read to check for the condition.
3934 ;* CARRY - Success flag (CARRY set if bit clr before time-out).
3935 ;*
3936 ;* CALLING SEQUENCE: MOV #130040,R1 ;PASS BIT 11 (13 OCTAL) AND
3937 ;* ; 32 (40 OCTAL) MS DELAY.
3938 ;* MOV #LABEL,R2 ;TEST BIT IN WORD AT "LABEL".
3939 ;* JSR PC,WAIBIC ;WAIT 32 MS FOR BIT 11 TO CLR.
3940 ;*
3941 ;* COMMENTS:
3942 ;*
3943 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
3944 ;-- *****
3945
3946 023206 WAIBIC:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023206 004567 160564 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3947 023212 010204 MOV R2,R4 ;SET UP THE ADDRESS PARAMETER FOR MSLGET.
3948 023214 010102 MOV R1,R2
3949 023216 042701 170000 BIC #170000,R1 ;SEPERATE DELAY COUNT OUT OF PASSED PARAMETER.
3950 023222 042702 007777 BIC #7777,R2 ;SEPERATE LINE NUMBER FIELD OF PASSED PARAM.
3951 023226 000302 SWAB R2 ;PUT LINE NUMBER FIELD IN LSBYTE.
3952 023230 006202 ASR R2 ;SHIFT THE LINE NUMBER FIELD INTO THE PROPER
3953 023232 006202 ASR R2 ; POSITION TO USE IT AS A WORD TABLE OFFSET
3954 023234 006202 ASR R2 ; FOR THE TABLE LOOKUP OF THE LINE BIT MAP.
3955 023236 016202 002410 MOV BITTBL(R2),R2 ;GET BIT MAP OF LINE TO TEST FROM TABLE.
3956 023242 005003 CLR R3 ;INDICATE THAT THE BIT SHOULD BE CLR.
3957 023244 004767 175070 JSR PC,MSLGET ;WAIT FOR THE BIT TO BE CLR WITHIN TIME-OUT.
3958 ; CARRY IS CORRECT UPON MSLGET RETURN.
3959 023250 010002 MOV R0,R2 ;PASS LAST VALUE READ AS OUTPUT PARAMETER.
3960 023252 010266 000006 60$: PASS R2 ;RESTORE GPRS, EXCEPT THE FOLLOWING:
023252 004736 MOV R2,R2SLOT(SP) ;PUT R2 IN STACK SLOT.
023256 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3961 ; R2 - LAST VALUE READ LOOKING FOR CONDITION.
3962 023260 000207 RTS PC ; CARRY - SUCCESS FLAG (SET IF BIT FOUND CLR).

```

GLOBAL SUBROUTINE

- WAIBIS -

```

3964 .SBTTL GLOBAL SUBROUTINE - WAIBIS -
3965 ;** *****
3966 ;* - Wait For Bit Set Routine -
3967 ;* This subroutine waits for the specified bit to become set. If the
3968 ;* specified bit goes to a set state within the specified time-out
3969 ;* period a success indication is returned by this routine.
3970 ;* The last value which is read looking for the condition is returned to
3971 ;* allow the use of this routine to look for destructive read conditions.
3972 ;*
3973 ;* INPUTS: R1 - Time-out value and bit number indication:
3974 ;* Bits 15 thru 12 - Number of bit to test (range 0 thru 15).
3975 ;* Bits 11 thru 0 - Time-out value in milli-seconds (4095 max).
3976 ;* R2 - Address of word containing the bit to test.
3977 ;* MSLCNT.
3978 ;*
3979 ;* OUTPUTS: R2 - The last word which was read to check for the condition.
3980 ;* CARRY - Success flag (CARRY set if bit set before time-out).
3981 ;*
3982 ;* CALLING SEQUENCE: MOV #130040,R1 ;PASS BIT 11 (13 OCTAL) AND
3983 ;* ; 32 (40 OCTAL) MS DELAY.
3984 ;* MOV #LABEL,R2 ;TEST BIT IN WORD AT "LABEL".
3985 ;* JSR PC,WAIBIS ;WAIT 32 MS FOR BIT 11 TO SET.
3986 ;*
3987 ;* COMMENTS:
3988 ;*
3989 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
3990 ;-- *****
3991
3992 023262 WAIBIS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023262 004567 160510 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
023262 MOV R2,R4 ;SET UP THE ADDRESS PARAMETER FOR MSLGET.
023262 MOV R1,R2
023270 010102 MOV #170000,R1 ;SEPERATE DELAY COUNT OUT OF PASSED PARAMETER.
023272 042701 170000 BIC #7777,R2 ;SEPERATE LINE NUMBER FIELD OF PASSED PARAM.
023276 042702 007777 BIC #7777,R2 ;PUT LINE NUMBER FIELD IN LSBYTE.
023302 000302 SWAB R2 ;SHIFT THE LINE NUMBER FIELD INTO THE PROPER
023304 006202 ASR R2 ; POSITION TO USE IT AS A WORD TABLE OFFSET
023306 006202 ASR R2 ; FOR THE TABLE LOOKUP OF THE LINE BIT MAP.
023310 006202 ASR R2
023312 016202 002410 MOV BITTBL(R2),R2 ;GET BIT MAP OF LINE TO TEST FROM TABLE.
023316 010203 MOV R2,R3 ;INDICATE THAT THE BIT SHOULD BE SET.
023320 004767 175014 JSR PC,MSLGET ;WAIT FOR THE BIT TO BE SET WITHIN TIME-OUT.
023324 010002 MOV R0,R2 ; CARRY IS CORRECT UPON MSLGET RETURN.
023326 010266 000006 60$: PASS R2 ;PASS LAST VALUE READ AS OUTPUT PARAMETER.
023332 004736 MOV R2,R2SLOT(SP) ;RESTORE GPRS, EXCEPT THE FOLLOWING:
; PUT R2 IN STACK SLOT.
; RETURN TO PREG05 SUBRT.
023332 004736 JSR PC,@(SP)+
023334 000207 RTS PC ; R2 - LAST VALUE READ LOOKING FOR CONDITION.
; CARRY - SUCCESS FLAG (SET IF BIT FOUND SET).

```


GLOBAL SUBROUTINE

- WDPDR -

```

4010 .SBTTL GLOBAL SUBROUTINE - WDPDR -
4011 ;* *****
4012 ;* - Write Data Pattern to Device Registers -
4013 ;* This routine writes a rotated data pattern to each of the 6 device
4014 ;* registers of each active line of the device under test.
4015 ;* The data pattern is rotated once after each write to a device register
4016 ;* on a particular line. The starting data pattern for each line
4017 ;* is rotated once after writing all the registers on a particular
4018 ;* line. This leads to the following data pattern:
4019 ;* Line 0, register 0 - shifted 0 bit positions
4020 ;* Line 0, register 1 - shifted 1 bit position
4021 ;*
4022 ;* Line 1, register 0 - shifted 1 bit position
4023 ;* Line 2, register 1 - shifted 2 bit positions
4024 ;*
4025 ;* Any bits fields in the device registers that cannot be altered
4026 ;* are masked out of the data pattern before it is written.
4027 ;* This routine will use either MOV, MOVB, BIS, BISB, BIC, or BICB
4028 ;* instructions. The upper or lower byte can be specified for writing.
4029 ;*
4030 ;* INPUTS: R2 - Used to pass in the data pattern to be rotated & written.
4031 ;* R3 - Byte indicator (- => lo byte, + => hi byte, 0 => both).
4032 ;* R4 - Operation type indicator (- => BIC, + => BIS, 0 => MOV).
4033 ;* ACTLNS - Bit map of the active lines on the device under test.
4034 ;* CSRA - Contains the CSR address of the Device under test.
4035 ;* DRADRT - Base address of device register address table.
4036 ;* LPRO - Equated to LPR reg offset from device CSR address.
4037 ;* NUMLNS - Number of lines on the device under test.
4038 ;* TXBFCO - Equated to TBUFFCT reg offset from device CSR address.
4039 ;* UNBTTB - Base address of the unused bit table.
4040 ;*
4041 ;* OUTPUTS: Device registers on all active device lines are modified.
4042 ;*
4043 ;* CALLING SEQUENCE: JSR PC,WDPDR
4044 ;*
4045 ;* COMMENTS: This routine does not write any data to the TX.CHAR registers.
4046 ;* The CSR is cleared except for the IND.ADR.REG field.
4047 ;*
4048 ;* SUBORDINATE ROUTINES CALLED: ROLDAP.
4049 ;*-- *****
4050
4051 023336 WDPDR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023336 004567 160434 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4052 ;*
4053 ; Set up outer loop which writes the data pattern to each line's registers
4054 ;*
4055 023342 005005 CLR R5 ;CLEAR LINE COUNTER TO SELECT LINE 0.
4056 ;*
4057 ; The outer loop follows. Each pass through this loop writes data to all of
4058 ; the device registers for a particular line if it is active.
4059 ;*
4060 023344 010204 2$: MOV R2,R4 ;SAVE THE OUTER LOOP DATA PATTERN.
4061 023346 010577 156672 MOV R5,@CSRA ;SET CSR IND.ADR.REG FIELD TO THIS LINE.
4062 023352 006305 ASL R5 ;TURN LINE NUMBER INTO A WORD OFFSET.
4063 023354 036567 002410 156654 BIT BITTEL(R5),ACTLNS
4064 023362 001451 BEQ 20$ ;LINE ACTIVE? NO, SKIP THIS LINE.
4065 023364 012701 000004 MOV @LPRO,R1 ;YES, INITIALIZE THE REGISTER OFFSET.

```

GLOBAL SUBROUTINE

- WDPDR -

```

4066      ;*
4067      ; The inner loop follows. Each pass through this loop writes data to a
4068      ; device register.
4069      ;-
4070 023370 010200      4$:  MOV    R2,R0
4071 023372 046100 002264      BIC    UNBITB(R1),R0      ;CLEAR BIT FIELDS FOR UNUSED REGISTER BITS.
4072 023376 016103 002244      MOV    DRADRT(R1),R3      ;GET THE ADDRESS OF THE DEVICE REGISTER.
4073 023402 005766 000010      TST    R3SLOT(SP)      ;CHECK THE OPERAND TYPE INDICATOR.
4074 023406 003402      BLE    6$      ;HIGH BYTE? NO, SKIP HIGH BYTE ADDRESS SET UP.
4075 023410 005203      INC    R3      ;YES, SET THE REG ADDRESS TO THE HIGH BYTE.
4076 023412 000300      SWAB   R0      ;MOVE HIGH BYTE DATA INTO THE LOW BYTE.
4077 023414 005766 000010      6$:  TST    R3SLOT(SP)      ;CHECK THE OPERAND TYPE INDICATOR.
4078 023420 001412      BEQ    12$      ;WORD ACCESS? YES, GO PERFORM WORD ACCESS.
4079
4080      ;*
4081      ;Perform byte access to the specified byte of the specified register.
4082      ;-
4082 023422 005766 000012      TST    R4SLOT(SP)      ;NO, CHECK THE ACCESS TYPE INDICATOR.
4083 023426 100403      BMI    8$      ;USE BIC? YES, GO PERFORM BICB INSTRUCTION.
4084 023430 001404      BEQ    10$      ;USE MOV? YES, GO PERFORM MOV B INSTRUCTION.
4085 023432 150013      BISB   R0,(R3)      ;NEITHER. PERFORM BISB ACCESS TO REGISTER.
4086 023434 000415      BR    18$
4087 023436 140013      8$:  BICB   R0,(R3)      ;PERFORM BICB ACCESS TO REGISTER.
4088 023440 000413      BR    18$
4089 023442 110013      10$: MOVB   R0,(R3)      ;PERFORM MOV B ACCESS TO REGISTER.
4090 023444 000411      BR    18$
4091
4092      ;*
4093      ;Perform word access to the specified register.
4094      ;-
4094 023446 005766 000012      12$: TST    R4SLOT(SP)      ;CHECK THE ACCESS TYPE INDICATOR.
4095 023452 100403      BMI    14$      ;USE BIC? YES, GO PERFORM BIC INSTRUCTION.
4096 023454 001404      BEQ    16$      ;USE MOV? YES, GO PERFORM MOV INSTRUCTION.
4097 023456 050013      BIS    R0,(R3)      ;NEITHER. PERFORM BIS ACCESS TO REGISTER.
4098 023460 000403      BR    18$
4099 023462 040013      14$: BIC    R0,(R3)      ;PERFORM BIC ACCESS TO REGISTER.
4100 023464 000401      BR    18$
4101 023466 010013      16$: MOV    R0,(R3)      ;PERFORM MOV ACCESS TO REGISTER.
4102
4103      ;*
4104      ; Prepare the data pattern and offset for the next register on this line.
4105      ;-
4105 023470 004767 176074      18$: JSR    PC,ROLDAP      ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
4106 023474 062701 000002      ADD    #2,R1      ;INCREMENT OFFSET FOR NEXT REGISTER.
4107 023500 020127 000016      CMP    R1,#TXBFC0      ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
4108 023504 003731      BLE    4$      ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
4109
4110      ;*
4111      ; Back into the outer loop. Now set up for next line. Loop if not done.
4112      ;-
4112 023506 010402      20$: MOV    R4,R2      ;SET UP TO ROTATE THE DATA PATTERN.
4113 023510 004767 176054      JSR    PC,ROLDAP      ;ROTATE THE DATA PATTERN.
4114 023514 006205      ASR    R5      ;CONVERT BACK TO LINE NUMBER FROM WORD OFFSET.
4115 023516 005205      INC    R5      ;COUNT THIS LINE.
4116 023520 020527 000010      CMP    R5,#NUMLNS      ;COMPARE LINE COUNT WITH NUMBER OF LINES.
4117 023524 002707      BLT    2$      ;LOOP IF SOME LINES NOT DONE.
4118
4119      60$: PASS
4120 023526 004736      JSR    PC,@(SP)      ;RESTORE GPRS.
4120 023530 000207      RTS    PC      ;RETURN TO PREG05 SLBRT.

```

GLOBAL SUBROUTINE

- WTWLNC -

```

4122 .SBTTL GLOBAL SUBROUTINE - WTWLNC -
4123 ;* *****
4124 ;* - Line Control Register Setup Routine -
4125 ;* This subroutine is used to set the Device Under Test (DUT) Line
4126 ;* Control Registers (LNCTRL) to the specified state. Only the LNCTRLS
4127 ;* for the specified lines are altered.
4128 ;*
4129 ;* INPUTS: R0 - New line parameters.
4130 ;* R5 - Bit map of lines to be altered.
4131 ;* CSRA - Contains address of the DUT CSR.
4132 ;* IESTAT - Contains the current state of the TX and RX interrupt
4133 ;* enable bits in the CSR.
4134 ;* LNCTRA - Contains address of the DUT LNCTRL registers.
4135 ;*
4136 ;* OUTPUTS: LNCTRL - Specified DUT Line Control Registers are altered.
4137 ;*
4138 ;* CALLING SEQUENCE: JSR PC,WTWLNC
4139 ;*
4140 ;* COMMENTS:
4141 ;*
4142 ;* SUBORDINATE ROUTINES CALLED: ALTFLD.
4143 ;* *****
4144 ;*
4145 023532 WTWLNC:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023532 004567 160240 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4146 ;*
4147 ; Set up the parameters for the call to ALTFLD.
4148 ;*
4149 023536 016701 156512 MOV LNCTRA,R1 ;SET UP THE REGISTER ADDRESS PARAMETER.
4150 023542 010002 MOV R0,R2 ;SET UP THE DESIRED REGISTER CONTENTS.
4151 023544 010503 MOV R5,R3 ;SET UP THE BIT MAP OF LINES TO ALTER.
4152 023546 012704 177777 MOV #-1,R4 ;SELECT ALL REGISTER BITS TO BE ALTERED.
4153 ;*
4154 ; Call the subroutine which alters the register contents.
4155 ;*
4156 023552 004767 174052 JSR PC,ALTFLD ;ALTER THE REGISTER CONTENTS.
4157 ;*
4158 023556 004736 60$: PASS ;RESTORE GPRS.
023556 000207 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4159 023560

```

GLOBAL SUBROUTINE

- WTWLNS -

```

4161 .SBTTL GLOBAL SUBROUTINE - WTWLNS -
4162 ;*****
4163 ;* - Write Word to all Lines routine -
4164 ;* This subroutine writes a specified word to the specified DHV device
4165 ;* register for each of the DHV lines. It could be used to clear all
4166 ;* of the LNCTRL registers or to initialize all of the LPR registers to
4167 ;* the same parameters.
4168 ;*
4169 ;* INPUTS: R1 - Address of the specified registers.
4170 ;* R2 - Word to write into the specified registers.
4171 ;* IESTAT - Saved states of the TX.IE and RX.IE bits.
4172 ;* MAPLNS - Equated to bit map of lines on device (8 for DHV11).
4173 ;* CSRA.
4174 ;*
4175 ;* OUTPUTS: DEVICE REGISTERS - Specified registers given new value.
4176 ;* CSR IND.ADR.REG field - Destroyed.
4177 ;* CSR Interrupt Enable bits - Set to states in IESTAT.
4178 ;*
4179 ;* CALLING SEQUENCE: JSR PC,WTWLNS
4180 ;*
4181 ;* COMMENTS: Note that the specified registers for all lines are altered
4182 ;* by this routine. This routine should NOT be used to alter
4183 ;* the states of partial register fields or to alter a register
4184 ;* for fewer than all of the lines.
4185 ;* The specified registers are read before being written.
4186 ;*
4187 ;* SUBROUTINES CALLED: ALTFLD.
4188 ;*****
4189
4190 023562 WTWLNS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023562 004567 160210 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4191 ;+
4192 ; Set up the bit map of lines to change and mask of bits to alter parameters.
4193 ;-
4194 023566 012703 000377 MOV #MAPLNS,R3 ;GET THE BIT MAP OF LINES TO CHANGE.
4195 023572 012704 177777 MOV #-1,R4 ;INDICATE ALL 16 BITS TO BE CHANGED.
4196 ;+
4197 ; Call the subroutine to write the specified registers.
4198 ;-
4199 023576 004767 174026 JSR PC,ALTFLD ;CHANGE THE REGISTERS.
4200
4201 023602 60$: PASS ;RESTORE GPRS.
023602 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4202 023604 000207 RTS PC

```

GLOBAL SUBROUTINE

- WTWLPR -

```

4204 .SBTTL GLOBAL SUBROUTINE - WTWLPR -
4205 ;* *****
4206 ;* - Line Parameter Register Setup Routine -
4207 ;* This subroutine is used to set the Device Under Test (DUT) Line
4208 ;* Parameter Registers (LPR) to the specified state. Only the LPRs for
4209 ;* the specified lines are altered.
4210 ;*
4211 ;* INPUTS: R0 New line parameters.
4212 ;* R5 - Bit map of lines to be altered.
4213 ;* CSRA - Contains address of the DUT CSR.
4214 ;* IESTAT - Contains the current state of the TX and RX interrupt
4215 ;* enable bits in the CSR.
4216 ;* LPRA - Contains address of the DUT LPR.
4217 ;*
4218 ;* OUTPUTS: LPR - Specified DUT Line Parameter Registers are altered.
4219 ;*
4220 ;* CALLING SEQUENCE: JSR PC,WTWLPR
4221 ;*
4222 ;* COMMENTS:
4223 ;*
4224 ;* SUBORDINATE ROUTINES CALLED: ALTFLD.
4225 ;* *****
4226
4227 023606 WTL PR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023606 004567 160164 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4228
4229 ; Set up the parameters for the call to ALTFLD.
4230 ;-
4231 023612 016701 156432 MOV LPRA,R1 ;SET UP THE REGISTER ADDRESS PARAMETER.
4232 023616 010002 MOV R0,R2 ;SET UP THE DESIRED REGISTER CONTENTS.
4233 023620 010503 MOV R5,R3 ;SET UP THE BIT MAP OF LINES TO ALTER.
4234 023622 012704 177777 MOV #1,R4 ;SELECT ALL REGISTER BITS TO BE ALTERED.
4235
4236 ;+
4237 ; Call the subroutine which alters the register contents.
4238 023626 004767 173776 ;-
JSR PC,ALTFLD ;ALTER THE REGISTER CONTENTS.
4239
4240 023632 60$: PASS ;RESTORE GPRS.
023632 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4241 023634 000207 RTS PC

```

INTERRUPT SERVICE ROUTINE - CACHRX -

```

4243 .SBTTL INTERRUPT SERVICE ROUTINE - CACHRX -
4244 ;* *****
4245 ;* - Catch Receiver interrupt.
4246 ;* This routine is used in several tests, to log a count of the
4247 ;* number of receiver interrupts that occur.
4248 ;*
4249 ;* INPUTS: CSRA - Contains the address of the CSR.
4250 ;* RXINTC - Holds the count of the number of RX interrupts
4251 ;* that occurred.
4252 ;*
4253 ;* OUTPUTS: RXINTC - Contains the updated interrupt count.
4254 ;*
4255 ;*
4256 ;* CALLING SEQUENCE: Put the address of the label CACHRX in the vector
4257 ;* location.
4258 ;*
4259 ;* COMMENTS:
4260 ;*
4261 ;* SUBORDINATE ROUTINES CALLED: none
4262 ;* - *****
4263
4264 023636 CACHRX::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023636 004567 160134 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4265 023642 016701 156466 MOV RXINTC,R1 ;GET THE RECEIVER INTERRUPT COUNT
4266 023646 005201 INC R1 ;INCREMENT THE COUNT
4267 023650 102001 BVC 2$ ;BRANCH IF NO OVERFLOW OCCURRED
4268 023652 005301 DEC R1 ;RESET THE COUNT TO 177777
4269 023654 010167 156454 2$: MOV R1,RXINTC ;SAVE NEW COUNT VALUE
4270 023660 60$: PASS ;RESTORE GPRS.
023660 004736 JSP PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4271 023662 000002 RTI

```

INTERRUPT SERVICE ROUTINE - CACHTX -

```

4273 .SBTTL INTERRUPT SERVICE ROUTINE - CACHTX -
4274 ;+ *****
4275 ;* - Catch Transmitter interrupt.
4276 ;* This routine is used in several tests, to log a count of the
4277 ;* number of transmission interrupts that occur.
4278 ;*
4279 ;* INPUTS: CSRA - Contains the address of the CSR.
4280 ;* TXINTC - Holds the count of the number of TX interrupts
4281 ;* that occurred.
4282 ;*
4283 ;* OUTPUTS: TXINTC - Contains the updated interrupt count.
4284 ;*
4285 ;*
4286 ;* CALLING SEQUENCE: Put the address of the label CACHTX in the vector
4287 ;* location.
4288 ;*
4289 ;* COMMENTS:
4290 ;*
4291 ;* SUBORDINATE ROUTINES CALLED: none
4292 ;-- *****
4293
4294 023664 CACHTX::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023664 004567 160106 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4295 023670 016701 156444 MOV TXINTC,R1 ;GET THE TRANSMISSION INTERRUPT COUNT
4296 023674 005201 INC R1 ;INCREMENT THE COUNT
4297 023676 102001 BVC 2$ ;BRANCH IF NO OVERFLOW OCCURRED
4298 023700 005301 DEC R1 ;RESET THE COUNT TO 177777
4299 023702 010167 156432 2$: MOV R1,TXINTC ;SAVE NEW COUNT VALUE
4300 023706 004736 60$: PASS ;RESTORE GPRS.
023706 004736 JSP PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4301 023710 000002 RTI

```

INTERRUPT SERVICE ROUTINE - CLKINT -

```

4303 .SBTTL INTERRUPT SERVICE ROUTINE - CLKINT -
4304 ;* *****
4305 ;* This routine is executed CLKHRZ times per second. It decrements the
4306 ;* two timer counters down to zero.
4307 ;*
4308 ;* INPUTS: TIMER1 - Timer counter #1.
4309 ;*          TIMER2 - Timer counter #2.
4310 ;*          TIMER3 - Timer counter for call of BREAK macro.
4311 ;*
4312 ;* OUTPUTS: The 2 timer counters are decremented if they are not zero.
4313 ;*
4314 ;* CALLING SEQUENCE: Put #CLKINT in the clock interrupt vector slot.
4315 ;*                  Put the desired time period (seconds times CLKHRZ) in
4316 ;*                  either TIMER1 or TIMER2 and poll the respective timer
4317 ;*                  counter to detect its going to 0 on time-out.
4318 ;*
4319 ;* COMMENTS: The 2 counters will not wraparound but will stop at 0. This
4320 ;*           allows the detection of a time-out any time after the time out
4321 ;*           has occurred until the timer counter is set to another value.
4322 ;*
4323 ;* SUBORDINATE ROUTINES CALLED: None.
4324 ;* *****
4325
4326 023712 005767 156446 CLKINT:: TST TIMER1 ;CHECK FOR TIMER1 AT ZERO.
4327 023716 001402 BEQ 2$ ;BRANCH TO LEAVE IT AT ZERO IF IT IS ZERO.
4328 023720 005367 156440 DEC TIMER1 ;DECREMENT TIME COUNT.
4329 023724 005767 156436 2$: TST TIMER2 ;CHECK FOR TIMER2 AT ZERO.
4330 023730 001402 BEQ 4$ ;BRANCH TO LEAVE IT ALONE IF IT'S ALREADY ZERO.
4331 023732 005367 156430 DEC TIMER2 ;DECREMENT TIME COUNT.
4332 023736 005367 156426 4$: DEC TIMER3 ;DECREMENT THE BREAK COUNT.
4333 023742 001006 BNE 60$ ;EXIT IF NOT TIME TO CALL BREAK.
4334 023744 016767 156422 156416 MOV BCOUNT,TIMER3 ;SET UP TIME TILL NEXT BREAK.
4335 023752 010046 MOV RO,-(SP) ;SAVE CONTENTS OF RO FROM BREAK MACRO.
4336 023754 BREAK ;CHECK FOR OPERATOR CONTROL/C. TRAP C$BRK
4337 023756 012600 MOV (SP)+,RO ;RESTORE CONTENTS OF RO.
4338 023760 000002 60$: RTI

```


INTERUPT SERVICE ROUTINE - RXBRRT -

```

4340 .SBTTL INTERUPT SERVICE ROUTINE RXBRRT -
4341 ;* *****
4342 ;* - BR Level Test Receive Interrupt Service Routine -
4343 ;* This service routine handles Receive Interrupts during the Interrupt
4344 ;* BR Level Test. This routine counts the interrupt and sets a flag
4345 ;* to indicate that the interrupt has occurred. It also checks the
4346 ;* flag which indicates that a TX interrupt has occurred. If the TX
4347 ;* interrupt flag is set, this routine sets an interrupt order error
4348 ;* flag indicating that a transmit interrupt was serviced before a
4349 ;* simultaneous receive interrupt.
4350 ;*
4351 ;* INPUTS: RXINTC - Holds the count of the number of RX interupts.
4352 ;* RXINTF - RX Interrupt flags.
4353 ;*
4354 ;* OUTPUTS: RXINTC - Contains the updated interupt count.
4355 ;* RXINTF - RX Int flags:
4356 ;* (Bit 0 set, bit 14 set if TXINTF bit 0 is set.)
4357 ;*
4358 ;* CALLING SEQUENCE: Put the address of the label RXBRRT in the vector
4359 ;* location.
4360 ;*
4361 ;* COMMENTS: NOTE: The FIFO is purged by this routine.
4362 ;*
4363 ;* SUBORDINATE ROUTINES CALLED: None.
4364 ;*
4365 ;* *****
4366 RXBRRT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023762 004567 160010 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
023762 017700 156254 MOV @RBUFA,R0 ;READ THE CHAR OUT OF THE FIFO.
4367 023766 017700 156254 MOV RXINTC,R1 ;GET THE INTERUPT COUNT.
4368 023772 016701 156336 MOV R1 ;INCREMENT THE COUNT.
4369 023776 005201 INC R1 ;BYPASS UPDATING COUNT IF OVERFLOW OCCURRED.
4370 024000 001402 BEQ 2$ ;SAVE NEW COUNT VALUE.
4371 024002 010167 156326 MOV R1,RXINTC ;GET THE RX INTERRUPT FLAGS.
4372 024006 016701 156324 2$: MOV RXINTF,R1 ;SET THE RX INTERRUPT HAS OCCURRED FLAG.
4373 024012 052701 000001 BIS #BIT0,R1 ;TEST THE "TX INT HAS OCCURRED" FLAG.
4374 024016 032767 000001 156316 BIT #BIT0,TXINTF ;SKIP SETTING ERROR FLAG IF NO TX INT.
4375 024024 001402 BEQ 4$ ;SET THE INTERRUPT ORDER ERROR FLAG.
4376 024026 052701 040000 BIS #BIT14,R1
4377 ;*
4378 ; 8 FIFO codes will cause 8 interrupts, after these 8 codes we don't want
4379 ; to check the interrupt order, because perhaps a BMP code has come in
4380 ; between the servicing of the 8 FIFO code interrupts and the servicing
4381 ; of one of the TX interrupts.
4382 ;*
4383 024032 026767 156276 153750 4$: CMP RXINTC,NUMLNS ;TEST FOR ALL SELFTEST CODE INTS DONE.
4384 024040 003002 BGT 60$ ;SKIP UPDATING RX INT FLAGS IF EXTRA RX INTS.
4385 024042 010167 156270 MOV R1,RXINTF ;UPDATE THE RX INTERRUPT FLAGS.
4386 024046 004736 60$: PASS ;RESTORE GPRS.
024046 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4387 024050 000002 RTI

```

INTERUPT SERVICE ROUTINE - RXINPT -

```

4389 .SBTTL INTERUPT SERVICE ROUTINE - RXINPT -
4390 ;* *****
4391 ;* - Receive Character Input Interrupt Service Routine -
4392 ;* This service routine inputs a character from the DUT and loads the
4393 ;* char (complete with status flags) into a receive char buffer in
4394 ;* memory. The interrupt is also counted. The receive char buffer is
4395 ;* monitored to ensure that it does not overflow.
4396 ;*
4397 ;* INPUTS: BUFEND - Labels the end of the host memory buffer.
4398 ;* BUFPTR - Contains address of next free buffer location.
4399 ;* CSRA - Contains the address of the DUT CSR.
4400 ;* RBUFA - Contains the address of the RBUF DUT register.
4401 ;* RXINTC - Holds the count of the number of RX interrupts.
4402 ;* RXINTF - RX Interrupt flags.
4403 ;*
4404 ;* OUTPUTS: BUFPTR - Contains updated address of next free buffer location.
4405 ;* RXINTC - Contains the updated interrupt count.
4406 ;* RXINTF - RX Int flags (bit 15 set if RX.DATA.AVAIL is clear).
4407 ;*
4408 ;* CALLING SEQUENCE: Put the address of the label RXINPT in the vector
4409 ;* location.
4410 ;*
4411 ;* COMMENTS: In case of overflow of the memory buffer, BUFPTR will be
4412 ;* maintained equal to BUFEND and the word at BURFPTR will be
4413 ;* the last word read from the DUT FIFO.
4414 ;* NOTE: This routine can destroy TX.ACTIONS by reading the CSR.
4415 ;*
4416 ;* SUBORDINATE ROUTINES CALLED: None.
4417 ;-- *****
4418
4419 024052 RXINPT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4420 024052 004567 157720 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4421 024056 032777 000200 156160 BIT #BIT7,@CSRA ;TEST RX.DATA.AVAIL BIT OF THE CSR (READS CSR).
4422 024064 001003 BNE 2$ ;BRANCH AROUND SETTING FLAG IF BIT IS SET.
4423 024066 052767 100000 156242 BIS #BIT15,RXINTF ;SET THE RX.DATA.AVAIL CLEAR FLAG.
4424 024074 016701 156234 2$: MOV RXINTC,R1 ;GET THE INTERRUPT COUNT.
4425 024100 005201 INC R1 ;INCREMENT THE COUNT.
4426 024102 001402 BEQ 4$ ;BYPASS UPDATING COUNT IF OVERFLOW OCCURRED.
4427 024104 010167 156224 MOV R1,RXINTC ;SAVE NEW COUNT VALUE.
4428 024110 016702 156210 4$: MOV BUFPTR,R2 ;GET THE POINTER TO NEXT FREE BUFFER WORD.
4429 024114 017722 156126 MOV @RBUFA,(R2)+ ;READ A CHAR FROM THE FIFO INTO BUFFER.
4430 024120 020267 157602 CMP R2,BUFEND ;TEST FOR POINTER BEYOND END OF BUFFER.
4431 024124 103002 BHIS 60$ ;SKIP THE PTR UPDATE IF PTR OUT OF BOUNDS.
4432 024126 010267 156172 MOV R2,BUFPTR ;UPDATE THE BUFFER POINTER.
4433 024132 004736 60$: PASS ;RESTORE GPRS.
024132 000002 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
RTI

```

GLOBAL TRAP SERVICE ROUTINE - TP4RTN -

```

4435 .SBTTL GLOBAL TRAP SERVICE ROUTINE - TP4RTN -
4436 ;*****
4437 ;* Bus Time out Trap (004 trap) Service Routine -
4438 ;* This routine is used during the Device Register Address Access Test.
4439 ;* It determines if the 004 trap was caused by an "expected" error or
4440 ;* not by examining the return PC value on the stack. If the trap is
4441 ;* unexpected, this routine jumps to the normal Diagnostic Supervisor
4442 ;* 004 trap handling routine.
4443 ;*
4444 ;* INPUTS: SP - Points to the PC where the trap occurred.
4445 ;* ADRPTR - Label at the address where "expected" traps occur.
4446 ;* TP4FLG - 004 trap flags.
4447 ;*
4448 ;* OUTPUTS: TP4FLG - Bit 15 is set if "expected" trap occurred.
4449 ;*
4450 ;* CALLING SEQUENCE: Put address pointed to by TP4RTN in 004 vector.
4451 ;* Occurrence of 004 trap vectors to this routine.
4452 ;*
4453 ;* COMMENTS: Any 004 trap which occurs at an address other than that labeled
4454 ;* ADRPTR will be handled by the normal 004 trap service routine.
4455 ;*
4456 ;* SUBORDINATE ROUTINES CALLED: None.
4457 ;*****
4458
4459 024136 021627 020140 TP4RTN:: CMP (SP),#ADRPTR ;COMPARE EXPECTED ADR AGAINST TRAP RET PC.
4460 024142 001402 BEQ 2$ ;IF THEY MATCH, CONTINUE THIS ROUTINE.
4461 024144 000177 156174 JMP @TP4VEC ;IF NOT,JUMP TO NORMAL 004 TRAP SERVICE RTN.
4462 024150 052767 100000 156170 2$: BIS #BIT15,TP4FLG ;SET THE 004 TRAP OCCURED FLAG.
4463 024156 000002 RTI ;ALL DONE, GO BACK TO THE TEST.

```

INTERUPT SERVICE ROUTINE - TXINTR -

```

4465 .SBTTL INTERUPT SERVICE ROUTINE - TXINTR -
4466 ;* *****
4467 ;* - Transmit Interrupt Service Routine -
4468 ;* This routine handles a transmit interrupt from the Device Under Test
4469 ;* (DUT) by counting the interrupt and reading the DUT CSR to clear the
4470 ;* interrupt request. This routine also sets a flag to indicate that
4471 ;* a TX interrupt has occurred and sets a flag if the TX.ACTION bit is
4472 ;* not set in the read contents of the DUT CSR.
4473 ;*
4474 ;* INPUTS: CSRA - Contains the address of the CSR.
4475 ;* TXINTC - Holds the count of the number of TX interrupts.
4476 ;* TXINTF - TX Interrupt flags.
4477 ;*
4478 ;* OUTPUTS: TXINTC - Contains the updated TX interrupt count.
4479 ;* TXINTF - TX Int flags (bit 0 set, bit 15 set if TX.ACTION clr).
4480 ;*
4481 ;* CALLING SEQUENCE: Put the address of the label TXINTR in the vector
4482 ;* location.
4483 ;*
4484 ;* COMMENTS:
4485 ;*
4486 ;* SUBORDINATE ROUTINES CALLED: none
4487 ;*-- *****
4488
4489 024160 TXINTR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
024160 004567 157612 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4490 024164 016701 156150 MOV TXINTC,R1 ;GET THE TX INTERRUPT COUNT.
4491 024170 005201 INC R1 ;INCREMENT THE COUNT.
4492 024172 102001 BVC 2$ ;BRANCH IF NO OVERFLOW OCCURRED.
4493 024174 005301 DEC R1 ;RESET THE COUNT TO 177777.
4494 024176 010167 156136 2$: MOV R1,TXINTC ;SAVE NEW COUNT VALUE.
4495 024202 016703 156134 MOV TXINTF,R3 ;GET THE TX INTERRUPT FLAGS.
4496 024206 017702 156032 MOV @CSRA,R2 ;READ THE CSR.
4497 024212 100402 BMI 4$ ;SKIP SETTING OF FLAG IF TX.ACTION IS SET.
4498 024214 052703 100000 BIS #BIT15,R3 ;SET THE TX.ACTION CLEAR FLAG.
4499 024220 052703 000001 4$: BIS #BIT0,R3 ;SET THE TX INT HAS OCCURRED FLAG.
4500 024224 010367 156112 MOV R3,TXINTF ;UPDATE THE TX INTERRUPT FLAGS.
4501 024230 60$: PASS ;RESTORE GPRS.
024230 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4502 024232 000002 RTI

```

INTERUPT SERVICE ROUTINE - TXINTR -

```

4504
4505 ;*****
** 4506 ;
4507 ; VDHA.RPT
4508 ;
4509 ;*****
4510
4511
4512
4513 .SBTTL REPORT CODING SECTION
4514
4515 ;**
4516 ; THE REPORT CODING SECTION CONTAINS THE
4517 ; "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
4518 ;--
4519
4520 024234 BGNRPT
      024234 L$RPT.:
4521
4522 024234 EXIT RPT
      024234 000167 .WORD J$JMP
      024236 000000 .WORD L10017-2-.
4523
4524 .EVEN
4525
4526 024240 ENDRPT
      024240 L10017:
      024240 104425 TRAP C$RPT

```

PROTECTION TABLE

4528
4529
4530

.SBTTL PROTECTION TABLE

4531
4532
4533

;
;
;

FVTSKL4.P11

4534
4535
4536
4537
4538

;*

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

;

4539
4540
4541
4542

4543 024242
024242

BGNPROT

L\$PROT::

4544
4545 024242 177777
4546 024244 177777
4547 024246 177777

-1 ;OFFSET INTO P-TABLE FOR CSR ADDRESS
-1 ;OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
-1 ;OFFSET INTO P-TABLE FOR DRIVE NUMBER

4548
4549 024250
4550

ENDPROT

PROTECTION TABLE

```

4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586 024250
      024250
4587
4588 024250
      024250 012700 000040
      024254 104447
4589 024256
      024256 103416
4590
4591 024260
      024260 012700 000037
      024264 104447
4592 024266
      024266 103556
4593
4594 024270
      024270 012700 000035
      024274 104447
4595 024276
      024276 103555
4596
4597 024300
      024300 012700 000036
      024304 104447
4598 024306
      024306 103161
4599 024310 000167 000552
4600 024314
4601 024314
      024314 104433
4602
4603
4604
4605 024316
      024316 012700 000114
      024322 104462

```

```

*****
:
:          VDHA.IN_
:
*****

.SBTTL INITIALIZE SECTION
:**
*****
:*      This section contains the code which is performed at the beginning of
:*      each pass or after a continue command.
:*      This code performs the following actions:
:*
:*      Moves the information held in the hardware P-table into the global
:*      data area.
*****
:
:          BGNINIT
:
:          L$INIT::
:SEE IF PROGRAM JUST STARTED, BR IF YES
:          READEF #EF.START
:
:          BCOMPLETE      NEWSTA
:
:          BCS      NEWSTA
:SEE IF PROGRAM JUST RESTARTED, BR IF YES
:          READEF #EF.RESTART
:
:          BCOMPLETE      NEWRES
:
:          BCS      NEWRES
:SEE IF THIS IS A NEW PASS, BR IF YES
:          READEF #EF.NEW
:
:          BCOMPLETE      NEWPAS
:
:          BCS      NEWPAS
:SEE IF PROGRAM WAS JUST CONTINUED
:          READEF #EF.CONTINUE
:
:          BNCOMPLETE     GETPRM
:          JMP      ENDIT
:          BCC      GETPRM
NEWSTA:
:          BRESET
:          ;RESET THE BUS TO PREVENT ILLEGAL INTERRUPTS.
:          TRAP      C$RESET
:**
:          Set up for Line Time Clock interrupts.
:
:          CLOCK      L,R1
:          ;GET THE CLOCK PARAMETERS.
:
:          MOV      #L,RO
:          TRAP      C$CLCK

```

INITIALIZE SECTION

```

024324 010001
4606 024326 012167 156022      MOV      (R1)+,CLKCSR      ;STORE CLOCK CSR ADDRESS.
4607 024332 012167 156020      MOV      (R1)+,CLKBRL     ;STORE CLOCK BUS REQ INT LEVEL.
4608 024336 012167 156016      MOV      (R1)+,CLKVEC     ;STORE CLOCK INTERRUPT VECTOR.
4609 024342 012167 156014      MOV      (R1)+,CLKHRZ     ;STORE CLOCK FREQUENCY.
4610 024346 026727 156010 000062  CMP      CLKHRZ,#50.      ;TEST FOR 50HZ LINE FREQUENCY.
4611 024354 001004              BNE      2$               ;BRANCH IF CLOCK IS NOT 50HZ.
4612 024356 012767 000024 156010  MOV      #20.,MSTICK      ;INDICATE 20MS PER CLOCK TICK.
4613 024364 000403              BR       4$
4614 024366 012767 000021 156000 2$:  MOV      #17.,MSTICK      ;INDICATE 17 MS PER CLOCK TICK.
4615 024374 012767 000021 156000 4$:  SETVEC  CLKVEC,#CLKINT,#PRI06 ;INITIALIZE CLOCK INTERRUPT VECTOR.
                                MOV      #PRI06,-(SP)
                                MOV      #CLKINT,-(SP)
                                MOV      CLKVEC,-(SP)
                                MOV      #3,(SP)
                                TRAP    C$SVEC
                                ADD     #10,SP
4616 024422 016700 155734      MOV      CLKHRZ,R0        ;INITIALIZE THE BREAK COUNT
4617 024426 006200              ASR      R0                ; TO CAUSE A BREAK
4618 024430 010067 155736      MOV      R0,BCOUNT        ; EVERY 1/2 SECOND.
4619 024434 012700 000240      SETPRI  #PRI05            ;ALLOW CLOCK INTERRUPTS DISABLE OTHERS.
                                MOV      #PRI05,R0
                                TRAP    C$SPRI
024434 012700 000240
024440 104441
4620
4621 ;+
4622 ; Enable the Line Time Clock (LTC) checking to make sure that the CSR
4623 ; is accessible.
4624 ; First set up to catch any 004 traps which occur:
4625 024442 016767 153336 155674  MOV      4,TP4VEC         ;SAVE THE EXISTING 004 TRAP VECTOR.
4626 024450 012767 024136 153326  MOV      #TP4RTN,4       ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4627
4628 ;+
4629 ; Enable LTC checking for 004 trap in case CSR is not there.
4630 024456 005067 155664              CLR      TP4FLG           ;CLEAR THE 004 TRAP FLAG.
4631 024462 012767 000100 155660  MOV      #BIT6,WORD1      ;SET UP TO SET BIT6 OF THE LTC CSR.
4632 024470 012700 002350              MOV      #WORD1,R0        ;SET UP WORD1 AS THE CKTRAP MOVE SOURCE.
4633 024474 016701 155654              MOV      CLKCSR,R1        ;SET UP LTC CSR AS DESTINATION FOR CKTRAP MOVE.
4634 024500 004767 173422              JSR      PC,CKTRAP        ;MOVE AND CHECK FOR TRAP.
4635 024504 016767 155634 153272  MOV      TP4VEC,4         ;RESTORE THE NORMAL 004 TRAP VECTOR.
4636 024512 103403              BCS     6$                ;IF NO TRAP, LTC IS THERE SO CONTINUE.
4637 024514 005067 155642              CLR      CLKHRZ          ;CLEAR LTC FREQUENCY WORD TO INDICATE NO LTC.
4638 024520 000402              BR      8$                ;BYPASS THE FOLLOWING CALIBRATION PROCEDURES.
4639
4640 ;+
4641 ; Calibrate the DELAY routine milli-second delay count value.
4642 024522 004767 173154 6$:  JSR      PC,CALMSL
4643
4644 ;+
4645 ; Check for Memory Management present on this machine.
4646 ; If MEM MGT is present, disable it.
4647 024526 016767 153252 155610 8$:  MOV      4,TP4VEC         ;SAVE THE EXISTING 004 TRAP VECTOR.
4648 024534 012767 024136 153242  MOV      #TP4RTN,4       ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4649 024542 005067 155600              CLR      TP4FLG           ;CLEAR THE 004 TRAP FLAG.
4650 024546 005067 155576              CLR      WORD1            ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4651 024552 012700 002350              MOV      #WORD1,R0        ;SELECT CLEARED WORD AS CKTRAP RTN SOURCE.
4652 024556 016701 155616              MOV      MMSRO,R1        ;SELECT MEM MGT SRO REGISTER AS DESTINATION.
4653 024562 005067 155614              CLR      MMPRES           ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN T.

```


INITIALIZE SECTION

```

4654 024566 005067 155612          CLR    MMENAB          ;INDICATE MEM MGT IS NOT ENABLED.
4655 024572 004767 173330          JSR    PC,CKTRAP      ;CLEAR THE MEM MGT SRO REG AND CHECK FOR TRAP.
4656 024576 016767 155542 153200    MOV    TP4VEC,4       ;RESTORE THE NORMAL 004 TRAP VECTOR.
4657 024604 103003                    BCC    10$            ;SKIP INDICATING MEM MGT PRESENT IF IT ISN'T.
4658 024606 012767 000001 155566    MOV    #1,MMPRES      ;INDICATE THAT MEM MGT IS PRESENT.
4659 024614 005067 155512 10$:    CLR    PASCNT         ;CLR COUNTER USED IN REPORTING ROM VERSION #.
4660 024620 000167 000006          JMP    NEWPAS         ;SKIP AROUND THE BUS RESET, IT'S BEEN DONE.
4661
4662 024624                    NEWRES: BRESET        ;RESET THE BUS TO PREVENT ILLEGAL INTERRUPTS.
      024620 104433                    TRAP C$RESET
4663 024626 005067 155500          CLR    PASCNT         ;CLR COUNTER USED IN REPORTING ROM VERSION #.
4664 024632                    NEWPAS:
4665 024632 012767 177777 155400    MOV    #-1,UNITN     ;RESET LOGICAL DEVICE TO 1
4666
4667 ;*
4668 ; Increment the pass counter, correct for any overflow.
4669 ; This counter is used in the Rom version test.
      ;
4670 024640 005267 155466          INC    PASCNT         ;INCREMENT THE PASS COUNTER.
4671 024644 001002                    BNE    GETPRM        ;BRANCH IF WE HAVE NOT YET! OVERFLOWED.
4672 024646 005367 155460          DEC    PASCNT         ;SET PASS COUNT TO 177777 OCTAL.
4673
4674 ; GET THE HARDWARE PARAMETERS FOR THIS UNIT.
      GETPRM:
4675 024652                    INC    UNITN          ;INCREMENT LOGICAL DEVICE NUMBER
4676 024652 005267 155362          CMP    UNITN,L$UNIT  ;SEE IF MAXIMUM UNIT NO. EXCEEDED
4677 024656 026767 155356 155126    BGE    NEWPAS         ;BR IF YES
4678 024664 002362                    GPHARD UNITN,R1      ;GET P-TABLE POINTER INTO R1
4679
4680 024666                    MOV    TRAP          UNITN,RO
      024666 016700 155346                    TRAP C$GPHRD
      024672 104442                    MOV    RO,R1
      024674 010001
4681 024676                    BCOMPLETE          30$ ;BR IF DEVICE AVAILABLE
      024676 103401
4682 024700 000764                    BR    GETPRM         ;SKIP THIS DEVICE
4683
4684
4685 ;***** HARDWARE PARAMETER MOVING CODE *****
4686 024702 012167 155336 30$:    MOV    (R1)+,CSRA     ;STORE DHV11-M CSR ADDRESS IN DEV.REG.ADDRESS TABLE
4687 024706 012102                    MOV    (R1)+,R2      ;GET THE RX INTERRUPT VECTOR ADDRESS.
4688 024710 010267 155316                    MOV    R2,RXVECA     ;STORE RX INT VECTOR ADDRESS.
4689 024714 062702 000004                    ADD    #4,R2         ;CALCULATE TX INTERRUPT VECTOR ADDRESS.
4690 024720 010267 155310                    MOV    R2,TXVECA     ;STORE TX INT VECTOR ADDRESS.
4691 024724 012167 155306                    MOV    (F1)+,ACTLNS  ;STORE DHV11-M ACTIVE LINE BIT MAP
4692 024730 012702 000377                    MOV    #MAPLNS,R2    ;GET THE BIT MAP FOR ALL LINES.
4693 024734 005102                    COM    R2            ;GET A BIT MAP OF NON-EXISTANT LINES.
4694 024736 040267 155274                    BIC    R2,ACTLNS     ;CLEAR NON-EXISTANT LINES FROM ACTLNS.
4695 024742 112167 155274                    MOVB   (R1)+,BRLEVL  ;STORE DHV11-M INTERUPT BUS REQUEST LEVEL
4696
4697 ;*
4698 ; CALCULTE DEVICE REGISTER ADDRESSES,AND PUT THEM IN THE
4699 ; DEVICE REGISTER ADDRESS TABLE.
      ;
4700 024746 016701 155272          MOV    CSRA,R1       ;COPY CSR ADDRESS
4701 024752 005201                    INC    R1            ;INCREMENT CSR ADDRESS
4702 024754 005201                    INC    R1            ; COPY BY 2.
4703 024756 012703 000007          MOV    #7,R3         ;SET UP REGISTER COUNT
4704 024762 012702 002246          MOV    #RBUFA,R2     ;GET LOCATION WHERE RBUF ADDRESS GOES IN TABLE
4705 024766 010122 12$:    MOV    R1,(R2)+     ;STORE REGISTER ADDRESS IN TABLE

```

INITIALIZE SECTION

```

4706 024770 005201      INC      R1      ;INCREMENT REGISTER ADDRESS
4707 024772 005201      INC      R1      ; BY 2, FOR THE NEXT DEVICE REGISTER.
4708 024774 005303      DEC      R3      ;DECREMENT REGISTER COUNT
4709 024776 001373      BNE      12$    ;LOOP IF NOT DONE
4710
4711
4712      ;+
4713      ; Initialise the BMP code queue.
4714 025000 012700 002526      MOV      #BMPCQB,R0      ;GET THE START ADDRESS OF THE QUEUE.
4715 025004 012701 002726      MOV      #BMPCQE,R1      ;GET THE END ADDRESS OF THE QUEUE.
4716 025010 010067 155510      MOV      R0,BMPCQP      ;SET THE POINTER TO THE START OF THE QUEUE.
4717 025014 005020 14$:      CLR      (R0)+          ;CLEAR OUT THE CONTENTS OF THE QUEUE.
4718 025016 020001          CMP      R0,R1          ;CHECK IF END OF QUEUE HAS BEEN REACHED.
4719 025020 103775          BLO      14$           ;LOOP IF NOT ALL DONE.
4720
4721      ;+
4722      ; Report the Unit number if the software P-table question was answered YES,
4723      ; and the maximum unit number is greater than 1.
4724 025022 032767 000020 155176      BIT      #BIT4,OPTION    ;CHECK IF THE QUESTION WAS ANSWERED YES.
4725 025030 001416          BEQ      16$           ;SKIP REPORTING UNIT NUMBER IF IT IS DISABLED.
4726 025032 026727 154754 000001      CMP      L$UNIT,#1      ;CHECK MAXIMUM NUMBER OF UNITS SELECTED.
4727 025040 003412          BLE      16$           ;DO NOT REPORT UNIT NUMBER IF MAX NUMBER < 1.
4728 025042          PRINTF  #MFUNIT,UNITN ;REPORT UNIT NUMBER.
4729 025042 016746 155172          MOV      UNITN,-(SP)
4730 025046 012746 004076          MOV      #MFUNIT,-(SP)
4731 025052 012746 000002          MOV      #2,-(SP)
4732 025056 010600          MOV      SP,R0
4733 025060 104417          TRAP    C$PNTF
4734 025062 062706 000006          ADD      #6,SP
4735 025066 16$:
4736
4737
4738
4739 025066 005067 155260      ENDIT: CLR      CTRLCF      ;CLR THE CTRL-C TEST ABORT FLAG.
4740
4741      ;+
4742      ; Set the processor priority to allow LTC interrupts but not others.
4743      ;-
4744 025072          SETPRI  #PRI05          ;SET PROCESSOR PRIORITY TO 5.
4745 025072 012700 000240          MOV      #PRI05,R0
4746 025076 104441          TRAP    C$SPRI
4747
4748      ;+
4749      ; Enable Line Time Clock if one is available.
4750      ;-
4751 025100 005767 155256      TST      CLKHRZ          ;CHECK FOR A LTC BEING PRESENT.
4752 025104 001403          BEQ      18$           ;LTC PRESENT? NO, SKIP LTC ENABLE.
4753 025106 012777 000100 155240      MOV      #BIT6,@CLKCSR  ;YES, ENABLE THE LTC.
4754 025114 18$:
4755
4756      ENDINIT
4757
4758      L10021:
4759          TRAP    C$INIT
4760
4761      TNUM == 0
4762          ;INITIALIZE THE ASSEMBLER TEST NUMBER VARIABLE.

```

INITIALIZE SECTION

```

4749 ;*****
4750 ;
4751 ;           VDHA.ATD
4752 ;
4753 ;*****
4754
4755
4756
4757 .SBTTL  AUTODROP SECTION
4758
4759
4760 ;**
4761 ; THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
4762 ; THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
4763 ; SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
4764 ; DROPPED FROM TESTING.
4765 ;--
4766
4767 025116          BGNAUTO
4768 025116
4769
4770
4771
4772
4773
4774
4775
4776 025116          ENDAUTO
4777 025116
4778 025116 104461
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799

```

L\$AUTO::

L1002: TRAP C\$AUTO

AUTODROP SECTION

```

4778
4779
4780 ;*****
4781 ;
4782 ;
4783 ;*****
4784
4785
4786
4787 .SBTTL CLEANUP CODING SECTION
4788
4789 ;**
4790 ; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
4791 ; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
4792 ;--
4793
4794 025120 BGNCLN
4795 025120 L$CLEAN::
4804 025120 005767 155226 TST CTRLCF ;DID WE GET HERE BY CTRL-C FROM TEST?
4805 025124 001404 BEQ 2$ ;CTRL-C FROM TEST? NO, SKIP BUS RESET.
4806 025126 104433 BRESET ;YES, CLR ANY DMAS OR OUTSTANDING INTERRUPTS.
4807 025130 SETPRI #PRI05 ;DISABLE DEVICE INTERRUPTS, ALLOW LTC INTS.
4808 025136 012700 000240 TRAP C$RESET
4809 025136 104441 2$: MOV #PRI05,RO
4810 025136 0C5767 155220 BEQ 3$ TRAP C$SPRI
4811 025142 001402 CLR @CLKCSR
4812 025150 3$: EXIT CLN
4813 025150 104432 TRAP C$EXIT
4814 025152 000002 .WORD L10023-.
4826
4827 .EVEN
4828
4829 025154 ENDCLN
025154 L10023: TRAP C$CLEAN
025154 104412

```

CLEANUP CODING SECTION

```

4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847 025156
      025156
4848
4849
4850
4851
4852
4853
4854
4855
4856 025156
      025156 010046
      025160 012746 025202
      025164 012746 000002
      025170 010600
      025172 104417
      025174 062706 000006
4857 025200 000427
4858
4859 025202 045 101 040 DROP: .ASCIZ/*A UNIT*D6*A DROPPED FROM FURTHER TESTING.*N/
      025205 125 116 111
      025210 124 045 104
      025213 066 045 101
      025216 040 104 122
      025221 117 120 120
      025224 105 104 040
      025227 106 122 117
      025232 115 040 106
      025235 125 122 124
      025240 110 105 122
      025243 040 124 105
      025246 123 124 111
      025251 116 107 056
      025254 045 116 000
4860
4861 025260
4862 025260
      025260 000167
      025262 000000
4863
4864

```

```

;*****
;
;          VDHA.DRP
;*****
.SBTTL DROP UNIT SECTION
;+
; THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
; TO NO LONGER BE TESTED.
;-
      BGNU
                                L$DU:
;*****
;          INSERT DROP CODE HERE. THIS CODE WILL BE EXECUTED AFTER
;          A "DROP" COMMAND OR A "DODU" MACRO EXECUTION. THE PURPOSE
;          OF THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A
;          UNIT HAS BEEN DROPPED. THIS SECTION IS OPTIONAL.
;*****
      PRINTF #DROP,RO           ;REPORT UNIT THAT HAS BEEN DROPPED.
                                MOV      RO,-(SP)
                                MOV      #DROP,-(SP)
                                MOV      #2,-(SP)
                                MOV      SP,RO
                                TRAP    C$PNTF
                                ADD     #6,SP
      BR      EDROP             ;BRANCH AROUND THE MESSAGE.
DROP: .ASCIZ/*A UNIT*D6*A DROPPED FROM FURTHER TESTING.*N/
      .EVEN
EDROP: EXIT DU
                                .WORD   J$JMP
                                .WORD   L10024-2

```

DROP UNIT SECTION

4865
4866 025264
025264
025264 104453

ENDDU

L10024: TRAP C\$DU

DROP UNIT SECTION

```

4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885 025266
      025266
4886
4887
4888
4889
4890
4891
4892
4893
4894 025266
      025266 000167
      025270 000000
4895
4896
4897
4898
4899 025272
      025272
      025272 104452
4900

```

```

;*****
;
;           VDHA:ADD
;
;*****
.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.
;--
      BGNAU
                                     L$AU::
;*****
;
;   INSERT ADD CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
;   AN "ADD" COMMAND.  THE PURPOSE OF THIS CODE IS TO DO ANY
;   HOUSEKEEPING THAT MAY BE NECESSARY AFTER A UNIT HAS BEEN ADDED.
;   THIS SECTION IS OPTIONAL.
;*****
      EXIT  AU
                                     .WORD  J$JMP
                                     .WORD  L10025-2-.
      .EVEN
      ENDAU
                                     L10025:
                                     TRAP   C$AU

```

HARDWARE TEST

- ADRA -

```

4902 .SBTTL HARDWARE TEST - ADRA -
4903 ;**
4904 ;*****
4905 ;*
4906 ;* - REGISTER ADDRESS TEST -
4907 ;*
4908 ;* This test verifies that the Q-bus can read and write to the DHV11
4909 ;* device registers. If the DHV11 does not respond to the access
4910 ;* attempts (If the DHV11 is at the wrong address, for example) the
4911 ;* 004 bus time-out trap is detected by this routine and an error
4912 ;* is reported.
4913 ;*
4914 ;*****
4915 ;
4916 025274 BGNTST
025274
4917 000001 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER. T1::
4918 025274 012767 000001 155024 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (1)
4919 025302 012767 177777 155042 MOV #1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
4920 ;*
4921 ; Set up to catch any 004 traps which occur:
4922 ;-
4923 025310 016767 152470 155026 MOV 4,TP4VEC ;SAVE THE EXISTING 004 TRAP VECTOR.
4924 025316 012767 024136 152460 MOV #TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4925 025324 005005 CLR R5 ;CLEAR THE ERROR FLAGS.
4926 ;*
4927 ;
4928 ; Set up for the initial iteration of the test loop:
4929 ;-
4930 025326 005004 CLR R4 ;CLEAR THE LINE COUNTER.
4931 ;*
4932 ;
4933 ; Here begins the loop to test the registers for a line.
4934 ; First test the CSR and set the IND.ADR.REG (I.A.R) field.
4935 ;
4936 025330 005067 155012 2$: CLR TP4FLG ;CLEAR THE 004 TRAP FLAG.
4937 025334 016700 154704 MOV CSRA,R0 ;SET UP CSR AS THE CKTRAP MOVE SOURCE.
4938 025340 012701 025554 MOV #52$,R1 ;SET UP DESTINATION LOCATION FOR CKTRAP MOVE.
4939 025344 004767 172556 JSR PC,CKTRAP ;MOVE AND CHECK FOR TRAP.
4940 025350 103402 BCS 4$ ;IF NO TRAP, BYPASS ERROR.
4941 025352 052705 100001 BIS #100001,R5 ;SET FATAL READ ERROR FLAGS.
4942 025356 042767 000017 000170 4$: BIC #17,52$ ;CLEAR THE I.A.R FIELD OF THE CSR DATA.
4943 025364 050467 000164 BIS R4,52$ ;OR IN THE LINE COUNTER TO THE I.A.R FIELD.
4944 025370 010100 MOV R1,R0 ;USE OLD DESTINATION FOR SOURCE OF CKTRAP MOVE.
4945 025372 016701 154646 MOV CSRA,R1 ;SET UP CSR AS THE CKTRAP MOVE DESTINATION.
4946 025376 004767 172524 JSR PC,CKTRAP ;MOVE AND CHECK FOR TRAP.
4947 025402 103403 BCS 6$ ;IF NO TRAP, BYPASS ERROR.
4948 025404 052705 100002 BIS #100002,R5 ;SET FATAL WRITE ERROR FLAGS.
4949 025410 000440 BR 40$ ;EXIT AND REPORT FATAL ERROR.
4950 ;*
4951 ;
4952 ; Now, we test each register for this line.
4953 025412 012702 000010 6$: MOV #10,R2 ;INIT REGISTER COUNTER TO 8.
4954 025416 016767 154622 000126 MOV CSRA,50$ ;INITIALIZE THE REGISTER POINTER.
4955 025424 012700 025552 8$: MOV #50$,R0 ;SET UP REGISTER AS THE SOURCE FOR CKTRAP MOVE.
4956 025430 012701 025554 MOV #52$,R1 ;SET UP LOCAL STORAGE AS THE DES FOR CKTRAP.
4957 025434 004767 172466 JSR PC,CKTRAP ;PERFORM THE MOVE, CHECK FOR TRAP.

```


HARDWARE TEST

- ADRA -

```

4958 025440 103402          BCS 10$          ;IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.
4959 025442 052705 100001  BCS #100001,R5 ;SET FATAL READ ERROR FLAGS.
4960 025446 010100          MOV R1,R0       ;USE OLD DEST AS SRC FOR CKTRAP MOVE.
4961 025450 012701 025552  MOV #50$,R1     ;SET UP REGISTER AS THE DEST FOR CKTRAP MOVE.
4962 025454 004767 172446  JSR PC,CKTRAP   ;PERFORM THE MOVE, CHECK FOR TRAP.
4963 025460 103402          BCS 12$          ;IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.
4964 025462 052705 100002  BCS #100002,R5 ;SET FATAL WRITE ERROR FLAGS.
4965 025466 005267 000060  INC 50$         ;INCREMENT THE REGISTER
4966 025472 005267 000054  INC 50$         ; POINTER BY 2.
4967 025476 005302          DEC R2          ;COUNT THE REGISTER.
4968 025500 001351          BNE 8$          ;LOOP TO TEST THE NEXT REGISTER ADDRESS.
4969
4970
4971
4972
4973 025502 005204          ;+
4974 025504 020427 000010  INC R4          ; Now we set up to test the next line, or to exit if we are done.
4975 025510 002707          ;-
4976
4977
4978
4979
4980
4981 025512 016767 154626 152264 40$: MOV TP4VEC,4    ;INCREMENT THE LINE COUNTER.
4982 025520 005705          TST R5          ;COMPARE LINE COUNTER AGAINST NUMBER OF LINES.
4983 025522 100015          BPL 60$         ;LOOP TO TEST THE NEXT LINE IF WE'RE NOT DONE.
4984
4985 025524          ; Done checking device register addresses.
025524 104455          ; Report any errors and exit.
025526 000145          ;-
025530 006067          ;+
025532 016070          ; Done checking device register addresses.
025534          ; Report any errors and exit.
4986
4987 025534          ;-
025534 016700 154500 40$: MOV TP4VEC,4    ;RESTORE THE NORMAL 004 TRAP VECTOR.
025540 104451          TST R5          ;CHECK THE ERROR FLAGS.
4988 025542 005067 154604  BPL 60$         ;EXIT ROUTINE IF NO ERRORS.
4989 025546          ; REPORT "DEVICE REGISTER ACCESS ERRORS"
025546 104444          ERRDF 101,EM0103,ER0101; >>>> ERROR #101 <<<<<.
4990 025550 000402          TRAP C$ERDF
4991
4992
4993
4994 025552 000000          .WORD 0
4995 025554 000000          .WORD 0
4996 025556 005067 154570  CLR CTRLCF     ;INDICATE NO CTRL-C ABORT FROM TEST.
4997 025562          DOCLN
025562 104401          BR 60$         ;ABORT THIS SUB PASS.
                                TRAP C$DCLN
                                ;
                                ;+
                                ; Local storage.
                                ;-
025562 104401          50$: .WORD 0          ;STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.
                                52$: .WORD 0          ;STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.
                                60$: CLR CTRLCF     ;INDICATE THAT WE ARE NOT WITHIN A TEST.
                                ENDTST
                                L10026:
                                TRAP C$ETST

```

HARDWARE TEST

- MRSTA

```

4999 .SBTTL HARDWARE TEST - MRSTA -
5000 ;* *****
5001 ;* - Master Reset With Selftest Test -
5002 ;* This test verifies that the Master Reset bit will clear after a device
5003 ;* reset and the performance of the DUT ROM based selftest.
5004 ;*
5005 ;*
5006 025564 ;-- *****
    025564 BGNST
5007
5008 025564 000002 TNUM := TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5009 025572 012767 000002 154534 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (2)
5010 025600 012767 177777 154552 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
5011 025600 012767 000001 156160 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5012 025606 012767 006125 156156 MOV #EM0201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5013 025614 012767 016406 156152 MOV #ERO201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5014 ;+
5015 ; Wait up to 5 seconds for the DUT Master Reset bit to clear.
5016 025622 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5017 025626 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5018 025632 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5019 025634 016704 154404 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5020 025640 004767 172474 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5021 025644 103410 BCS 2$ ;SKIP TO RESET DUT IF MR CLEAR.
5022 ;+
5023 ; DUT Master Reset bit did not go clear. Device may be stuck in some
5024 ; odd state. Try to reset device with a bus reset.
5025 ;-
5026 025646 BRESET ;NO, TRY TO JOG DEVICE WITH BUS RESET.
    025646 104433 TRAP C$RESET
5027 025650 004767 174462 JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST.
5028 025654 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS
5029 025660 004767 172454 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5030 025664 103016 BCC 4$ ;GO REPORT ERROR IF MR BIT DID NOT CLEAR.
5031 ;+
5032 ; Set the Master Reset bit and verify that it clears within the proper time.
5033 ;-
5034 025666 012701 011610 2$: MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS
5035 025672 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5036 025674 004767 172440 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5037 025700 103010 BCC 4$ ;GO REPORT ERROR IF MR BIT DID NOT CLEAR.
5038 025702 012702 011610 MOV #5000.,R2
5039 025706 160102 SUB R1,R2 ;CALCULATE # OF MS FOR MR TO CLEAR.
5040 025710 001413 BEQ 6$ ;GO REPORT ERROR IF MR CLEAR IMMEDIATELY.
5041 025712 020227 000764 CMP R2,#500.
5042 025716 002417 BLT 8$ ;GO REPORT ERROR IF MR CLEAR IN < 1/2 SECOND.
5043 025720 000424 BR 60$ ;EXIT THE TEST WITHOUT ERROR.
5044 ;+
5045 ; Error reports:
5046 ;-
5047 ;Report MR bit would not clear after a DUT reset.
5048 025722 012767 000311 156040 4$: MOV #201.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5049 025730 012701 006173 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5050 025734 ERROR ;REPORT ERROR. >>>> ERROR #201 <<<<<
    025734 104460 TRAP C$ERROR
5051 025736 000415 BR 60$ ;EXIT THE TEST.
5052

```

HARDWARE TEST

- MRSTA -

```

5053 ;Report MR bit clear immediately after DUT reset.
5054 025740 012767 000312 156022 6$: MOV #202,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5055 025746 012701 006346 MOV #EM0203,R1 ;SELECT ERROR MESSAGE.
5056 025752 ERROR ;REPORT ERROR. >>>> ERROR #202 <<<<<
      025752 104460 TRAP C$ERROR
5057 025754 000406 BR 60$ ;EXIT THE TEST.
5058
5059 ;Report MR clear within 1/2 second of DUT reset.
5060 025756 012767 000313 156004 8$: MOV #203,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5061 025764 012701 006511 MOV #EM0204,R1 ;SELECT ERROR MESSAGE.
5062 025770 ERROR ;REPORT ERROR. >>>> ERROR #203 <<<<<
      025770 104460 TRAP C$ERROR
5063
5064 025772 005067 154354 60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5065
5066 025776 ENDTST
      025776 L10027:
      025776 104401 TRAP C$ETST

```

HARDWARE TEST

- MRSSTA -

```

5068
5069
5070
5071
5072
5073
5074
5075 026000
      026000
5076      000003
5077 026000 012767 00003 154320
5078 026006 012767 177777 154336
5079 026014 012767 000001 155744
5080 026022 012767 006670 155742
5081 026030 012767 016406 155736
5082
5083
5084
5085 026036 012701 011610
5086 026042 012702 000040
5087 026046 005003
5088 026050 016704 154170
5089 026054 004767 172260
5090 026060 103410
5091
5092
5093
5094
5095 026062
      026062 104433
5096 026064 004767 174246
5097 026070 012701 011610
5098 026074 004767 172240
5099 026100 103024
5100
5101
5102
5103
5104 026102 012701 000310
5105 026106 010214
5106 026110 004767 174222
5107 026114 004767 172220
5108 026120 103007
5109 026122 012702 000310
5110 026126 160102
5111 026130 020227 000012
5112 026134 002415
5113 026136 000431
5114
5115
5116
5117 026140 012701 011300
5118 026144 004767 172170
5119 026150 103416
5120
5121
5122

```

```

.SBTTL HARDWARE TEST - MRSSTA -
;.. *****
;* - Master Reset With Skip Selftest Test -
;* This test verifies that the Master Reset bit will clear after a device
;* reset and the skipping of the DUT ROM based selftest.
;*
;-- *****
      BGNTST
;..
;.. T3::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (3)
      MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
      MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV #EM0301,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
      MOV #ERO201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
;..
;.. Wait up to 5 seconds for the DUT Master Reset bit to clear.
;..
      MOV #5000.,R1 ;TIME OUT VALUE IS 5.0 SECONDS.
      MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
      CLR R3 ;WAITING FOR BIT TO CLEAR.
      MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
      JSR PC,MSLGET ;WAIT FOR DUT CSR MR BIT TO CLEAR.
      BCS 2$ ;SKIP TO RESET DUT IF MR CLEAR.
;..
;.. DUT Master Reset bit did not go clear. Device may be stuck in some
;.. odd state. Try to reset device with a bus reset.
;..
      BRESET ;NO, TRY TO JOG DEVICE WITH BUS RESET.
;..
      JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST. TRAP C$RESET
      MOV #5000.,R1 ;TIME OUT VALUE IS 5.0 SECONDS.
      JSR PC,MSLGET ;WAIT FOR DUT CSR MR BIT TO CLEAR.
      BCC 6$ ;GO REPORT ERROR IF MR BIT DID NOT CLEAR.
;..
;.. Set the Master Reset bit, try to skip the selftest, and verify that the
;.. MR bit clears within 1/5 second.
;..
2$: MOV #200.,R1 ;TIME-OUT VALUE IS 1/5 SECOND.
      MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
      JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST.
      JSR PC,MSLGET ;WAIT FOR DUT CSR MR BIT TO CLEAR.
      BCC 4$ ;GO FIND OUT WHAT IS WRONG IF MR NOT CLEAR.
      MOV #200.,R2
      SUB R1,R2 ;CALCULATE # OF MS FOR MR TO CLEAR.
      CMP R2,#10.
      BLT 8$ ;GO REPORT ERROR IF MR CLEAR IN < 10 MS.
      BR 60$ ;EXIT THE TEST WITHOUT ERROR.
;..
;.. MR did not clear within 1/5 second, see if it clears within 5 seconds.
;..
4$: MOV #4800.,R1 ;TIME-OUT VALUE IS 5 SECONDS MINUS 1/5 SECOND.
      JSR PC,MSLGET ;WAIT FOR DUT CSR MR BIT TO CLEAR.
      BCS 10$ ;GO REPORT ERROR IF MR CLEARED FINALLY.
;..
;.. Error reports:
;..

```

HARDWARE TEST

- MRSSTA -

```

5123
5124 026152 012767 000455 155610 6$: ;Report MR bit would not clear after a DUT reset.
      MOV #0301.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5125 026160 012701 006173          MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5126 026164          ERROR ;REPORT ERROR. >>>> ERRJR #0301 <<<<<
      026164 104460          TRAP C$ERROR
5127 026166 000415          BR 60$ ;EXIT THE TEST.
5128
5129
5130 026170 012767 000456 155572 8$: ;Report MR bit clear within 10 ms after DUT reset.
      MOV #0302.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5131 026176 012701 006733          MOV #EM0302,R1 ;SELECT ERROR MESSAGE.
5132 026202          ERROR ;REPORT ERROR. >>>> ERROR #0302 <<<<<
      026202 104460          TRAP C$ERROR
5133 026204 000406          BR 60$ ;EXIT THE TEST.
5134
5135
5136 026206 012767 000457 155554 10$: ;Report MR cleared between 1/5 second and 5 seconds of DUT reset.
      MOV #0303.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5137 026214 012701 007073          MOV #EM0303,R1 ;SELECT ERROR MESSAGE.
5138 026220          ERROR ;REPORT ERROR. >>>> ERROR #0303 <<<<<
      026220 104460          TRAP C$ERROR
5139
5140 026222 005067 154124          60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5141
5142 026226          ENDTST
      026226          L10030:
      026226 104401          TRAP C$ETST

```

HARDWARE TEST

- RXCHRA -

```

5144 .SBTTL HARDWARE TEST - RXCHRA -
5145 ;** *****
5146 ;* - RBUF Register RX Character Field Test -
5147 ;* This test verifies that the RX Character Field of the DUT RBUF register
5148 ;* appears to be functioning correctly. This test uses the codes which
5149 ;* should be in the FIFO after a board reset and skip selftest sequence.
5150 ;*
5151 ;-- *****
5152 026230 BGNTST
5153 026230 000004 T4::
5154 026230 012767 000004 154070 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5155 026236 012767 177777 154106 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (4)
5156 026244 012767 000001 155514 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5157 026252 012767 007252 155512 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5158 026260 012767 016406 155506 MOV #EM0401,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5159 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5160 ;*
5161 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
5162 ; and wait up to 5 seconds for the MR bit to clear.
5163 026266 012701 011610 ;--
5164 026272 012702 000040 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5165 026276 005003 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5166 026300 016704 153740 CLR R3 ;WAITING FOR BIT TO CLEAR.
5167 026304 010214 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5168 026306 004767 174024 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5169 026312 004767 172022 JSR PC,SKPSTS ;SKIP THE SELFTEST.
5170 026316 103015 JSR PC,MSLGET ;WAIT FOR DUT CSR MR BIT TO CLEAR.
5171 BCC 4$ ;GO REPORT ERROR IF MR DID NOT CLEAR.
5172 ;*
5173 ; Read 6 characters from the DUT and verify that they are valid selftest
5174 ; codes.
5175 026320 012400 ;--
5176 026322 012701 000006 MOV (R4)+,R0 ;INCREMENT POINTER TO POINT TO DUT RBUF REGSTR.
5177 026326 011402 2$: MOV #6,R1 ;INITIALIZE THE LOOP COUNTER.
5178 026330 010200 MOV (R4),R2 ;READ A CHARACTER FROM THE DUT RBUF REGISTER.
5179 026332 042700 177476 MOV R2,R0
5180 026336 020027 000201 BIC #177476,R0 ;REMOVE ALL BUT BITS SPECIFIC TO SELFTEST CODE.
5181 026342 001012 CMP R0,#201 ;CHECK THAT BITS 0,6, AND 7 ARE CORRECT.
5182 026344 005301 BNE 6$ ;GO REPORT ERROR IF CODE IS NOT SELFTEST CODE.
5183 026346 001367 DEC R1 ;COUNT THIS LOOP ITERATION.
5184 026350 000415 BNE 2$ ;LOOP IF NOT ALL LINES DONE.
5185 BR 60$ ;EXIT TEST, NO ERROR FOUND.
5186 ;*
5187 ; Error reports:
5188 ;--
5189 ;Report MR bit would not clear after a DUT reset.
5190 026352 012767 000621 155410 4$: MOV #0401.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5191 026360 012701 006173 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5192 026364 104460 ERROR ;REPORT ERROR. >>>> ERROR #0401 <<<<<
5193 026366 000406 BR 60$ ;EXIT THE TEST. TRAP C$ERROR
5194
5195 ;Report improper code found in DUT RBUF after reset (skip selftest).
5196 026370 012767 000622 155372 6$: MOV #0402.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5197 026376 012701 007321 MOV #EM0402,R1 ;SELECT ERROR MESSAGE.
5198 026402 ERROR ;REPORT ERROR. >>>> ERROR #0402 <<<<<

```

HARDWARE TEST - RXCHRA -

5199	026402	104460							TRAP	C\$ERROR
5200	026404	005067	153742	60\$:	CLR	CTRLCF			;INDICATE THAT WE COMPLETED THE TEST.	
5201										
5202	026410					ENDTST				
	026410							L10031:		
	026410	104401						TRAP		C\$ETST

HARDWARE TEST

- RXFFDA -

```

5204 .SBTTL HARDWARE TEST - RXFFDA -
5205 ;+ *****
5206 ;* - RBUF Register RX Flag Field Test -
5207 ;* This test verifies that the Field of 3 flag bits in the RBUF reads
5208 ;* as all ones when the Selftest codes are being read from the DUT
5209 ;* after a board reset and skip selftest sequence.
5210 ;*
5211 ;-- *****
5212 026412 BGNTST
5213 026412 T5::
5213 000005 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5214 026412 012767 000005 153706 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (5)
5215 026420 012767 177777 153724 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5216 026426 012767 000001 155332 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5217 026434 012767 007471 155330 MOV #EM0501,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5218 026442 012767 016406 155324 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5219
5220 ;+
5220 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
5221 ; and wait up to 5 seconds for the MR bit to clear.
5222 ;--
5223 026450 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5224 026454 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5225 026460 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5226 026462 016704 153556 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5227 026466 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5228 026470 004767 173642 JSR PC,SKPSTS ;SKIP THE SELFTEST.
5229 026474 004767 171640 JSR PC,MSLGET ;WAIT FOR DUT CSR MR BIT TO CLEAR.
5230 026500 103013 BCC 4$ ;GO REPORT ERROR IF MR DID NOT CLEAR.
5231
5232 ;+
5232 ; Read 8 characters from the DUT and verify that all 3 RX error flags are
5233 ; set for each characters.
5234 ;--
5235 026502 012400 MOV (R4)+,R0 ;INCREMENT POINTER TO POINT TO DUT RBUF REGSTR.
5236 026504 012701 000010 MOV #8.,R1 ;INITIALIZE THE LOOP COUNTER.
5237 026510 011402 2$: MOV (R4),R2 ;READ A CHARACTER FROM THE DUT RBUF REGISTER.
5238 026512 012700 070000 MOV #70000,R0
5239 026516 040200 BIC R2,R0 ;CALCULATE BIT MAP OF CLEAR RX ERROR FLAGS.
5240 026520 001012 BNE 6$ ;GO REPORT ERROR IF NOT ALL RX ERROR FLAGS SET.
5241 026522 005301 DEC R1 ;COUNT THIS LOOP ITERATION.
5242 026524 001371 BNE 2$ ;LOOP IF NOT ALL LINES DONE.
5243 026526 000415 BR 60$ ;EXIT TEST, NO ERROR FOUND.
5244
5245 ;+
5245 ; Error reports:
5246 ;
5247 ;
5248 ;Report MR bit would not clear after a DUT reset.
5249 026530 012767 000765 155232 4$: MOV #0501.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5250 026536 012701 006173 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5251 026542 ERROR ;REPORT ERROR. >>>> ERROR #0501 <<<<<
5251 026542 104460 TRAP C$ERROR
5252 026544 000406 BR 60$ ;EXIT THE TEST.
5253
5254 ;Report one or more RX error flags found set with selftest code.
5255 026546 012767 000766 155214 6$: MOV #0502.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5256 026554 012701 007537 MOV #EM0502,R1 ;SELECT ERROR MESSAGE.
5257 026560 ERROR ;REPORT ERROR. >>>> ERROR #0502 <<<<<
5257 026560 104460 TRAP C$ERROR

```


HARDWARE TEST - RXFFDA -

5258
5259 026562 005067 153564 60\$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5260
5261 026566
026566
026566 104401
L10032: TRAP C\$ETST

HARDWARE TEST

- RDAA -

```

5263 .SBTTL HARDWARE TEST - RDAA -
5264 ;+ *****
5265 ;* - CSR RX Data Available Bit Test -
5266 ;* This test verifies that the DUT CSR RX Data Available Bit is set by the
5267 ;* inclusion of the selftest codes in the DUT FIFO and that the bit clears
5268 ;* after the FIFO has been emptied.
5269 ;*
5270 ;-- *****
5271 026570 BGNTST
      026570
5272 000006 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5273 026570 012767 000006 153530 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (6)
5274 026576 012767 177777 153546 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5275 026604 012767 000001 155154 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5276 026612 012767 010073 155152 MOV #EM0601,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5277 026620 012767 016406 155146 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5278
5279 ;+
5280 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
5281 ; and wait up to 5 seconds for the MR bit to clear.
5282 026626 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5283 026632 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5284 026636 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5285 026640 016704 153400 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5286 026644 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5287 026646 004767 173464 JSR PC,SKPSTS ;SKIP THE SELFTEST.
5288 026652 004767 171462 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5289 026656 103016 BCC 4$ ;GO REPORT ERROR IF MR DID NOT CLEAR.
5290
5291 ;+
5292 ; Check that the RX Data Available bit is set.
5293 026660 032714 000200 BIT #BIT7,(R4) ;TEST THE DUT RX.DATA.AVAIL BIT.
5294 026664 001422 BEQ 6$ ;GO REPORT ERROR IF BIT IS NOT SET.
5295
5296 ;+
5297 ; Read characters from the DUT RX FIFO and wait for RX.DATA.AVAIL to go clear.
5298 026666 012705 001130 MOV #600.,R5 ;ALLOW READING 600 CHARS BEFORE ERROR.
5299 026672 010403 MOV R4,R3
5300 026674 012300 MOV (R3)+,R0 ;CALCULATE THE RBUF ADDRESS.
5301 026676 011300 MOV (R3),R0 ;READ A CHARACTER FROM THE RX FIFO.
5302 026700 032714 000200 BIT #BIT7,(R4) ;TEST THE DUT RX.DATA.AVAIL BIT.
5303 026704 001427 BEQ 60$ ;EXIT TEST WITHOUT ERROR IF RX.DATA.AVAIL CLR.
5304 026706 005305 DEC R5 ;COUNT THE CHARACTER JUST READ.
5305 026710 001372 BNE 2$ ;LOOP IF NOT TOO MANY CHARS READ FROM FIFO.
5306 026712 000416 BR 8$ ;GO REPORT ERROR IF RX.DATA.AVAIL WOULDN'T CLR.
5307
5308 ;+
5309 ; Error reports:
5310 ;
5311 ;Report MR bit would not clear after a DUT reset.
5312 026714 012767 001131 155046 4$: MOV #0601.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5313 026722 012701 006173 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5314 026726 ERROR ;REPORT ERROR. >>>> ERROR #0601 <<<<<
      026726 104460 TRAP C$ERROR
5315 026730 000415 BR 60$ ;EXIT THE TEST.
5316
5317 ;Report that RX.DATA.AVAIL bit was not set after a reset complet'on.

```

HARDWARE TEST

- RDAA -

```

5318 026732 012767 001132 155030 6$:   MOV   #0602.,ERRNBR   ;SET THE ERROR NUMBER IN ERROR TABLE.
5319 026740 012701 010127              MOV   #EM0602,R1     ;SELECT ERROR MESSAGE.
5320 026744              ERROR          ;REPORT ERROR.      >>>> ERROR #0602 <<<<<
                    026744 104460              TRAP   C$ERROR
5321 026746 000406              BR    60$           ;EXIT THE TEST.
5322
5323              ;Report that RX.DATA.AVAIL bit could not be cleared by purging FIFO.
5324 026750 012767 001133 155012 8$:   MOV   #0603.,ERRNBR   ;SET THE ERROR NUMBER IN ERROR TABLE.
5325 026756 012701 010307              MOV   #EM0603,R1     ;SELECT ERROR MESSAGE.
5326 026762              ERROR          ;REPORT ERROR.      >>>> ERROR #0603 <<<<<
                    026762 104460              TRAP   C$ERROR
5327
5328 026764 005067 153362          60$:   CLR   CTRLCF        ;INDICATE THAT WE COMPLETED THE TEST.
5329
5330 026770              ENDTST
                    026770              L10033:
                    026770 104401              TRAP   C$ETST

```

HARDWARE TEST

- RDVA -

```

5332 .SBTTI  HARDWARE TEST          - RDVA -
5333 ;+ *****
5334 ;*          - RBUF RX Data Valid Bit Test -
5335 ;*          This test verifies that the DUT RBUF RX Data Valid Bit is set by the
5336 ;*          inclusion of the selftest codes in the DUT FIFO and that the bit clears
5337 ;*          after the FIFO has been emptied.
5338 ;*
5339 ;-- *****
5340 026772          BGNTST
      026772
5341          000007          TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5342 026772 012767 000007 153326          MOV #TNUM,TSTNUM          ;SET UP THE TEST NUMBER. (7)
5343 027000 012767 177777 153344          MOV #-1,CTRLCF          ;INDICATE THAT WE ARE WITHIN A TEST.
5344 027006 012767 000001 154752          MOV #1,ERRTYP          ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5345 027014 012767 010472 154750          MOV #EM0701,ERRMSG          ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5346 027022 012767 016406 154744          MOV #ER0201,ERRBLK          ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5347
5348 ;+
5349 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
5350 ; and wait up to 5 seconds for the MR bit to clear.
5351 027030 012701 011610          MOV #5000.,R1          ;TIME-OUT VALUE IS 5.0 SECONDS.
5352 027034 012702 000040          MOV #BIT05,R2          ;WAITING FOR MASTER RESET BIT.
5353 027040 005003          CLR R3          ;WAITING FOR BIT TO CLEAR.
5354 027042 016704 153176          MOV CSRA,R4          ;BIT IS IN THE DUT'S CSR.
5355 027046 010214          MOV R2,(R4)          ;SET THE DUT MASTER RESET BIT.
5356 027050 004767 173262          JSR PC,SKPSTS          ;SKIP THE SELFTEST.
5357 027054 004767 171260          JSR PC,MSLGET          ;WAIT FOR DUT_CSR MR BIT TO CLEAR.
5358 027060 103012          BCC 4$          ;GO REPORT ERROR IF MR DID NOT CLEAR.
5359
5360 ;+
5361 ; Check that the RX Data Valid bit is set.
5362 027062 012400          MOV (R4)+,R0          ;INCREMENT POINTER TO PNT TO DUT RBUF REG.
5363 027064 005714          TST (R4)          ;TEST THE DUT RX.DATA.VALID BIT.
5364 027066 100016          BPL 6$          ;GO REPORT ERROR IF BIT IS NOT SET.
5365
5366 ;+
5367 ; Read characters from the DUT RX FIFO and wait for RX.DATA.VALID to go clear.
5368 027070 012705 001130          MOV #600.,R5          ;ALLOW READING 600 CHARS BEFORE ERROR.
5369 027074 011400          2$: MOV (R4),R0          ;READ A CHARACTER FROM THE RX FIFO.
5370 027076 100027          BPL 60$          ;EXIT TEST WITHOUT ERROR IF BIT IS CLEAR.
5371 027100 005305          DEC R5          ;COUNT THE CHARACTER JUST READ.
5372 027102 001374          BNE 2$          ;LOOP IF NOT TOO MANY CHARS READ FROM FIFO.
5373 027104 000416          BR 8$          ;GO REPORT ERROR IF RX.DATA.VALID WOULDN'T CLR.
5374
5375 ;+
5376 ; Error reports:
5377 ;-
5378 ;Report MR bit would not clear after a DUT reset.
5379 027106 012767 001275 154654 4$: MOV #0701.,ERRNBR          ;SET THE ERROR NUMBER IN ERROR TABLE.
5380 027114 012701 006173          MOV #EM0202,R1          ;SELECT ERROR MESSAGE.
5381 027120 104460          ERROR          ;REPORT ERROR. >>>> ERROR #0701 <<<<<
5382 027122 000415          BR 60$          ;EXIT THE TEST. TRAP C$ERROR
5383
5384 ;+
5385 027124 012767 001276 154636 6$: ;Report that RX.DATA.VALID bit was not set after a reset completion.
5386 027132 012701 010527          MOV #0702.,ERRNBR          ;SET THE ERROR NUMBER IN ERROR TABLE.
          MOV #EM0702,R1          ;SELECT ERROR MESSAGE.

```

HARDWARE TEST

- RDVA -

```

5387 027136          ERROR          ;REPORT ERROR.          >>>> ERROR #0702 <<<<<
      027136 104460          ;                               TRAP      C$ERROR
5388 027140 000406          BR          60$          ;EXIT THE TEST.
5389
5390          ;Report that RX.DATA.VALID bit could not be cleared by purging FIFO.
5391 027142 012767 001277 154620 8$:  MOV      #0703,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5392 027150 012701 010707          MOV      #EM0703,R1 ;SELECT ERROR MESSAGE.
5393 027154          ERROR          ;REPORT ERROR.          >>>> ERROR #0703 <<<<<
      027154 104460          ;                               TRAP      C$ERROR
5394
5395 027156 005067 153170          60$:  CLR      CTRLCF          ;INDICATE THAT WE COMPLETED THE TEST.
5396
5397 027162          ENDTST
      027162
      027162 104401          L10034:
                               TRAP      C$ETST

```

HARDWARE TEST

- RLNA -

```

5399 .SBTTL HARDWARE TEST - RLNA -
5400 ;+ *****
5401 ;* - RBUF RX Line Number Field Test -
5402 ;* This test verifies that the DUT RBUF RX Line Number field is working
5403 ;* correctly by utilizing the selftest codes which are put in the RX
5404 ;* FIFO after a board reset.
5405 ;*
5406 ;-- *****
5407 027164 BGNTST
      027164
5408 000010 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5409 027164 012767 000010 153134 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (8)
5410 027172 012767 177777 153152 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5411 027200 012767 000001 154560 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5412 027206 012767 011072 154556 MOV #EM0801,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5413 ;+
5414 ; Set the DUT CSR Maser Reset (MR) bit, perform the skip selftest sequence,
5415 ; and wait up to 5 seconds for the MR bit to clear.
5416 ;-
5417 027214 012701 011610 MOV #5000.,R1 ;TIME OUT VALUE IS 5.0 SECONDS.
5418 027220 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5419 027224 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5420 027226 016704 153012 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5421 027232 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5422 027234 004767 173076 JSR PC,SKPSTS ;SKIP THE SELFTEST.
5423 027240 004767 171074 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5424 027244 103016 BCC 4$ ;GO REPORT ERROR IF MR DID NOT CLEAR.
5425 ;+
5426 ; Read characters from the DUT RX FIFO and verify that the line numbers are
5427 ; correct.
5428 ; One character is read from the FIFO for each possible line on the DUT.
5429 ;-
5430 027246 005001 CLR R1 ;CLEAR THE LINE COUNTER.
5431 027250 012400 MOV (R4)+,R0 ;INCREMENT POINTER TO PNT TO THE DUT RBUF REG.
5432 027252 011402 2$: MOV (R4),R2 ;READ A CHARACTER FROM THE DUT RX FIFO.
5433 027254 010203 MOV R2,R3
5434 027256 000303 - SWAB R3
5435 027260 042703 177760 BIC #177760,R3 ;REMOVE ALL BUT LINE NUMBER BITS.
5436 027264 020301 CMP R3,R1 ;COMPARE WITH EXPECTED LINE NUMBER.
5437 027266 001017 BNE 6$ ;GO REPORT ERROR IF LINE NUMBERS DON'T MATCH.
5438 027270 005201 INC R1 ;INCREMENT THE EXPECTED LINE NUMBER.
5439 027272 020127 000010 CMP R1,#NUMLNS ;COMPARE WITH NUMBER OF LINES ON DUT.
5440 027276 001365 BNE 2$ ;LOOP UNTIL CODES FOR ALL LINES ARE READ.
5441 027300 000423 BR 60$ ;EXIT TEST WITHOUT ERROR.
5442 ;+
5443 ; Error reports:
5444 ;-
5445 ;Report MR bit would not clear after a DUT reset.
5446 4$: MOV #0801.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5447 027302 012767 001441 154460 MOV #ER0503,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5448 027310 012767 016470 154456 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5449 027316 012701 006173 ERROR ;REPORT ERROR. >>>> ERROR #0801 <<<<<
5450 027322 104460 TRAP C$ERROR
5451 027324 000411 BR 60$ ;EXIT THE TEST.
5452 ;Report that RX Line Number field is wrong for selftest code.
5453

```

HARDWARE TEST

- RLNA -

```

5454 027326 012767 001442 154434 6$:  MOV #0802,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5455 027334 012767 016406 154432  MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5456 027342 012701 011132  MOV #EM0802,R1 ;SELECT ERROR MESSAGE.
5457 027346  ERROR ;REPORT ERROR. >>>>> ERROR #0802 <<<<<
      027346 104460 TRAP C$ERROR
5458
5459 027350 005067 152776 60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5460
5461 027354  ENDTST
      027354 L10035:
      027354 104401 TRAP C$ETST

```

HARDWARE TEST

- BMPCHK -

```

5463 .SBTTL HARDWARE TEST - BMPCHK -
5464 ;* *****
5465 ;* - BMP check Test -
5466 ;* This test is used to verify that the DUT does not immediately fail
5467 ;* the on-board background-monitor program, and hence invalidate
5468 ;* succeeding tests.
5469 ;* This test looks for BMP codes in the fifo for a set period immediately
5470 ;* after the self-test is skipped.
5471 ;* Any BMP codes that are found are saved on the queue and are also
5472 ;* reported in this test.
5473 ;*
5474 ;*
5475 ;-- *****
    027356 BGNTST
    027356
5476 000011 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5477 027356 012767 000011 152742 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (9)
5478 027364 012767 177777 152760 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5479 027372 012767 000001 154366 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5480 027400 012767 001605 154362 MOV #0901.,ERRNBR ;SET THE ERROR NUMBER.
5481 ;*
5482 ; Wait up to 3 seconds for the DUT Master Reset bit to clear.
5483 ; If time-out occurs, then exit this test.
5484 ;--
5485 027406 012701 005670 MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
5486 027412 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5487 027416 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5488 027420 016704 152620 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5489 027424 004767 170710 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5490 027430 103027 BCC 50$ ;ABORT THE TEST IF MR DID NOT CLEAR.
5491 ;*
5492 ; Reset the DUT, skip the self-test.
5493 ;--
5494 027432 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5495 027434 004767 172676 JSR PC,SKPSTS ;WRITE THE SKIP SELFTST CODES TO THE DUT.
5496 ;*
5497 ; Wait for Master reset to clear. Delay for 500 milli-secs before purging
5498 ; the fifo.
5499 ;--
5500 027440 012704 000764 MOV #500.,R4 ;TIME-OUT VALUE IS 500 MILLI-SECONDS.
5501 027444 004767 170630 JSR PC,DELAY ;WAIT FOR BMP TO BEGIN EXECUTION.
5502 027450 004767 171242 JSR PC,PUFIFO ;PURGE THE FIFO, SAVING ANY BMP CODES.
5503 027454 103015 BCC 50$ ;ABORT THE TEST IF THE FIFO DID NOT CLEAR.
5504 ;*
5505 ; Report the error if any BMP codes were found.
5506 ;--
5507 027456 016702 153042 MOV BMPCQP,R2 ;GET THE CONTENTS OF THE POINTER TO THE BMP Q.
5508 027462 012703 002526 MOV #BMPCQB,R3 ;GET THE START ADDRESS OF THE QUEUE.
5509 027466 020203 CMP R2,R3 ;SEE IF THE POINTER HAS MOVED FROM THE BASE.
5510 027470 001414 BEQ 60$ ;EXIT NO CODES IN THE QUEUE.
5511 ;*
5512 ; There is at least one BMP code in the queue. Report the error.
5513 ;--
5514 ;Report error BMP CODE FOUND IN TEST nn, BMP CODE:nnnnnn"
5515 027472 012701 011236 MOV #EM0902,R1 ;PASS THE MESSAGE TO BE REPORTED.
5516 027476 104455 ERRDF 0901,EM0901,ER9301 ; >>>> ERROR #0901 <<<<<.
    027500 001605 TRAP C$ERDF
    .WORD 901

```


HARDWARE TEST

- BMPCHK -

```

027502 011205
027504 017450
5517 027506 000405 BR 60$ .WORD EM0901
5518 .WORD ER9301
5519 027510 012767 001606 154252 50$: MOV #902,ERRNBR ;SET >>>> ERROR #0902 <<<<<
5520 027516 004767 172742 JSR PC,TSABRT ;REPORT NON-TEST RELATED ERROR.
5521
5522 027522 005067 152624 60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5523
5524 027526 ENDTST
027526
027526 104401 L10036: TRAP C$ETST

```

HARDWARE TEST

- BMPCHK -

```

5526
5527
5528 .SBTTL HARDWARE TEST - SKSELF -
5529 ;* *****
5530 ;* - Skip Self-test Test -
5531 ;* This test verifies that the DUT skips the self-test within the
5532 ;* time allowed, and that the fifo contains the correct codes after its
5533 ;* completion.
5534 ;*
5535 ;*
5536 027530 BCNTST
      027530
5537
5537 000012 TNUM = TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5538 027530 012767 000012 152570 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (10)
5539 027536 012767 177777 152606 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5540 027544 012767 000001 154214 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5541 027552 012767 011272 154212 MOV #EM1001,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5542 027560 012767 016470 154206 MOV #ER0503,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5543
5544 ;*
5544 ; Wait up to 3 seconds for the DUT Master Reset bit to clear.
5545 ; If time-out occurs, then exit this test.
5546 ;-
5547 027566 012701 005670 MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
5548 027572 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5549 027576 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5550 027600 016704 152440 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5551 027604 004767 170530 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5552 027610 103037 BCC 50$ ;ABORT THE TEST IF MR DID NOT CLEAR.
5553
5554 ;*
5554 ; Determine if the DUT takes too short or too long a time to skip the self-test
5555 ; Set-up a time-out of 50 milli-second, if MR is clear in less than 10 milli
5556 ; -second, or greater than 50 milli-seconds, report the error.
5557 ;-
5558 027612 012701 000062 MOV #50.,R1 ;TIME-OUT VALUE IS 50 MILLI SECONDS.
5559 027616 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5560 027620 004767 172512 JSR PC,SKPSTS ;WRITE THE SKIP SELFTEST CODES TO THE DUT.
5561 027624 004767 170510 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5562 027630 103011 BCC 2$ ;GO REPORT ERR IF SKIPPING STEST TOOK TOO LONG.
5563 027632 020127 000050 CMP R1,#40.
5564 027636 003015 BGT 4$ ;GO REP ERR IF SELFTEST COMPLETED IN < 10 MS.
5565
5566 ;*
5566 ; Self-test completed within 10 milli-sec to 50 milli-seconds.
5567 ; Verify that the self-test codes in the fifo are "good" codes ,ie the DUT
5568 ; successfully completed the self-test.
5569 ; This subroutine reports errors with numbers >>>> 1003 thru 1007 <<<<<.
5570 ;-
5571 027640 012767 001753 154122 MOV #1003.,ERRNBR ;SET ERROR NUMBER TO 1003.
5572 027646 004767 171750 JSR PC,RSTRPT ;CHECK SELF-TEST CODES IN THE FIFO.
5573 027652 000423 BR 60$ ;EXIT TEST.
5574
5575 ;*
5575 ; Error reports:
5576 ;-
5577 ;Report Skip Self-test took too long.
5578 027654 012767 001751 154106 2$: MOV #1001.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5579 027662 012701 011336 MOV #EM1002,R1 ;SELECT ERROR MESSAGE.
5580 027666 104460 ERROR ;REPORT ERROR. >>>> ERROR #1001 <<<<<
      TRAP C$ERROR

```

HARDWARE TEST

- SKSELF -

```

5581 027670 000414          BR      60$          ;EXIT THE TEST.
5582
5583
5584 027672 012767 001752 154070 4$: ;Report Skip Self-test completed too soon.
      MOV      #1002.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5585 027700 012701 011423          MOV      #EM1003,R1 ;SELECT ERROR MESSAGE.
5586 027704          ERROR          ;REPORT ERROR. >>>> ERROR #1002 <<<<<
      027704 104460          TRAP      C$ERROR
5587 027706 000405          BR      60$          ;EXIT THE TEST.
5588
5589 027710 012767 001753 154052 50$: MOV      #1003.,ERRNBR ;SET ERROR NUMBER.
5590 027716 004767 172542          JSR      PC,TSABRT ;REPORT NON-TEST RELATED ERROR.
5591
5592 027722 005067 152424          60$: CLR      CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5593
5594 027726          ENDTST
      027726          L10037:
      027726 104401          TRAP      C$ETST

```

HARDWARE TEST

- SKSELF -

```

5596
5597
5598
5599
5600
5601
5602
5603
5604
5605 027730
      027730
5606      000013
5607 027730 012767 000013 152370
5608 027736 012767 177777 152406
5609 027744 012767 000001 154014
5610 027752 012767 011501 154012
5611 027760 012767 016470 154006
5612
5613
5614
5615
5616 027766 012701 005670
5617 027772 012702 000040
5618 027776 005003
5619 030000 016704 152240
5620 030004 004767 170330
5621 030010 103044
5622
5623
5624
5625 030012 010214
5626 030014 004767 172316
5627
5628
5629
5630
5631 030020 012701 000005
5632 030024 012702 020000
5633 030030 010203
5634 030032 016704 152206
5635 030036 004767 170276
5636 030042 103020
5637
5638
5639
5640
5641
5642
5643
5644 030044 012701 000017
5645 030050 005003
5646 030052 004767 170262
5647 030056 103012
5648 030060 010105
5649 030062 012701 000001
5650 030066 052703 020000
5651 030072 004767 170242

```

```

.SBTTL HARDWARE TEST - DFSKST -
;+ *****
;* - Diagnostic fail bit, skip self-test test
;* This test verifies that the diagnostic fail bit of the DUT, correctly
;* changes state as the on-board selftest is skipped.
;*
;-- *****
      BGNTST
                                     T11::
5606      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5607      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (11)
5608      MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5609      MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5610      MOV #EM1101,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5611      MOV #ER0503,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
;+
; Wait up to 3 seconds for the DUT Master Reset bit to clear.
; If time-out occurs, then exit this test.
;-
      MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
      MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
      CLR R3 ;WAITING FOR BIT TO CLEAR.
      MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
      JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
      BCC 50$ ;ABORT THE TEST IF MR DID NOT CLEAR.
;+
; Reset the DUT, skip the self test.
;-
      MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
      JSR PC,SKPSTS ;WRITE THE SKIP SELFTEST CODES TO THE DUT.
;+
; Set time out of 5 milli seconds, wait for Diag_fail bit to set.
; If time-out occurs go report the error.
;-
      MOV #5,R1 ;TIME-OUT VALUE IS 5 MILLI-SECONDS.
      MOV #BIT13,R2 ;WAITING FOR DIAGNOSTIC FAIL BIT.
      MOV R2,R3 ;WAITING FOR BIT TO SET.
      MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
      JSR PC,MSLGET ;WAIT FOR DUT_CSR_DF BIT TO CLEAR.
      BCC 4$ ;IF DIAG_FAIL DID NOT SET, GO REPORT ERROR.
;+
; Set time-out of 15 milli-secs, wait for Diag Fail to clear.
; If time-out occurs go report the error.
; Verify the Diag_fail bit is in a stable state before continuing. Loop
; back if the state was transitory, using the remainder of the 15 ms time out.
;
2$:
      MOV #15.,R1 ;TIME OUT VALUE IS 15 MILLI-SECONDS.
      CLR R3 ;WAITING FOR BIT TO CLEAR.
      JSR PC,MSLGET ;WAIT FOR DUT_CSR_DF BIT TO CLEAR.
      BCC 4$ ;IF DIAG_FAIL DID NOT CLEAR, GO REPORT ERROR.
      MOV R1,R5 ;SAVE THE REMAINING TIME OUT VALUE.
      MOV #1,R1 ;SET TIME OUT OF 1 MILLI-SECOND.
      BIS #BIT13,R3 ;WAIT FOR BIT TO SET.
      JSR PC,MSLGET ;DOUBLE CHECK TO ELIMINATE NOISE PROBLEMS.

```

HARDWARE TEST

- DFSKST -

```

5652 030076 103016          BCC      60$      ;EXIT IF DIAG_FAIL BIT STILL CLEAR.
5653 030100 010501          MOV      R5,R1    ;PASS THE REMAINING TIME-OUT VALUE.
5654 030102 000762          BR       2$      ;LOOP TO CHECK AGAIN.
5655
5656          ;+
5657          ; Error reports:
5658          ;-
5659 030104 012767 002115 153656 4$: ;Report Diagnostic fail bit bad.
5660 030112 012701 011742          MOV      #1101.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5661 030116          MOV      #EM1205,R1 ;SELECT ERROR MESSAGE.
5662 030120 000405          BR       60$      ;REPORT ERROR. >>>>> ERROR #1101 <<<<<
5663          ;EXIT THE TEST. TRAP      C$ERROR
5664 030122 012767 002116 153640 50$: MOV      #1102.,ERRNBR ;SET THE ERROR NUMBER FOR TSABRT RTN.
5665 030130 004767 172330          JSR      PC,TSABRT ;REPORT NON TEST RELATED ERROR.
5666
5667 030134 005067 152212          CLR      CTRLCF   ;INDICATE THAT WE COMPLETED A TEST.
5668
5669 030140          ENDTST
030140
030140 104401          L10040: TRAP      C$ETST
    
```

HARDWARE TEST

- SELFTS -

```

5671 .SBTTL HARDWARE TEST - SELFTS -
5672 ;** *****
5673 ;* - Self-test Test -
5674 ;* This test verifies that the DUT's Self-Test executes within the
5675 ;* time allowed, and that the fifo contains the correct codes after its
5676 ;* completion.
5677 ;*
5678 ;-- *****
5679 030142 BGNTST
      030142
5680 000014 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5681 030142 012767 000014 152156 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (12)
5682 030150 012767 177777 152174 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5683 030156 012767 000001 153602 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5684 030164 012767 011525 153600 MOV #EM1201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5685 030172 012767 016470 153574 MOV #ER0503,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5686
5687 ;+
5688 ; Wait up to 3 seconds for the DUT Master Reset bit to clear.
5689 ; If time-out occurs, then exit this test.
5690 030200 012701 005670 MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
5691 030204 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5692 030210 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5693 030212 016704 152026 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5694 030216 004767 170116 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5695 030222 103062 BCC 50$ ;ABORT THE TEST IF MR DID NOT CLEAR.
5696
5697 ;+
5698 ; Determine if the Self-test takes too short or too long a time to complete.
5699 ; Set-up a time-out of 3 second, if MR is clear in less than 1/2 second, or
5700 ; greater than 3 seconds, report the error.
5701 030224 012701 005670 MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
5702 030230 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5703 030232 004767 170102 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5704 030236 103030 BCC 4$ ;GO REPORT ERROR SELFTEST TOOK TOO LONG.
5705 030240 012702 005670 MOV #3000.,R2 ;CALCULATE # OF MS SELFTEST TO COMPLETE.
5706 030244 160102 SUB R1,R2
5707 030246 020227 000062 CMP R2,#50.
5708 030252 002431 BLT 6$ ;SELFTEST SKIPPED? YES, GO REPORT ERROR.
5709 030254 020227 000764 CMP R2,#500.
5710 030260 002434 BLT 8$ ;GO REP ERR IF SELFTEST COMPLETED IN < 1/2 SEC.
5711
5712 ;+
5713 ; Self test completed within 1sec to 3 seconds.
5714 ; Check the state of the Diagnostic Fail bit, report error if it is set.
5715 030262 032714 020000 BIT #BIT13,(R4) ;DETERMINE IF THE DIAG_FAIL BIT IS CLEAR.
5716 030266 001406 BEQ 2$ ;SKIP ERROR REPORT IF BIT IS CLEAR.
5717 ;Report Diagnostic fail bit bad.
5718 030270 012767 002264 153472 MOV #1204.,ERRNBR ;SET ERROR NUMBER TO IN ERROR TABLE.
5719 030276 012701 011742 MOV #EM1205,R1 ;SELECT THE ERROR MESSAGE.
5720 030302 ERROR ; >>>>> ERROR #1204 <<<<<
      030302 104460 TRAP C$ERROR
5721
5722 ;+
5723 ; Verify that the self-test codes in the fifo are "good" codes ,ie the DUT
5724 ; successfully completed the self test.
5725 ; This subroutine reports errors with numbers >>>>> 1205 thru 1209 <<<<<.

```

HARDWARE TEST

- SELFTS -

```

5726 030304 012767 002265 153456 2$:   MOV   #1205.,ERRNBR ;SET ERROR NUMBER TO 1205.
5727 030312 004767 171304             JSR   PC,RSTRPT   ;CHECK SELF-TEST CODES IN THE FIFO.
5728 030316 000431             BR    60$        ;EXIT TEST.
5729
5730           ;+
5731           ; Error reports:
5732           ;-
5733 030320 012767 002261 153442 4$:   ;Report Self-test took too long to complete.
5734 030326 012701 011544             MOV   #1201.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5735 030332             MOV   #EM1202,R1 ;SELECT ERROR MESSAGE.
5736 030334 104460             ERROR ;REPORT ERROR. >>>> ERROR #1201 <<<<<
5737             BR    60$        TRAP   C$ERROR
5738
5739 030336 012767 002262 153424 6$:   ;Report Self-test did not execute after DUT reset.
5740 030344 012701 011706             MOV   #1202.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5741 030350             MOV   #EM1204,R1 ;SELECT ERROR MESSAGE.
5742 030350 104460             ERROR ;REPORT ERROR. >>>> ERROR #1202 <<<<<
5743             TRAP   C$ERROR
5744 030352 012767 002263 153410 8$:   ;Report Self test competed too soon.
5745 030360 012701 011630             MOV   #1203.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5746 030364             MOV   #EM1203,R1 ;SELECT ERROR MESSAGE.
5747 030366 104460             ERROR ;REPORT ERROR. >>>> ERROR #1203 <<<<<
5748             TRAP   C$ERROR
5749 030370 012767 002272 153372 50$:  MOV   #1210.,ERRNBR ;SET THE ERROR NUMBER FOR TSABRT RTN.
5750 030376 004767 172062             JSR   PC,TSABRT   ;REPORT NON-TEST RELATED ERROR.
5751
5752 030402 005067 151744             60$: CLR   CTRLCF     ;INDICATE THAT WE COMPLETED THE TEST.
5753
5754 030406             ENDTST
5755 030406
5756 030406 104401             L10041:
5757             TRAP   C$ETST

```

HARDWARE TEST - SELFTS -

```

5756
5757
5758
5759
5760
5761
5762
5763
5764
5765
5766 030410
      030410
5767      000015
5768 030410 012767 000015 151710
5769 030416 012767 177777 151726
5770 030424 012767 000001 153334
5771 030432 012767 011766 153332
5772 030440 012767 016470 153326
5773 030446 012767 002425 153314
5774
5775
5776
5777
5778 030454 012701 005670
5779 030460 012702 000040
5780 030464 005003
5781 030466 016704 151552
5782 030472 004767 167642
5783 030476 103120
5784
5785
5786
5787
5788 030500 010214
5789 030502 004767 171630
5790 030506 012701 000764
5791 030512 004767 167622
5792 030516 103110
5793 030520 005267 153244
5794 030524 012700 000010
5795 030530 005003
5796 030532 016704 151510
5797 030536 011402
5798 030540 100077
5799 030542 042702 007402
5800 030546 020227 170001
5801 030552 001002
5802 030554 012703 177777
5803 030560 005300
5804 030562 001365
5805 030564 005703
5806 030566 100466
5807
5808
5809
5810
5811

```

```

.SBTTL HARDWARE TEST - STFAIL -
;+ *****
;* - Self-test fail test -
;* This test verifies that the DUT will report selftest errors via the
;* fifo. And that the Diagnostic fail bit will indicate the error.
;* This is accomplished via a software 'hook' in the self-test, which
;* forces a "Procl to ram error" to be placed in the fifo.
;*
;-- *****
      BGNTST
;+
; T13::
; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
; SET UP THE TEST NUMBER. (13)
; INDICATE THAT WE ARE WITHIN A TEST.
; SET ERROR TYPE AS FATAL IN ERROR TABLE.
; SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
; SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
; SET ERROR NUMBER TO 1301.
      TNUM == TNUM + 1
      MOV #TNUM,TSTNUM
      MOV #-1,CTRLCF
      MOV #1,ERRTYP
      MOV #EM1301,ERRMSG
      MOV #ER0503,ERRBLK
      MOV #1301.,ERRNBR
;+
; Wait up to 3 seconds for the DUT Master Reset bit to clear.
; If time-out occurs, then exit this test.
;--
      MOV #3000.,R1
      MOV #BIT05,R2
      CLR R3
      MOV CSRA,R4
      JSR PC,MSLGET
      BCC 50$
;+
; Reset the DUT, check for rom version 0.
; If Version 0 is found then exit this test.
;--
      MOV R2,(R4)
      JSR PC,SKPSTS
      MOV #500.,R1
      JSR PC,MSLGET
      BCC 50$
      INC ERRNBR
      MOV #8.,R0
      CLR R3
      MOV RBUFA,R4
      MOV (R4),R2
      BPL 50$
      BIC #7402,R2
      CMP R2,#170001
      BNE 4$
      MOV #1,R3
      DEC R0
      BNE 2$
      TST R3
      BMI 60$
;+
; RESET THE DUT
; SKIP THE SELF TEST.
; PASS TIME OUT VALUE OF 500 MILLISECS.
; WAIT FOR MR BIT TO CLEAR.
; GO REPORT ERROR IF TIME-OUT OCCURRED.
; SET ERROR NUMBER TO 1302.
; SET MAXIMUM READ COUNT.
; CLEAR THE ROM VERSION 0 FLAG.
; GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
; READ A CODE FROM THE FIFO.
; GO REPORT ERROR IF THE FIFO IS EMPTY.
; REMOVE THE LINE NUMBER AND PROC INDICATOR.
; COMPARE WITH ROM VERSION #0 CODE.
; ROM VERSION #0? NO, SKIP SETTING FLAG.
; YES, SET THE ROM VERSION #0 FLAGS.
; DECREMENT MAX READ COUNTER.
; LOOP IF 8 CODES HAVE NOT BEEN READ.
; CHECK THE ROM VERSION #1 INDICATOR.
; ROM VERSION 0 IN EITHER PROCESSOR? YES, EXIT.
;+
; Reset the DUT, delay for 25 milli seconds before writing the Fail_self_test
; code to TBUFFCT register on channel 0.
;

```


HARDWARE TEST

- STFAIL -

```

5812 030570 012777 000040 151446      MOV    #BIT05,@CSRA    ;SET DUT MASTER RESET BIT, SELECT CHANNEL 0.
5813 030576 012704 000031              MOV    #25.,R4        ;PASS DELAY PERIOD OF 25 MILLI SECS.
5814 030602 004767 167472              JSR    PC,DELAY        ;WAIT FOR SELFTEST TO INITIALISE.
5815 030606 012777 146314 151446      MOV    #146314,@TXBFCA ;WRITE THE FAIL SELF-TEST CODE TO TBUFFCT REG.
5816
5817      ;+
5818      ; Wait up to 2 seconds for the self-test to complete.
5819      ; If time-out occurs, then exit this test.
5820 030614 005267 153150              ; -
5821 030620 012701 003720              INC    ERRNBR          ;SET ERROR NUMBER TO 1303.
5822 030624 012702 000040              MOV    #2000.,R1      ;TIME-OUT VALUE IS 2.0 SECONDS.
5823 030630 005003              MOV    #BIT05,R2     ;PASS THE BIT MAP OF THE BIT TO TEST.
5824 030632 016704 151406              CLR    R3             ;SET UP THE EXPECTED STATE.
5825 030636 004767 167476              MOV    CSRA,R4       ;BIT IS IN THE DUT'S CSR.
5826 030642 103036              JSR    PC,MSLGET      ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5827      BCC    50$       ;GO REPORT ERROR IF MR DID NOT CLEAR.
5828      ;+
5829      ; Verify the diagnostic fail bit is set, indicating the error.
5830      ; Report error if diagnostic fail bit is clear.
5831 030644 005267 153120              ; -
5832 030650 032714 020000              INC    ERRNBR          ;SET ERROR NUMBER TO 1304.
5833 030654 001425              BIT    #BIT13,(R4)    ;CHECK THE STATE OF THE DIAG_FAIL BIT.
5834      BEQ    10$       ;GO REPORT ERROR IF DIAG_FAIL BIT CLEAR.
5835      ;+
5836      ; Remove the 8 self test codes form the fifo, and verify that at least
5837      ; one is a Procl to ram error code (231).
5838 030656 005267 153106              ; -
5839 030662 012700 000010              INC    ERRNBR          ;SET ERROR NUMBER TO 1305.
5840 030666 005001              MOV    #8.,R0         ;SET MAXIMUM READ COUNT.
5841 030670 016704 151352              CLR    R1             ;CLEAR THE CORRECT CODE COUNTER.
5842 030674 011402              MOV    RBUFA,R4       ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
5843 030676 100020              6$: MOV    (R4),R2      ;READ A CODE FROM THE FIFO.
5844 030700 042702 007400              BPL    50$            ;GO REPORT ERROR IF THE FIFO IS EMPTY.
5845 030704 120227 170231              BIC    #7400,R2       ;REMOVE THE LINE NUMBER FROM THE CODE.
5846 030710 001001              CMPB   R2,#170231     ;IS IT THE CORRECT ERROR CODE?.
5847 030712 005201              BNE    8$            ;SKIP NEXT INSTRUCTION, IF NOT A 231 CODE.
5848 030714 005300              INC    R1             ;INCREMENT COUNTER.
5849 030716 001366              8$: DEC    R0         ;DECREMENT MAX READ COUNTER.
5850 030720 005701              BNE    6$            ;LOOP IF 8 CODES HAVE NOT BEEN READ.
5851 030722 001010              TST    R1             ;WERE ANY 231 CODES FOUND?.
5852 030724 005267 153040              BNE    60$           ;YES, THEN EXIT.
5853      INC    ERRNBR    ;NO, SET ERROR NUMBER TO 1306 AND REPORT ERROR.
5854 030730 012701 012012              10$: ;Report Self test error reporting bad.
5855 030734 104460              MOV    #EM1302,R1    ;SELECT ERROR MESSAGE.
5856 030736 000402              ERROR ;REPORT ERROR. >>>> ERROR <<<<
5857      BR    60$       ;EXIT THE TEST. TRAP C$ERROR
5858 030740 004767 171520              50$: JSR    PC,TSABRT  ;REPORT NON RELATED TEST ERROR.
5859
5860 030744 005067 151402              60$: CLR    CTRLCF    ;INDICATE THAT WE COMPLETED THE TEST.
5861
5862 030750              ENDTST
030750
030750 104401              L10042: TRAP C$ETST

```

HARDWARE TEST - STFAIL -

```

5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875 030752
      030752
5876      000016
5877 030752 012767 000016 151346
5878 030760 012767 177777 151364
5879 030766 012767 000001 152772
5880 030774 012767 012051 152770
5881 031002 012767 016470 152764
5882
5883
5884
5885
5886 031010 012701 005670
5887 031014 012702 000040
5888 031020 005003
5889 031022 016704 151216
5890 031026 004767 167306
5891 031032 103131
5892
5893
5894
5895 031034 010214
5896 031036 004767 171274
5897 031042 012701 005670
5898 031046 004767 167266
5899 031052 103121
5900
5901
5902
5903
5904
5905
5906
5907 031054 012705 000040
5908 031060 012703 000143
5909 031064 010304
5910 031066 012767 002571 152674
5911 031074 012701 012101
5912
5913 031100 017702 151142
5914 031104 100077
5915
5916
5917
5918 031106 012700 000301
5919 031112 040200

```

```

.SBTTL HARDWARE TEST - ROMVER -
;+ *****
;* - Rom Version Test -
;* This test verifies that the DUT's Self-Test places valid Rom version
;* numbers in the fifo after it has been skipped. The Rom version numbers
;* will be reported (on the first pass only), if an affirmative answer
;* was given to the software P-table question.
;+ *****
;-- BGNTST
;
; T14:
; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
; SET UP THE TEST NUMBER. (14)
; INDICATE THAT WE ARE WITHIN A TEST.
; SET ERROR TYPE AS FATAL IN ERROR TABLE.
; SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
; SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
;
; Wait up to 3 seconds for the DUT Master Reset bit to clear.
; If time-out occurs, then exit this test.
;--
; MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
; MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
; CLR R3 ;WAITING FOR BIT TO CLEAR.
; MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
; JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
; BCC 50$ ;ABORT THE TEST IF MR DID NOT CLEAR.
;+
; Set the Master Reset bit, and skip the self test.
;--
; MOV R2,(R4) ;SET THE MASTER RESET BIT.
; JSR PC,SKPSTS ;SKIP THE SELF TEST.
; MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
; JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
; BCC 50$ ;ABORT THE TEST IF MR DID NOT CLEAR.
;+
; Remove characters from the FIFO until either;
; (a) The FIFO is purged, go report the error.
; (b) The maximum try counter is zero, go report the error.
; (c) PROC_1's rom version number was found before PROC_2's, go report error.
; (d) Both ROM version numbers have been found.
;--
; MOV #4*NUMLNS,R5 ;SET MAXIMUM TRY COUNTER.
; MOV #99.,R3 ;SET AN INVALID ROM VERSION NUMBER FOR PROC 1.
; MOV R3,R4 ;SET AN INVALID ROM VERSION NUMBER FOR PROC_2.
; MOV #1401.,ERRNBR ;SET THE ERROR NUMBER TO 1401.
; MOV #EM1402,R1 ;SELECT MESSAGE TO BE REPORTED IF FIFO EMPTY.
;
; 2$: MOV @RBUFA,R2 ;READ THE NEXT CHAR FROM THE FIFO.
; BPL 12$ ;GO REPORT ERROR IF FIFO EMPTY.
;+
; Check if the read data is a BMP code.
;--
; MOV #301,R0 ;SET-UP A BIT MASK OF A BMP CODE.
; BIC R2,R0 ;TRY TO CLEAR THE BIT MASK WITH THE READ DATA.

```

HARDWARE TEST

- ROMVER -

```

5920 031114 001003          BNE      4$          ;BRANCH IF NOT A BMP CODE.
5921 031116 004767 171146 JSR      PC,SAVBMP   ;SAVE THE BMP CODE ON THE QUEUE.
5922 031122 000435          BR       8$          ;
5923
5924          ;+
5925          ; Check if the read data is a self-test code.
5926 031124 012700 000201 4$:      MOV      #201,R0   ;SET-UP A BIT MASK OF A SELFTEST CODE.
5927 031130 040200          BIC      R2,R0       ;TRY TO CLEAR THE BIT MASK WITH THE READ DATA.
5928 031132 001431          BEQ      8$          ;BRANCH IF IT IS A SELFTEST CODE.
5929
5930          ;+
5931          ; The read data is a ROM version number, determine which one it is.
5932
5933 031134 032702 000002          BIT      #BIT1,R2   ;CHECK THE PROCESSOR NUMBER BIT IN THE CODE.
5934 031140 001407          BEQ      6$          ;BRANCH IF IT IS PROC_1 ROM VERSION NUMBER.
5935 031142 010204          MOV      R2,R4       ;SAVE PROC_2 ROM VERSION NUMBER.
5936 031144 042704 177603          BIC      #177603,R4 ;CLEAR ANY UNWANTED BITS.
5937 031150 000241          CLC          ;CLEAR THE CARRY BIT.
5938 031152 006004          ROR      R4          ;SHIFT THE CODES ALONG TO GET THE ROM
5939 031154 006004          ROR      R4          ; VERSION NUMBER IN THE LOW 5 BITS.
5940 031156 000417          BR       8$          ;
5941 031160 010203          6$:      MOV      R2,R3       ;SAVE PROC_1 ROM VERSION NUMBER.
5942 031162 042703 177603          BIC      #177603,R3 ;CLEAR ANY UNWANTED BITS.
5943 031166 000241          CLC          ;CLEAR THE CARRY BIT.
5944 031170 006003          ROR      R3          ;SHIFT THE CODE ALONG TO GET THE ROM
5945 031172 006003          ROR      R3          ; VERSION NUMBER IN THE LOW 5 BITS.
5946 031174 020427 000143          CMP      R4,#99.    ;CHECK IF WE HAVE RECEIVE PROC_2 ROM CODE.
5947 031200 001016          BNE      10$         ;GO REPORT BOTH ROM VERSION NUMBERS.
5948
5949          ;+
5950          ; Received ROM version numbers out of sequence.
5951          ; ie, Proc_1's ROM version number found in the fifo before Proc_2's.
5952 031202 012701 012167          MOV      #EM1403,R1 ;SELECT THE ERROR MESSAGE TO BE REPORTED.
5953 031206 012767 002572 152554      MOV      #1402.,ERRNBR ;SET THE ERROR NUMBER.
5954 031214 000433          BR       12$        ;GO REPORT ERROR.
5955
5956 031216 005305          8$:      DEC      R5          ;DECREMENT THE MAX TRY COUNTER.
5957 031220 001327          BNE      2$          ;LOOP TO GET THE NEXT CHAR FROM THE FIFO.
5958 031222 012701 012242          MOV      #EM1404,R1 ;SELECT THE ERROR MESSAGE TO BE REPORTED.
5959 031226 012767 002573 152534      MOV      #1403.,ERRNBR ;SET THE ERROR NUMBER.
5960 031234 000423          BR       12$        ;GIVE UP, GO REPORT ERROR.
5961
5962          ;+
5963          ; If this is the first pass, and software P-table question was answered YES,
5964          ; then report the rom version numbers to the operator.
5965 031236 032767 000001 150762 10$:     BIT      #BIT0,OPTION ;CHECK ON THE STATE OF THE SOFTWARE SWITCH.
5966 031244 001431          BEQ      60$         ;EXIT IF NO ROM VERSION PRINTOUT WAS REQUESTED.
5967 031246 026727 151060 000001      CMP      PASCNT,#1  ;CHECK IF THIS IS THE FIRST PASS.
5968 031254 003025          BGT      60$         ;EXIT IF ROM VERS HAVE ALREADY BEEN REPORTED.
5969 031256          PRINTB #EF1401,R3,R4 ;PRINT THE ROM VERSION NUMBERS.
          MOV      R4,-(SP)
          MOV      R3,-(SP)
          MOV      #EF1401,(SP)
          MOV      #3,-(SP)
          MOV      SP,R0
          TRAP    C$PNTB
          ADD     #10,SP

```

HARDWARE TEST

- ROMVER -

```

5970 031302 000412          BR      60$          ;EXIT THIS TEST.
5971
5972          ;-
5973          ; Error reports:
5974 031304 012767 016562 152462 12$:  MOV    #ER1401,ERRBLK ;SELECT THE ERROR REPORTING ROUTINE.
5975 031312          ERROR          ;REPORT ERROR.          >>>>> ERROR <<<<<
5976 031314 104460          TRAP    C$ERROR
5977
5978 031316 012767 002575 152444 50$:  MOV    #1405.,ERRNBR ;SET UP ERROR NUMBER FOR TSABRT RTN.
5979 031324 004767 171134          JSR    PC,TSABRT    ;REPORT NON-TEST RELATED ERROR.
5980
5981 031330 005067 151016          60$:  CLR    CTRLCF      ;INDICATE THAT WE COMPLETED THE TEST.
5982
5983 031334          ENDTST
          L10043:
          TRAP    C$ETST
031334 104401

```

HARDWARE TEST

- REGWRW -

```

5985
5986
5987
5988
5989
5990
5991
5992
5993
5994 031336
      031336
5995      000017
5996 031336 012767 000017 150762
5997 031344 012767 177777 151000
5998 031352 012767 000001 152406
5999 031360 012767 003101 152402
6000 031366 012767 012445 152376
6001 031374 005067 151062
6002 031400 012700 002464
6003 031404 004767 166570
6004
6005
6006
6007
6008
6009 031410 004767 170042
6010 031414 103402
6011 031416 000167 000116
6012
6013
6014
6015 031422 005267 152342
6016 031426 012702 000017
6017 031432 016704 150606
6018 031436 010214
6019 031440 011401
6020 031442 042701 177760
6021 031446 020102
6022 031450 001406
6023
6024 031452 012767 016712 152314
6025 031460 005003
6026 031462 005005
6027 031464
      031464 104460
6028 031466 005302
6029 031470 002362
6030
6031
6032
6033
6034
6035 031472 005267 152272
6036 031476 005003
6037 031500 012704 000002
6038 031504 004767 167510
6039

```

```

.SBTTL HARDWARE TEST - REGWRW -
;+ *****
;* - Device Register Word Access Read and Write Test -
;*
;* This test verifies that the device registers can be read and written
;* correctly using word accesses.
;*
;-- *****

      BGNTST
      T15::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (16)
      MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
      MOV #1,ERRTYP ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
      MOV #1601,ERRNBR ;SET UP ERROR NUMBER IN THE ERROR TABLE.
      MOV #EM1604,ERRMSG ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
      CLR ERSRFR ;CLEAR THE ERROR SUMMARY FLAGS.
      MOV #ERCNTB,R0
      JSR PC,CLR16W ;CLEAR THE ERROR COUNTER TABLE.

;+
; Reset the DUT to a known state, do not remove the status codes from the fifo.
; Clear TX and RX interrupt enable bits in the CSR.
; This subroutine reports errors >>>> 1601 <<<<.
;-
      JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCS .+6 ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
      JMP 60$ ;YES, EXIT THE TEST.

;+
; Verify read/write capability to indirect address field of CSR
;-
      INC ERRNBR ;SET THE ERROR REPORT NUMBER TO 1602.
      MOV #17,R2 ;SET LOOP COUNT.
      MOV CSRA,R4 ;GET CSR ADDRESS.
2$: MOV R2,(R4) ;WRITE COUNT TO CSR.
      MOV (R4),R1 ;READ BACK THE CONTENTS OF THE CSR
      BIC #177760,R1 ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
      CMP R1,R2 ;CHECK FOR CORRECT DATA WRITTEN/READ.
      BEQ 4$ ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
      MOV #ER1601,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
      CLR R3 ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
      CLR R5 ;CAUSE REPORT OF LINE 0.
      ERROR ; >>>> ERROR # 1602 <<<<
      TRAP C$ERROR
4$: DEC R2 ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
      BGE 2$ ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.

;+
; Write and verify 16 data patterns in all used bits of all registers on all
; active lines. Before writing each pattern, clear all the bits.
; REGTST routine reports errors with numbers >>>> ERROR 1603 - 1605 <<<<.
;-
      INC ERRNBR ;SET THE ERROR NUMBER TO 1603.
      CLR R3 ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
      MOV #2,R4 ;INDICATE R/W ACCESS, CLEAR FIRST.
      JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
;+

```


HARDWARE TEST - REGWRM -

```

6057 .SBTTL HARDWARE TEST - REGWRM -
6058 ;* *****
6059 ;* - Device Register Word Access Read/Modify/Write Test -
6060 ;*
6061 ;* This test verifies that the device registers can be written correctly
6062 ;* using word read/modify/write accesses.
6063 ;*
6064 ;*
6065 ;* *****
6066 031546 BGNTST
        031546
6067 000020 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6068 031546 012767 000020 150552 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (17)
6069 031554 012767 177777 150570 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
6070 031562 012767 000001 152176 MOV #1,ERRTYP ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
6071 031570 012767 003245 152172 MOV #1701,ERRNBR ;SET UP ERROR NUMBER IN THE ERROR TABLE.
6072 031576 012767 012513 152166 MOV #EM1701,ERRMSG ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
6073 031604 005067 150652 CLR ERSMRF ;CLEAR THE ERROR SUMMARY FLAGS.
6074 031610 012700 002464 MOV #ERCNTB,R0
6075 031614 004767 166360 JSR PC,CLR16W ;CLEAR THE ERROR COUNTER TABLE.
6076
6077 ;*
6078 ;* Reset the DUT to a known state, do not remove the status codes from the fifo.
6079 ;* Clear TX and RX interrupt enable bits in the CSR.
6080 ;* This subroutine reports errors >>>> 1701 <<<<<.
6081 031620 004767 167632 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6082 031624 103402 BCS .+6 ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
6083 031626 000167 000122 JMP 60$ ;YES, EXIT THE TEST.
6084
6085 ;*
6086 ;* Verify read/modify/write capability to indirect address field of CSR
6087 031632 005267 152132 INC ERRNBR ;SET THE ERROR REPORT NUMBER TO 1702.
6088 031636 012702 000017 MOV #17,R2 ;SET LOOP COUNT.
6089 031642 016704 150376 MOV CSRA,R4 ;GET CSR ADDRESS.
6090 031646 042714 000017 2$: BIC #17,(R4) ;CLEAR THE DUT CSR USING READ/MODIFY/WRITE.
6091 031652 050214 BIS R2,(R4) ;WRITE COUNT TO CSR USING READ/MODIFY/WRITE.
6092 031654 011401 MOV (R4),R1 ;READ BACK THE CONTENTS OF THE CSR
6093 031656 042701 177760 BIC #177760,R1 ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
6094 031662 020102 CMP R1,R2 ;CHECK FOR CORRECT DATA WRITTEN/READ.
6095 031664 001406 BEQ 4$ ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
6096 ;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
6097 031666 012767 016712 152100 MOV #ER1601,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
6098 031674 005003 CLR R3 ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
6099 031676 005005 CLR R5 ;CAUSE REPORT OF LINE 0.
6100 031700 ERROR ; >>>> ERROR # 1702 <<<<<
        031700 104460 TRAP C$ERROR
6101 031702 005302 4$: DEC R2 ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
6102 031704 002360 BGE 2$ ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.
6103
6104 ;*
6105 ;* Write and verify 16 data patterns in all used bits of all registers on all
6106 ;* active lines using R/M/W. Before writing each pattern, clear all the bits.
6107 ;* REGTST routine reports errors with numbers >>>> ERROR 1703 - 1705 <<<<<.
6108 031706 005267 152056 INC ERRNBR ;SET THE ERROR NUMBER TO 1703.
6109 031712 005003 CLR R3 ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
6110 031714 012704 000001 MOV #1,R4 ;INDICATE R/M/W ACCESS, CLEAR FIRST.
6111 031720 004767 167274 JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.

```

HARDWARE TEST - REGWRM -

```

6112
6113      ;+
6114      ; Write and verify 16 data patterns in all used bits of all registers on all
6115      ; active lines using R/M/W. Before writing each pattern, set all the bits.
6116      ; REGTST routine reports errors with numbers >>>> ERROR 1706 - 1708 <<<<.
6117 031724 012767 003252 152036      ;-
6118      MOV    #1706.,ERRNBR      ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6119      CLR    R3                  ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
6120      NEG    R4                  ;INDICATE R/M/W ACCESS, SET FIRST.
6121      JSR    PC,REGTST           ;WRITE AND VERIFY DATA PATTERNS.
6122      ;+
6123      ; Print error summary reports if necessary.
6124      ; The following routine reports errors with number >>>> ERROR # 1709 <<<<
6125      ;-
6126      MOV    #1709.,ERRNBR      ;SET UP ERROR NUMBER FOR NEXT RTN.
6127      JSR    PC,REPSMR          ;REPORT ERROR SUMMARY IF NECESSARY.
6128 60$:   CLR    CTRLCF           ;INDICATE THAT WE COMPLETED THE TEST.
        ENDTST
        L10045:
                TRAP    C$ETST

```


HARDWARE TEST

- REGBRW -

```

6130 .SBTTL HARDWARE TEST - REGBRW -
6131 ;* *****
6132 ;* - Device Register Byte Access Read and Write Test -
6133 ;*
6134 ;* This test verifies that the device registers can be read and written
6135 ;* correctly using byte accesses.
6136 ;*
6137 ;*
6138 ;*
6139 ;* -- *****
        BGNTST
6140
6141 031762 000021          TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6142 031762 012767 000021 150336  MOV #TNUM,TSTNUM          ;SET UP THE TEST NUMBER. (18)
6143 031770 012767 177777 150354  MOV #-1,CTRLCF          ;INDICATE THAT WE ARE WITHIN A TEST.
6144 031776 012767 000001 151762  MOV #1,ERRTYP          ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
6145 032004 012767 003411 151756  MOV #1801.,ERRNBR      ;SET UP ERROR NUMBER IN THE ERROR TABLE.
6146 032012 012767 012570 151752  MOV #EM1801,ERRMSG     ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
6147 032020 005067 150436          CLR ERSMRF             ;CLEAR THE ERROR SUMMARY FLAGS.
6148 032024 012700 002464          MOV #ERCNTB,R0
6149 032030 004767 166144          JSR PC,CLR16W          ;CLEAR THE ERROR COUNTER TABLE.
6150
6151 ;*
6152 ;* Reset the DUT to a known state, do not remove the status codes from the fifo.
6153 ;* Clear TX and RX interrupt enable bits in the CSR.
6154 ;* This subroutine reports errors >>>> 1801 <<<<<.
6155 032034 004767 167416          JSR PC,RESETT         ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6156 032040 103402          BCS .+6             ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
6157 032042 000167 000152          JMP 60$             ;YES, EXIT THE TEST.
6158 032046 012767 003412 151714  MOV #1802.,ERRNBR     ;SET THE ERROR REPORT NUMBER TO 1802.
6159
6160 ;*
6161 ;* Verify read/write capability to indirect address field of CSR.
6162 ;* Use byte accesses.
6163 032054 012702 000017          MOV #17,R2           ;SET LOOP COUNT.
6164 032060 016704 150160          MOV CSRA,R4          ;GET CSR ADDRESS.
6165 032064 110214          2$: MOV R2,(R4)          ;WRITE COUNT TO CSR.
6166 032066 111401          MOV (R4),R1          ;READ BACK THE CONTENTS OF THE CSR
6167 032070 042701 177760          BIC #177760,R1      ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
6168 032074 020102          CMP R1,R2           ;CHECK FOR CORRECT DATA WRITTEN/READ.
6169 032076 001406          BEQ 4$             ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
6170 032100 012767 016712 151666  ;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
6171 032106 005003          MOV #ER1601,ERRBLK  ;SELECT THE PROPER ERROR REPORT ROUTINE.
6172 032110 005005          CLR R3             ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
6173 032112          CLR R5             ;CAUSE REPORT OF LINE 0.
6174 032114 005302          ERROR            ; >>>> ERROR # 1802 <<<<<
6175 032116 002362          4$: DEC R2          ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
6176          BGE 2$          ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.
6177
6178 ;*
6179 ;* Write and verify 16 data patterns in all used bits of all lower bytes of all
6180 ;* registers on all active lines. Use READ/WRITE accesses. Before writing
6181 ;* each pattern, clear all the used bits of all active registers.
6182 ;* REGTST routine reports errors with numbers >>>> ERROR 1803 - 1805 <<<<<.
6183 032120 005267 151644          INC ERRNBR          ;SET THE ERROR NUMBER TO 1803.
6184 032124 012703 177777          MOV #1,R3          ;INDICATE THAT LO BYTE ACCESSSES ARE TO BE USED.
6185 032130 012704 000002          MOV #2,R4          ;INDICATE R/W ACCESS, CLEAR FIRST.

```

HARDWARE TEST

- REGBRW -

```

6185 032134 004767 167060          JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
6186                                ;+
6187                                ; Write and verify 16 data patterns in all used bits of all high bytes of all
6188                                ; registers on all active lines. Use READ/WRITE accesses. Before writing
6189                                ; each pattern, clear all the used bits of all active registers.
6190                                ; REGTST routine reports errors with numbers >>>> ERROR 1806 - 1808 <<<<.
6191                                ;-
6192 032140 012767 003416 151622    MOV    #1806.,ERRNBR   ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6193 032146 005403                    NEG    R3              ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6194 032150 004767 167044          JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
6195                                ;+
6196                                ; Write and verify 16 data patterns in all used bits of all lower bytes of all
6197                                ; registers on all active lines. Use READ/WRITE accesses. Before writing
6198                                ; each pattern, set all the used bits of all active registers.
6199                                ; REGTST routine reports errors with numbers >>>> ERROR 1809 - 1811 <<<<.
6200                                ;-
6201 032154 012767 003421 151606    MOV    #1809.,ERRNBR   ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6202 032162 005403                    NEG    R3              ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
6203 032164 005404                    NEG    R4              ;INDICATE R/W ACCESS, SET FIRST.
6204 032166 004767 167026          JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
6205                                ;+
6206                                ; Write and verify 16 data patterns in all used bits of all high bytes of all
6207                                ; registers on all active lines. Use READ/WRITE accesses. Before writing
6208                                ; each pattern, set all the used bits of all active registers.
6209                                ; REGTST routine reports errors with numbers >>>> ERROR 1812 1814 <<<<.
6210                                ;-
6211 032172 012767 003424 151570    MOV    #1812.,ERRNBR   ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6212 032200 005403                    NEG    R3              ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6213 032202 004767 167012          JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
6214                                ;+
6215                                ; Print error summary reports if necessary.
6216                                ; The following routine reports errors with number >>>> ERROR # 1815 <<<<
6217                                ;
6218 032206 012767 003427 151554    MOV    #1815.,ERRNBR   ;SET UP ERROR NUMBER FOR NEXT RTN.
6219 032214 004767 167210          JSR    PC,REPSMR      ;REPORT ERROR SUMMARY IF NECESSARY.
6220 032220 005067 150126          60$: CLR    CTRLCF    ;INDICATE THAT WE COMPLETED THE TEST.
6221 032224                                ;
        032224                                ;
        032224 104401                                ;

```

L10046:

TRAP C\$ETST

HARDWARE TEST

- REGBRM -

```

6223
6224
6225
6226
6227
6228
6229
6230
6231
6232 032226
      032226
6233 000022
6234 032226 012767 000022 150072
6235 032234 012767 177777 150110
6236 032242 012767 000001 151516
6237 032250 012767 003555 151512
6238 032256 012767 012636 151506
6239 032264 005067 150172
6240 032270 012700 002464
6241 032274 004767 165700
6242
6243
6244
6245
6246
6247 032300 004767 167152
6248 032304 103402
6249 032306 000167 000156
6250 032312 012767 003556 151450
6251
6252
6253
6254
6255 032320 012702 000017
6256 032324 016704 147714
6257 032330 142714 000017
6258 032334 150214
6259 032336 111401
6260 032340 042701 177760
6261 032344 020102
6262 032346 001406
6263
6264 032350 012767 016712 151416
6265 032356 005003
6266 032360 005005
6267 032362
      032362 104460
6268 032364 005302
6269 032366 002360
6270
6271
6272
6273
6274
6275
6276 032370 005267 151374
6277 032374 012703 177777

```

```

.SBTTL HARDWARE TEST - REGBRM -
;* *****
;* - Device Register Byte Access Read/Modify/Write Test -
;*
;* This test verifies that the device registers can be read and written
;* correctly using byte accesses in Read/Modify/Write mode.
;*
;-- *****

      BGNTST
      T18::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (19)
      MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
      MOV #1,ERRTYP ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
      MOV #1901,ERRNBR ;SET UP ERROR NUMBER IN THE ERROR TABLE.
      MOV #EM1901,ERRMSG ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
      CLR ERSMRF ;CLEAR THE ERROR SUMMARY FLAGS.
      MOV #ERCNTB,R0
      JSR PC,CLR16W ;CLEAR THE ERROR COUNTER TABLE.

;*
; Reset the DUT to a known state, do not remove the status codes from the fifo.
; Clear TX and RX interrupt enable bits in the CSR.
; This subroutine reports errors >>>> 1901 <<<<<.
;
      JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCS .+6 ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
      JMP 60$ ;YES, EXIT THE TEST.
      MOV #1902,ERRNBR ;SET THE ERROR REPORT NUMBER TO 1902.

;*
; Verify read/write capability to indirect address field of CSR.
; Use byte accesses.
;
      MOV #17,R2 ;SET LOOP COUNT.
      MOV CSRA,R4 ;GET CSR ADDRESS.
2$: BICB #17,(R4) ;CLEAR THE DUT CSR USING READ/MODIFY/WRITE.
      BISB R2,(R4) ;WRITE COUNT TO CSR USING READ/MODIFY/WRITE.
      MOVB (R4),R1 ;READ BACK THE CONTENTS OF THE CSR
      BIC #177760,R1 ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
      CMP R1,R2 ;CHECK FOR CORRECT DATA WRITTEN/READ.
      BEQ 4$ ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
      MOV #ER1601,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
      CLR R3 ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
      CLR R5 ;CAUSE REPORT OF LINE 0.
      ERROR ; >>>> ERROR # 1902 <<<<<
      TRAP C$ERROR
4$: DEC R2 ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
      BGE 2$ ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.

;*
; Write and verify 16 data patterns in all used bits of all lower bytes of all
; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
; writing each pattern, clear all the used bits of all active registers.
; REGTST routine reports errors with numbers >>>> ERROR 1903 - 1905 <<<<<.
;
      INC ERRNBR ;SET THE ERROR NUMBER TO 1903.
      MOV #-1,R3 ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.

```

HARDWARE TEST

- REGBRM -

```

6278 032400 012704 000001      MOV #1,R4      ;INDICATE R/M/W ACCESS, CLEAR FIRST.
6279 032404 004767 166610      JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6280
6281 ;+
6282 ; Write and verify 16 data patterns in all used bits of all high bytes of all
6283 ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
6284 ; writing each pattern, clear all the used bits of all active registers.
6285 ; REGTST routine reports errors with numbers >>>> ERROR 1906 - 1908 <<<<.
6286 032410 012767 003562 151352      MOV #1906.,ERRNBR ;SET JP ERROR NUMBER FOR REGTST ROUTINE.
6287 032416 005403      NEG R3        ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6288 032420 004767 166574      JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6289
6290 ;+
6291 ; Write and verify 16 data patterns in all used bits of all lower bytes of all
6292 ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
6293 ; writing each pattern, set all the used bits of all active registers.
6294 ; REGTST routine reports errors with numbers >>>> ERROR 1909 - 1911 <<<<.
6295 032424 012767 003565 151336      MOV #1909.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6296 032432 005403      NEG R3        ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
6297 032434 005404      NEG R4        ;INDICATE R/M/W ACCESS, SET FIRST.
6298 032436 004767 166556      JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6299
6300 ;+
6301 ; Write and verify 16 data patterns in all used bits of all high bytes of all
6302 ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
6303 ; writing each pattern, set all the used bits of all active registers.
6304 ; REGTST routine reports errors with numbers >>>> ERROR 1912 - 1914 <<<<.
6305 032442 012767 003570 151320      MOV #1912.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6306 032450 005403      NEG R3        ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6307 032452 004767 166542      JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6308
6309 ;+
6310 ; Print error summary reports if necessary.
6311 ; The following routine reports errors with number >>>> ERROR # 1915 <<<<
6312 032456 012767 003573 151304      MOV #1915.,ERRNBR ;SET UP ERROR NUMBER FOR NEXT RTN.
6313 032464 004767 166740      JSR PC,REPSMR ;REPORT ERROR SUMMARY IF NECESSARY.
6314 032470 005067 147656      CLR CTRLCF    ;INDICATE THAT WE COMPLETED THE TEST.
6315 032474
        032474
        032474 104401

```

L10047: TRAP C\$ETST

HARDWARE TEST

- IDBIT -

```

6317 .SBTTL HARDWARE TEST - IDBIT -
6318 ;+ *****
6319 ;* - Device Register ID Bit Test -
6320 ;*
6321 ;* This test verifies that the DUT STAT register ID bit reads as clear.
6322 ;*
6323 ;-- *****
6324
6325 032476 BGNTST
        032476
6326 000023 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6327 032476 012767 000023 147622 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (20)
6328 032504 012767 177777 147640 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
6329 032512 012767 000001 151246 MOV #1,ERRTYP ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
6330 032520 012767 003721 151242 MOV #2001,ERRNBR ;SET UP ERROR NUMBER IN THE ERROR TABLE.
6331 032526 Q12767 012713 151236 MOV #EM2001,ERRMSG ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
6332
6333 ;+
6334 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
6335 ; Clear TX and RX interrupt enable bits in the CSR.
6336 ; This subroutine reports errors >>>> 2001 <<<<.
6337 032534 004767 166716 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6338 032540 103016 BCC 60$ ;FATAL RESET ERROR? YES, EXIT THE TEST.
6339
6340 ;+
6341 ; Read the STAT register ID bit and verify that it is clear.
6342 032542 017701 147504 MOV @STAT,R1 ;READ THE STAT REGISTER CONTENTS.
6343 032546 032701 000400 BIT #BIT8,R1 ;CHECK THE ID BIT.
6344 032552 001411 BEQ 60$ ;ID BIT CLEAR? YES, EXIT THE TEST.
6345 032554 012767 003722 151206 MOV #2002,ERRNBR ;NO, SET THE ERROR REPORT NUMBER TO 2002.
6346 032562 012701 012755 MOV #EM2002,R1 ;GET THE PROPER ERROR MESSAGE.
6347 032566 012767 016470 151200 MOV #ER0503,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
6348 032574 ERROR ;ERROR NUMBER >>>> 2002 <<<<
        032574 104460
6349 032576 005067 147550 60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST. TRAP C$ERROR
6350 032602 ENDTST
        032602
        032602 104401 L10050: TRAP C$ETST

```

HARDWARE TEST

- NOTXDV -

```

6352
6353
6354
6355
6356
6357
6358
6359
6360
6361 032604
      032604
6362
6363 032604 000024
6364 032612 012767 000024 147514
6365 032620 012767 177777 147532
6366 032626 012767 000001 151140
6367 032626 012767 004065 151134
6368 032634 012767 013030 151130
6368 032642 012767 017422 151124
6369
6370
6371
6372
6373
6374 032650 004767 165302
6375 032654 103054
6376 032656 005267 151106
6377
6378
6379
6380
6381
6382
6383 032662 016705 147350
6384 032666 012700 000200
6385 032672 004767 170634
6386 032676 012700 177670
6387 032702 004767 170700
6388 032706 012704 000012
6389 032712 004767 165362
6390 032716 004767 167654
6391
6392
6393
6394
6395
6396 032722 016705 147310
6397 032726 005004
6398 032730 000241
6399 032732 006005
6400 032734 103020
6401
6402
6403
6404
6405
6406 032736 010477 147302
6407 032742 012777 000012 147276

```

```

.SBTTL HARDWARE TEST          NOTXDV
;+ *****
;*          - No TX_DATA_VALID/No TX_ACTION Test -
;* This test verifies that if a data word is written without the
;* TX_DATA_VALID bit set, no TX_ACTION will be generated.
;* To ensure data is not accidentally transmitted, the test is performed
;* in internal loopback, and on all active lines.
;+ *****
;-- BGNTST
                                T20:
TNUM == TNUM + 1                ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV #TNUM,TSTNUM                ;SET UP THE TEST NUMBER. (21)
MOV #-1,CTRLCF                  ;INDICATE THAT WE ARE IN A TEST.
MOV #1,ERRTYP                    ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV #2101,ERRNBR                 ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
MOV #EM2101,ERRMSG               ;SET ERROR MESSAGE ADDRESS IN ERRTABL.
MOV #ER9101,ERRBLK              ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
;+
; Reset the DUT to a known state, remove the status codes from the fifo.
; Clear TX and RX interrupt enable bits in the CSR.
; This subroutine reports error >>>> 2101 <<<<<.
;--
JSR PC,CLNRST                    ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
BCC 60$                          ;RESET FAILURE?, ABORT THIS TEST.
INC ERRNBR                        ;SET THE ERROR NUMBER TO 2102.
;+
; Set internal loopback on all active lines.
; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
; 2 stop bits.
; Disable transmitters on all active lines.
;--
MOV ACTLNS,R5                    ;PASS THE ACTIVE LINE BIT MAP.
MOV #200,R0                      ;PASS THE LNCTRL CONTENTS.
JSR PC,WTWLNC                    ;INITIALISE THE LNCTRL REGISTERS.
MOV #177670,R0                  ;PASS THE LPR CONTENTS.
JSR PC,WTWLP                    ;INITIALISE THE LPR REGISTERS ON ALL LINES.
MOV #10,R4                      ;PASS DELAY TIME OF 10 MILLI SECS.
JSR PC,DELAY                    ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
JSR PC,TXDSBL                   ;DISABLE TRANSMITTERS ON ALL ACTIVE LINES.
;+
; Test all active lines individually.
; Write a data word to the TXCHAR register with TX_DATA_VALID clear.
; Verify no TX_ACTION is generated.
;--
MOV ACTLNS,R5                    ;GET THE ACTIVE LINE BIT MAP.
CLR R4                          ;CLEAR THE LINE NUMBER COUNTER.
2$: CLC                          ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
ROR R5                          ;SHIFT THE BIT MAP INTO THE CARRY BIT.
BCC 4$                          ;DO NOT TEST THE LINE IF IT IS INACTIVE.
;+
; Select the line under test.
; Write data word (ASCII <LF>) to TXCHR register with the most significant
; bit (TX_DATA_VALID) clear.
;--
MOV R4,@CSRA                    ;SELECT THE LINE CURRENTLY UNDER TEST.
MOV #12,@TXCHA                  ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.

```

HARDWARE TEST

- NOTXDV -

```

6408
6409
6410
6411
6412 032750 012701 170002
6413 032754 016702 147264
6414 032760 004767 170276
6415 032764 103004
6416
6417 032766 010401
6418 032770 012702 013073
6419
6420 032774
      032774 104460
6421
6422
6423
6424 032776 005204
6425 033000 005705
6426 033002 001352
6427 033004 000400
6428
6429 033006 005067 147340
6430 033012
      033012
      033012 104401

```

```

;+
; Wait for a TX_ACTION to be returned, report error if TX_ACTION found
; before time-out occurs.
;-
MOV #170002,R1 ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
MOV CSRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.
JSR PC,WAIBIS ;WAIT FOR TX_ACTION TO COME BACK.
BCC 4$ ;SKIP ERROR REPORT IF TX-ACTION NOT FOUND.

MOV R4,R1 ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
MOV #EM2102,R2 ;PASS THE ERROR MESSAGE TO BE REPORTED.
ERROR ;"TX_ACT FOUND AFTER INVALID DATA WORD WRITTEN"
; >>>> ERROR #4102 <<<<<.
; TRAP C$ERROR

;+
; Verify all active lines have been tested.
;-
4$: INC R4 ;INCREMENT THE LINE NUMBER COUNTER.
TST R5 ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
BNE 2$ ;YES; BRANCH TO TEST THE NEXT LINE.
BR 60$ ;NO; EXIT THIS TEST.

60$: CLR CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.
ENDTST
L10051: TRAP C$ETST

```

HARDWARE TEST

- TXDVAL -

```

6432 .SBTTL HARDWARE TEST - TXDVAL -
6433 ;+ *****
6434 ;* - TX_DATA_VALID/TX_ACTION Test -
6435 ;* This test verifies that if a data word is written to the TXCHAR register
6436 ;* with the TX_DATA_VALID bit set, a corresponding TX_ACTION will be
6437 ;* generated.
6438 ;* To ensure data is not accidentally transmitted, the test is performed
6439 ;* in internal loopback, and on all active lines.
6440 ;*
6441 ;-- *****
6442
6443 033014 BGNTST
6444 033014 T21::
6445 033014 000025 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6446 033022 012767 000025 147304 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (22)
6447 033022 012767 177777 147322 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
6448 033030 012767 000001 150730 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
6449 033036 012767 004231 150724 MOV #2201.,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
6450 033044 012767 013167 150720 MOV #EM2201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRTBL.
6451 033052 012767 017422 150714 MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
6452 ;+
6453 ; Reset the DUT to a known state, remove the status codes from the fifo.
6454 ; Clear TX and RX interrupt enable bits in the CSR.
6455 ; This subroutine reports error >>>> 2201 <<<<.
6456 033060 004767 165072 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6457 033064 103066 BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.
6458 ;+
6459 ; Set internal loopback on all active lines.
6460 ; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
6461 ; 2 stop bits.
6462 ; Disable transmitters on all active lines.
6463 ;--
6464 03306b 016705 147144 MOV ACTLNS,R5 ;PASS THE ACTIVE LINE BIT MAP.
6465 033072 012700 000200 MOV #200,R0 ;PASS THE LNCTRL CONTENTS.
6466 033076 004767 170430 JSR PC,WTWLNLC ;INITIALISE THE LNCTRL REGISTERS.
6467 033102 012700 177670 MOV #177670,R0 ;PASS THE LPR CONTENTS.
6468 033106 004767 170474 JSR PC,WTWLPR ;INITIALISE THE LPR REGISTERS ON ALL LINES.
6469 033112 012704 000012 MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI-SECONDS.
6470 033116 004767 165156 JSR PC,DELAY ;WAIT FOR LNCTR AND LPR REGS TO BE UPDATED.
6471 033122 004767 167450 JSR PC,TXDSBL ;DISABLE TRANSMITTERS ON ALL ACTIVE LINES.
6472 ;+
6473 ; Test all active lines individually.
6474 ; Write a data word to the TXCHAR register with TX_DATA_VALID set.
6475 ; Verify that a corresponding TX_ACTION is generated.
6476 ;--
6477 033126 016705 147104 MOV ACTLNS,R5 ;GET THE ACTIVE LINE BIT MAP.
6478 033132 005004 CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
6479 033134 012767 004232 150626 2$: MOV #2202.,ERRNBR ;SET THE ERROR NUMBER TO 2202.
6480 033142 000241 CLC ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
6481 033144 006005 ROR R5 ;SHIFT THE BIT MAP INTO THE CARRY BIT.
6482 033146 103032 BCC 8$ ;DO NOT TEST THE LINE IF IT IS INACTIVE.
6483 ;+
6484 ; Select the line under test.
6485 ; Write data word (ASCII <LF>) to TXCHR register with the most significant
6486 ; bit (TX_DATA_VALID) set.
6487 ;

```


HARDWARE TEST

- TXDVAL -

```

6488 033150 010477 147070      MOV    R4,@CSRA      ;SELECT THE LINE CURRENTLY UNDER TEST.
6489 033154 012777 100012 147064  MOV    #100012,@TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
6490
6491                               ;+
6492                               ; Wait for a TX_ACTION to be returned, report error if no TX_ACTION
6493                               ; found before time-out occurs.
6494 033162 012701 170002      MOV    #170002,R1    ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
6495 033166 016702 147052      MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6496 033172 004767 170064      JSR    PC,WAIBIS    ;WAIT FOR TX_ACTION TO COME BACK.
6497 033176 103403      BCS    4$          ;SKIP ERROR REPORT IF TX-ACTION FOUND.
6498 033200 012702 013224      MOV    #EM2202,R2   ;PASS THE ERROR MESSAGE TO BE REPORTED.
6499                               ;"NO TX_ACT FOUND AFTER VALID DATA WORD TX'D".
6500 033204 000411      BR     6$          ;GO REPORT THE ERROR.
6501
6502                               ;+
6503                               ; Verify TX_ACTION returned from correct line.
6504 033206 005267 150556      4$:   INC    ERRNBR    ;INCREMENT ERROR NUMBER TO 2103.
6505 033212 000302      SWAB   R2          ;GET THE LINE NUMBER IN THE LOW BYTE.
6506 033214 042702 177760      BIC    #177760,R2  ;CLEAR THE UNWANTED BITS.
6507 033220 020204      CMP    R2,R4       ;IS IT THE CORRECT LINE NUMBER?.
6508 033222 001404      BEQ    8$          ;YES; SKIP THE ERROR REPORT.
6509 033224 012702 013316      MOV    #EM2203,R2  ;PASS THE ERROR MESSAGE TO BE REPORTED.
6510                               ; "INCORRECT LINE # RETURNED WITH TX_ACT"
6511 033230 010401      6$:   MOV    R4,R1    ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
6512 033232 104460      ERROR  ;          >>>> ERROR <<<<<.
6513                               TRAP   C$ERROR
6514
6515                               ;+
6516                               ; Verify all active lines have been tested.
6517 033234 005204      8$:   INC    R4          ;INCREMENT THE LINE NUMBER COUNTER.
6518 033236 005705      TST   R5          ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
6519 033240 001335      BNE   2$          ;YES; BRANCH TO TEST THE NEXT LINE.
6520
6521 033242 005067 147104      60$:  CLR    CTRLCF     ;INDICATE THAT WE ARE NOT WITHIN A TEST.
6522 033246
        033246
        033246 104401
                               L10052:
                               TRAP   C$ETST
    
```

HARDWARE TEST

- TXENBI-

```

6524
6525
6526
6527
6528
6529
6530
6531
6532
6533 033250
      033250
6534      000026
6535 033250 012767 000026 147050
6536 033256 012767 177777 147066
6537 033264 012767 000001 150474
6538 033272 012767 004375 150470
6539 033300 012767 013423 150464
6540 033306 012767 017422 150460
6541
6542
6543
6544
6545
6546 033314 004767 164636
6547 033320 103110
6548
6549
6550
6551
6552
6553
6554 033322 016705 146710
6555 033326 012700 000200
6556 033332 004767 170174
6557 033336 012700 177670
6558 033342 004767 170240
6559 033346 012704 000012
6560 033352 004767 164722
6561 033356 012705 000377
6562 033362 004767 167304
6563
6564
6565
6566
6567 033366 012703 000001
6568 033372 005004
6569 033374 012767 004376 150366 24:
6570 033402 030367 146630
6571 033406 001447
6572
6573
6574
6575
6576
6577 033410 010305
6578 033412 004767 167160
6579 033416 010477 146622

```

```

.SBTTL HARDWARE TEST - TXENBI-
; * *****
; * - TX_ENABLE (Inactive) Test -
; * This test verifies that when the line under test's TX_ENABLE bit is
; * clear, transmission will not take place on that line.
; * This test is performed in internal loopback, and on all active lines.
; *
; -- *****

BGNTST
T22::
      TNUM = TNUM + 1 ; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM, TSTNUM ; SET UP THE TEST NUMBER. (23)
      MOV #-1, CTRLCF ; INDICATE THAT WE ARE IN A TEST.
      MOV #1, ERRNBR ; SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV #2301, ERRMSG ; SET THE FIRST ERROR NUMBER IN ERROR TABLE.
      MOV #EM2301, ERRMSG ; SET ERROR MESSAGE ADDRESS IN ERRMSG.
      MOV #ER9101, ERRBLK ; SELECT THE CORRECT ERROR REPORTING ROUTINE.

; *
; * Reset the DUT to a known state, remove the status codes from the fifo.
; * Clear TX and RX interrupt enable bits in the CSR.
; * This subroutine reports error >>>> 2301 <<<<.
; --
      JSR PC, CLNRST ; RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCC 60$ ; RESET FAILURE?, ABORT THIS TEST.

; *
; * Set internal loopback on all active lines.
; * Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
; * 2 stop bits.
; * Enable transmitters on all lines.
; --
      MOV ACTLNS, R5 ; PASS THE ACTIVE LINE BIT MAP.
      MOV #200, R0 ; PASS THE LNCTRL CONTENTS.
      JSR PC, WTLNCR ; INITIALISE THE LNCTRL REGISTERS.
      MOV #177670, R0 ; PASS THE LPR CONTENTS.
      JSR PC, WTLNCR ; INITIALISE THE LPR REGISTERS ON ALL LINES.
      MOV #10, R4 ; PASS DELAY TIME OF 10 MILLI-SECONDS.
      JSR PC, DELAY ; WAIT FOR LNCTR AND LPR REGS TO BE UPDATED.
      MOV #MAPLNS, R5 ; PASS THE BIT MAP CORRESPONDING TO ALL LINES.
      JSR PC, TXENBL ; ENABLE TRANSMITTERS ON ALL LINES.

; *
; * Test all active lines individually.
; * Disable transmission on each active line.
; --
      MOV #1, R3 ; SET UP THE LINE BIT MAP FOR CHANNEL 0.
      CLR R4 ; CLEAR THE LINE NUMBER COUNTER.
      MOV #2302, ERRNBR ; SET THE ERROR NUMBER TO 2302.
      BIT R3, ACTLNS ; CHECK IF THE LINE IS ACTIVE.
      BEQ 64$ ; SKIP TESTING THIS LINE IF IT IS INACTIVE.

; *
; * Clear the TX_ENABLE bit in TBUFAD2 register.
; * Select the line under test.
; * Verify it is clear, report error if set.
; --
      MOV R3, R5 ; PASS THE BIT MAP OF THE LINE UNDER TEST.
      JSR PC, TXDSBL ; DISABLE TRANSMISSION ON THE LINE UNDER TEST.
      MOV R4, @CSRA ; SELECT THE LINE CURRENTLY UNDER TEST.

```

HARDWARE TEST

- TXENBI-

```

6580 033422 005777 146632      TST    @TXAD2A      ;VERIFY THE TX_ENABLE BIT IS SET.
6581 033426 100433              BMI    4$           ;GO REPORT ERROR IF TX_ENABLE BIT SET.
6582                          ;*
6583                          ; Write data word (ASCII <LF>) to TXCHR register.
6584                          ; Wait for a TX_ACTION to be returned, report error if a TX_ACTION
6585                          ; is found before time-out occurs.
6586                          ;-
6587 033430 012767 004377 150332  MOV    #2303.,ERRNBR ;SET ERROR NUMBER TO 2303.
6588 033436 012777 100012 146602  MOV    #100012,@TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
6589 033444 012701 170002              MOV    #170002,R1    ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
6590 033450 016702 146570              MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6591 033454 004767 167602              JSR    PC,WAIBIS    ;WAIT FOR TX ACTION TO COME BACK.
6592 033460 103016              BCC    4$           ;GO REPORT ERROR IF NO TX-ACTION FOUND.
6593                          ;*
6594                          ; Wait for the data to appear in the fifo, report error if data found.
6595                          ;-
6596 033462 005267 150302      INC    ERRNBR      ;SET ERROR NUMBER TO 2304.
6597 033466 012701 070012      MOV    #70012,R1   ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
6598 033472 016702 146546      MOV    CSRA,R2    ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6599 033476 004767 167560      JSR    PC,WAIBIS  ;WAIT FOR RX DATA AVAILABLE TO SET.
6600 033502 103405              BCS    4$         ;REPORT ERROR IF DATA RECEIVED IN THE FIFO.
6601 033504 005267 150260      INC    ERRNBR     ;SET ERROR NUMBER TO 2305.
6602 033510 017702 146532      MOV    @RBUFA,R2  ;READ THE DATA FROM THE FIFO.
6603 033514 100004              BPL    6$         ;SKIP ERROR REPORT IF DATA IS THERE.
6604                          ;*
6605 033516 010401              4$:    MOV    R4,R1   ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
6606 033520 012702 013461      MOV    #EM2302,R2 ;PASS THE MESSAGE TO BE REPORTED.
6607                          ; "TX_ENABLE BIT BAD ON LINE: nn".
6608 033524              ERROR      ; >>>> ERROR <<<<<.
6609 033524 104460              TRAP    C$ERROR
6610                          ;*
6611                          ; Verify all active lines have been tested.
6612 033526 000241              6$:    CLC          ;CLEAR THE CARRY BIT PRIOR TO ROTATION.
6613 033530 006103              ROL    R3          ;SHIFT THE BIT MAP FOR THE NEXT LINE.
6614 033532 005204              INC    R4          ;INCREMENT THE LINE NUMBER COUNTER.
6615 033534 020427 000010      CMP    R4,#NUMLNS ;HAVE ALL THE LINES BEEN TESTED?.
6616 033540 002715              BLT    2$         ;NO; BRANCH TO TEST THE NEXT LINE.
6617                          ;*
6618 033542 005067 146604      60$:   CLR    CTRLCF    ;INDICATE THAT WE ARE NOT WITHIN A TEST.
6619 033546              ENDTST
        033546
        033546 104401              L10053:
        TRAP    C$ETST

```

HARDWARE TEST

- TXENBA

```

6621
6622
6623
6624
6625
6626
6627
6628
6629
6630 033550
      033550
6631      000027
6632 033550 012767 000027 146550
6633 033556 012767 177777 146566
6634 033564 012767 000001 150174
6635 033572 012767 004541 150170
6636 033600 012767 013517 150164
6637 033606 012767 017422 150160
6638
6639
6640
6641
6642
6643 033614 004767 164336
6644 033620 103127
6645
6646
6647
6648
6649
6650
6651 033622 016705 146410
6652 033626 012700 000200
6653 033632 004767 167674
6654 033636 012700 177670
6655 033642 004767 167740
6656 033646 012704 000012
6657 033652 004767 164422
6658 033656 012705 000377
6659 033662 004767 166710
6660
6661
6662
6663
6664 033666 012703 000001
6665 033672 005004
6666 033674 012767 004542 150066 2$:
6667 033702 030367 146330
6668 033706 001463
6669
6670
6671
6672
6673
6674 033710 010305
6675 033712 004767 166754
6676 033716 012705 000012

```

```

.SBTTL HARDWARE TEST - TXENBA-
;* *****
;* - TX_ENABLE (Active) Test -
;* This test verifies that when the TX_ENABLE bit is set in the appropriate
;* line register, transmission will take place on that line.
;* This test is performed in internal loopback, and on all active lines.
;* *****
;-- *****

      BGNTST
                                         T23::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (24)
      MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
      MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV #2401,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
      MOV #EM2401,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRTABL.
      MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
;*
; Reset the DUT to a known state, remove the status codes from the fifo.
; Clear TX and RX interrupt enable bits in the CSR.
; This subroutine reports error >>>> 2401 <<<<.
;
      JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.
;*
; Set internal loopback on all active lines.
; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
; 2 stop bits.
; Disable transmitters on all lines.
;--
      MOV ACTLNS,R5 ;PASS THE ACTIVE LINE BIT MAP.
      MOV #200,R0 ;PASS THE LNCTRL CONTENTS.
      JSR PC,WTWLNC ;INITIALISE THE LNCTRL REGISTERS.
      MOV #177670,R0 ;PASS THE LPR CONTENTS.
      JSR PC,WTWLPR ;INITIALISE THE LPR REGISTERS ON ALL LINES.
      MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI-SECONDS.
      JSR PC,DELAY ;WAIT FOR LNCTR AND LPR REGS TO BE UPDATED.
      MOV #MAPLNS,R5 ;PASS THE BIT MAP CORRESPONDING TO ALL LINES.
      JSR PC,TXDSBL ;DISABLE TRANSMITTERS ON ALL LINES.
;*
; Test all active lines individually.
; Enable transmission on each active line.
;--
      MOV #1,R3 ;SET UP THE LINE BIT MAP FOR CHANNEL 0.
      CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
      MOV #2402.,ERRNBR ;SET THE ERROR NUMBER TO 2402.
      BIT R3,ACTLNS ;CHECK IF THE LINE IS ACTIVE.
      BEQ 8$ ;SKIP TESTING HIS LINE IF IT IS INACTIVE.
;*
; Select the line under test.
; Set the TX_ENABLE bit in TBUFFAD2 register.
; Verify it is set, report error if clear.
;
      MOV R3,R5 ;PASS THE BIT MAP OF THE LINE UNDER TEST.
      JSR PC,TXENBL ;ENABLE TRANSMISSION ON THE LINE UNDER TEST.
      MOV #10.,R5 ;SET TXCHAR/LOOP COUNT TO 10.

```

HARDWARE TEST

- TXENBA-

```

6677 033722 010477 146316      MOV    R4,@CSRA      ;SELECT THE LINE CURRENTLY UNDER TEST.
6678 033726 005777 146326      TST    @TXAD2A      ;VERIFY THE TX_ENABLE BIT IS SET.
6679 033732 100045          BPL    6$           ;GO REPORT ERROR IF TX_ENABLE BIT CLEAR.
6680
6681      ;+
6682      ; Write data word (ASCII <LF>) to TXCHR register.
6683      ; Wait for a TX_ACTION to be returned, report error if no TX_ACTION
6684      ; found before time-out occurs.
6685 033734 012767 004543 150026 4$:  MOV    #2403.,ERRNBR ;SET ERROR NUMBER TO 2403.
6686 033742 012777 100012 146276      MOV    #100012,@TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
6687 033750 012701 170002          MOV    #170002,R1    ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
6688 033754 016702 146264          MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6689 033760 004767 167276          JSR    PC,WAIBIS    ;WAIT FOR TX ACTION TO COME BACK.
6690 033764 103030          BCC    6$           ;GO REPORT ERROR IF NO TX-ACTION FOUND.
6691
6692      ;+
6693      ; Wait for the data to appear in the fifo, report error if time-out.
6694 033766 005267 147776          INC    ERRNBR      ;SET ERROR NUMBER TO 2404.
6695 033772 012701 070012          MOV    #70012,R1   ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
6696 033776 016702 146242          MOV    CSRA,R2    ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6697 034002 004767 167254          JSR    PC,WAIBIS    ;WAIT FOR RX DATA AVAILABLE TO SET.
6698 034006 103017          BCC    6$           ;REPORT ERROR IF NO DATA RECEIVED IN THE FIFO.
6699 034010 005267 147754          INC    ERRNBR      ;SET ERROR NUMBER TO 2405.
6700 034014 017702 146226          MOV    @RBUFA,R2   ;READ THE DATA FROM THE FIFO.
6701 034020 100012          BPL    6$           ;GO REPORT ERROR IF THER IS'NT ANY DATA THERE.
6702 034022 005267 147742          INC    ERRNBR      ;SET ERROR NUMBER TO 2406.
6703 034026 000302          SWAB   R2          ;PUT THE LINE NUMBER IN THE LOW BYTE.
6704 034030 042702 177760          BIC    #177760,R2  ;CLEAR THE UNWANTED BITS.
6705 034034 020204          CMP    R2,R4      ;DID THE DATA COME FROM THE CORRECT LINE?.
6706 034036 001003          BNE    6$         ;NO; GO REPORT THE ERROR.
6707 034040 005305          DEC    R5         ;DECREMENT THE TXCHAR/LOOP COUNTER.
6708 034042 001334          BNE    4$         ;LOOP TO TX THE NEXT CHAR.
6709 034044 000404          BR    8$         ;GO TEST THE NEXT LINE.
6710
6711 034046 010401          6$:  MOV    R4,R1      ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
6712 034050 012702 013461          MOV    #EM2302,R2  ;PASS THE MESSAGE TO BE REPORTED.
6713
6714 034054          ERROR          ; "TX_ENABLE BIT BAD ON LINE: nn".
6715          034054 104460          ; >>>> ERROR <<<<<.
6716          ;+
6717          ; Verify all active lines have been tested.
6718 034056 010305          8$:  MOV    R3,R5      ;PASS THE BIT MAP OF THE LINE UNDER TEST.
6719 034060 004767 166512          JSR    PC,TXDSBL   ;CLEAR THE TX_ENABLE BIT ON THIS LINE.
6720 034064 000241          CLC          ;CLEAR THE CARRY BIT PRIOR TO ROTATION.
6721 034066 006103          ROL    R3         ;SHIFT THE BIT MAP FOR THE NEXT LINE.
6722 034070 005204          INC    R4         ;INCREMENT THE LINE NUMBER COUNTER.
6723 034072 020427 000010          CMP    R4,#NUMLNS ;HAVE ALL THE LINES BEEN TESTED?.
6724 034076 002676          BLT    2$         ;NO; BRANCH TO TEST THE NEXT LINE.
6725
6726 034100 005067 146246          60$: CLR    CTRLCF     ;INDICATE THAT WE ARE NOT WITHIN A TEST.
6727 034104          ENDTST
        034104
        034104 104401          L10054:
                                TRAP    C$ETST

```

HARDWARE TEST - INTA -

```

6729 .SBTTL HARDWARE TEST - INTA -
6730 ;* *****
6731 ;* - Interrupt Test -
6732 ;* This test verifies that the Device Under Test (DUT) will generate
6733 ;* reception and transmission interrupts correctly. This test does
6734 ;* not depend on the use of the serial line transmission or reception
6735 ;* capabilities of the DUT. The lines are put in internal loopback
6736 ;* to minimize any external effects that could be caused on devices
6737 ;* attached to the serial lines.
6738 ;*
6739 ;*
6740 ;-- *****
6741 034106 BGNTST
6742 034106 T24::
6743 000030 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6744 034106 012767 000030 146212 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (26)
6745 034114 012767 177777 146230 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
6746 034122 012767 000001 147636 MOV #1,ERRTYP ;SET ERROR FATAL ERROR TYPE IN ERROR TABLE.
6747 034130 012767 003101 147632 MOV #1601.,ERRNBR ;SET FIRST ERROR REPORT NUMBER IN ERROR TABLE.
6748 034136 012767 013553 147626 MOV #EM2601.,ERRMSG ;SET TEST ERROR MESSAGE IN ERROR TABLE.
6749 ;*
6750 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
6751 ; Clear TX and RX interrupt enable bits in the CSR.
6752 ; This subroutine reports errors from >>>> 2601 thru 2602 <<<<<.
6753 ;
6754 034144 004767 165306 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6755 034150 103402 BCS 2$ ;SKIP AROUND ABORTING TEST IF NO ERROR FOUND.
6756 034152 000167 000746 JMP 60$ ;ABORT TEST IF FATAL ERROR FOUND DURING RESET.
6757 034156 012767 005053 147604 2$: MOV #2603.,ERRNBR ;SET THE ERROR REPORT NUMBER TO 2603.
6758 ;*
6759 ; Enable transmitters on all lines.
6760 034164 012705 000377 4$: MOV #MAPLNS,R5 ;PASS ACTIVE LINE BIT MAP.
6761 034170 004767 166476 JSR PC,TXENBL ;ENABLE TRANSMISSION ON ALL LINES.
6762 ;*
6763 ; Test reception interrupts.
6764 ; Set up for RX and TX interrupts:
6765 ; RX interrupt service routine inputs a char and counts the interrupt.
6766 ; TX interrupt service routine counts TX interrupts.
6767 ;-
6768 034174 005067 146134 CLR RXINTC ;CLEAR THE RX INTERRUPT COUNTER.
6769 034200 005067 146132 CLR RXINTF ;CLEAR THE RX INTERRUPT FLAGS.
6770 034204 005067 146130 CLR TXINTC ;CLEAR THE TX INTERRUPT COUNTER.
6771 034210 012767 002726 146106 MOV #BUFBAS,BUFPTR ;LOAD THE BUFFER PTR WITH THE BUFFER BASE ADR.
6772 034216 012700 000240 SETPRI #PRI05 ;DISABLE DEVICE INTERRUPTS.
6773 034222 104441 MOV #PRI05,R0
6774 034224 SETVEC RXVECA,#RXINPT,#PRI05 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
6775 034224 012746 000240 TRAP C$SPRI
6776 034230 012746 024052 MOV #PRI05,-(SP)
6777 034234 016746 145772 MOV #RXINPT,(SP)
6778 034240 012746 000003 MOV RXVECA,-(SP)
6779 034244 104437 MOV #3,-(SP)
6780 034246 062706 000010 TRAP C$SVEC
6781 034252 SETVEC TXVECA,#CACHTX,#PRI05 ;SET UP INTERRUPT VECTOR TO CATCH TX INT.
6782 034252 012746 000240 ADD #10,SP
6783 034256 012746 023664 MOV #PRI05,-(SP)
6784 MOV #CACHTX,-(SP)

```

HARDWARE TEST - INTA -

```

034262 016746 145746
034266 012746 000003
034272 104437
034274 062706 000010
6775 034300          SETPRI #PRI00          ;ALLOW INTERRUPTS.
034300 012700 000000
034304 104441
6776
6777
6778
6779
6780
6781 034306 004767 165732
6782 034312 012704 000004
6783 034316 004767 163756
6784 034322 004767 165656
6785
6786
6787
6788
6789 034326 005767 146002
6790 034332 001017
6791
6792
6793
6794 034334 012701 013762
6795 034340 032777 000200 145676
6796 034346 001416
6797 034350 012701 013674
6798 034354 032777 100000 145664
6799 034362 001410
6800 034364 012701 013603
6801 034370 000405
6802
6803
6804
6805 034372 005767 145740
6806 034376 100006
6807 034400 012701 014056
6808
6809
6810
6811 034404
034404 104455
034406 005053
034410 013553
034412 016470
6812
6813
6814
6815 034414 016702 145720
6816 034420 001406
6817
6818 034422 012701 010003
6819 034426
034426 104455
034430 005054

```

```

MOV TXVECA, (SP)
MOV #3, -(SP)
TRAP C$SVEC
ADD #10, SP
MOV #PRI00, RO
TRAP C$SPRI
;+
;Enable reception interrupts.
;Delay 4 ms to allow time for the interrupts to take place.
;Disable reception interrupts.
;-
JSR PC, RXIE1 ;ENABLE THE RECEPTION INTERRUPTS.
MOV #4, R4 ;PASS 4 MS COUNT TO THE DELAY ROUTINE.
JSR PC, DELAY ;DELAY 4 MILLI-SECONDS.
JSR PC, RXIE0 ;DISABLE RECEPTION INTERRUPTS.
;+
; Verify that the correct interrupts took place.
; Test the int counter to verify that interrupts took place.
;-
TST RXINTC ;CHECK THE RX INTERRUPT COUNT.
BNE 6$ ;SKIP THE FOLLOWING ERRORS IF COUNT <> 0.
;+
; Determine reason for no RX interrupts and print proper error message.
;-
MOV #EM2604, R1 ;SET UP MSG IN CASE "RX.DATA.AVAIL IS CLR".
BIT #BIT7, @CSRA ;TEST THE RX.DATA.AVAIL BIT OF THE CSR.
BEQ 8$ ;GO REPORT ERROR IF RX.DATA.AVAIL IS CLR.
MOV #EM2603, R1 ;SET UP MSG IN CASE "DATA.VALID IS CLEAR".
BIT #BIT15, @RBUFA ;TEST THE DATA.VALID BIT OF THE FIFO.
BEQ 8$ ;GO REPORT ERROR IF DATA.VALID IS NOT CLEAR.
MOV #EM2602, R1 ;SET UP MSG, "DATA.VALID IS CLEAR".
BR 8$ ;GO REPORT THE ERROR.
;+
; If RX ints occurred with RX.DATA.AVAIL clear, report the error.
;-
6$: TST RXINTF ;CHECK THE RX INTERRUPT FLAGS.
BPL 10$ ;SKIP THE ERROR IF FLAG IS CLEAR.
MOV #EM2605, R1 ;SET UP THE PROPER MESSAGE.
;+
; Report the error which has been found.
;-
8$: ERRDF 2603, EM2601, ER0503; >>>> ERROR #2603 <<<<<.
TRAP C$ERDF
.WORD 2603
.WORD EM2601
.WORD ER0503
;+
; Verify that no TX interrupts have been generated so far in this test.
;-
10$: MOV TXINTC, R2 ;LOAD # OF TX INTERRUPTS FOR ER0504 RTN.
BEQ 12$ ;SKIP ERROR IF NO TX INTERRUPTS.
;REPORT "TX INTERRUPTS(S) RECEIVED WITH TX INTERRUPTS DISABLED."
MOV #EM0526, R1 ;SET UP MESSAGE ADR FOR INDIRECT PRINT.
ERRDF 2604, EM2601, ER0504; >>>> ERROR #2604 <<<<<.
TRAP C$ERDF
.WORD 2604

```

HARDWARE TEST

- INTA -

SEQ 0175

```

034432 013553
034434 016514
6820
6821
6822
6823 034436 012700 000240
034436 012700 000240
034442 104441
6824 034444
034444 016700 145562
034450 104436
6825 034452
034452 016700 145556
034456 104436
6826
6827
6828
6829
6830
6831
6832 034460 005067 145650
6833 034464 005067 145650
6834 034470 005067 145646
6835 034474
034474 012746 000240
034500 012746 023636
034504 016746 145522
034510 012746 000003
034514 104437
034516 062706 000010
6836 034522
034522 012746 000240
034526 012746 024160
034532 016746 145476
034536 012746 000003
034542 104437
034544 062706 000010
6837 034550
034550 012700 000000
034554 104441
6838
6839
6840
6841 034556 012705 000022
6842 034562 012701 000144
6843 034566 012702 100000
6844 034572 016704 145446
6845 034576 012703 100000
6846 034602 004767 163646
6847 034606 103020
6848 034610 005003
6849 034612 004767 163636
6850 034616 103005
6851 034620 005305
6852 034622 001365
6853
6854 034624 012701 014200

```

```

;+
; Clean out the interrupt vectors used in this test.
;--
12$:  SETPRI  #PRI05          ;DISABLE DEVICE INTERRUPTS.
                                MOV  #PRI05,R0
                                TRAP C$SPRI
                                MOV  RXVECA,R0
                                TRAP C$CVEC
                                MOV  TXVECA,R0
                                TRAP C$CVEC
;+
; Test transmission interrupts.
; Set up for RX and TX interrupts:
; RX interrupt service routine counts RX interrupts.
; TX interrupt service routine counts the interrupt and sets flags.
;--
        CLR  RXINTC          ;CLEAR THE RX INTERRUPT COUNTER.
        CLR  TXINTC          ;CLEAR THE TX INTERRUPT COUNTER.
        CLR  TXINTF          ;CLEAR THE RX INTERRUPT FLAGS.
        SETVEC RXVECA,#CACHRX,#PRI05 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
                                MOV  #PRI05, (SP)
                                MOV  #CACHRX, -(SP)
                                MOV  RXVECA, -(SP)
                                MOV  #3, -(SP)
                                TRAP C$SVEC
                                ADD  #10, SP
        SETVEC TXVECA,#TXINTR,#PRI05 ;SET UP INT VECTOR TO TX INT ROUTINE.
                                MOV  #PRI05, -(SP)
                                MOV  #TXINTR, -(SP)
                                MOV  TXVECA, (SP)
                                MOV  #3, -(SP)
                                TRAP C$SVEC
                                ADD  #10, SP
        SETPRI  #PRI00          ;ALLOW INTERRUPTS.
                                MOV  #PRI00,R0
                                TRAP C$SPRI
;+
; Verify that the TX_ACTION bit is clear.
;--
        MOV  #18.,R5          ;INITIALIZE THE LOOP COUNTER.
        MOV  #100.,R1         ;SET 100 MS TIME-OUT.
        MOV  #BIT15,R2        ;SELECT TX_ACTION BIT TO TEST.
        MOV  CSRA,R4          ;PASS OUT CSR AS THE WORD TO TEST.
14$:  MOV  #BIT15,R3          ;WAIT FOR TX_ACTION TO BE SET.
        JSR  PC,MSLOOP        ;WAIT UP TO 100 MS FOR TX_ACTION SET.
        BCC 20$              ;IF TIME-OUT, CONSIDER TX_ACTION CLEAR.
        CLR  R3               ;NOW, WAIT FOR TX_ACTION CLEAR.
        JSR  PC,MSLOOP        ;WAIT UP TO 100 MS FOR TX_ACTION CLEAR.
        BCC 16$              ;IF TIME-OUT, REPORT TX_ACTION WON'T CLEAR.
        DEC  R5               ;DECREMENT THE TX_ACTION SET COUNTER.
        BNE 14$              ;LOOP IF NOT TOO MANY TX_ACTIONS FOUND.
;REPORT "TX_ACTION SET REPEATEDLY AFTER RESET, NO DATA SENT."
        MOV  #EM2607,R1      ;SELECT ERROR MESSAGE.

```


HARDWARE TEST

- INTA -

```

6855 034630 000402          BR      18$          ;GO TO REPORT THE ERROR.
6856 034632 012701 014274 16$:   MOV      #EM2608,R1    ;SELECT TX_ACTION STUCK SET MSG.
6857 034636 104455          18$:   ERRDF    2605,EM2606,ER0503; >>>>> ERROR #2605 <<<<<.
        034636 104455          TRAP    C$ERDF
        034640 005055          .WORD  2605
        034642 014141          .WORD  EM2606
        034644 016470          .WORD  ER0503
6858 034646 000424          BR      24$          ;GO TO TEST WITH TX_ACTION SET.
6859
6860          ;+
6861          ; Verify that no interrupts occur with TX_ACTION clear.
6862 034650 004767 166152 20$:   JSR      PC,TXIE1    ;ENABLE TX_INTERRUPTS.
6863 034654 012704 000062          MOV      #50.,R4     ;PASS 50 MS TIME TO THE DELAY ROUTINE.
6864 034660 004767 163414          JSR      PC,DELAY    ;DELAY 50 MILLI-SECONDS TO ALLOW INTS TO OCCUR.
6865 034664 005767 145450          TST     TXINTC      ;TEST THE TX INTERRUPT COUNT.
6866 034670 001413          BEQ     24$         ;SKIP THE ERROR IF NO TX INTERRUPTS.
6867 034672 012701 014200          MOV      #EM2607,R1  ;SELECT MESSAGE IN CASE TX INT FLAG CLEAR.
6868 034676 005767 145440          TST     TXINTF      ;TEST THE TX INTERRUPT FLAGS.
6869 034702 100002          BPL     22$         ;GO REPORT ERROR IF TX FLAG IS CLEAR.
6870 034704 012701 014345          MOV      #EM2609,R1  ;TX FLAG IS SET, SELECT PROPER ERROR MESSAGE.
6871          ;REPORT "TRANSMIT INTERRUPT TEST ERROR:..."
6872 034710 104455          22$:   ERRDF    2606,EM2606,ER0503; >>>>> ERROR #2606 <<<<<.
        034710 104455          TRAP    C$ERDF
        034712 005056          .WORD  2606
        034714 014141          .WORD  EM2606
        034716 016470          .WORD  ER0503
6873
6874          ;+
6875          ; Prepare TX interrupt counter and flags.
6876 034720 005067 145414 24$:   CLR     TXINTC      ;CLEAR THE TX INTERRUPT COUNT.
6877 034724 005067 145412          CLR     TXINTF      ;CLEAR THE TX INTERRUPT FLAGS.
6878
6879          ;+
6880          ; Set up line parameters for transmission.
6881 034730 012705 000377          MOV     #MAPLNS,R5   ;PASS ACTIVE LINES BIT MAP.
6882 034734 012700 000200          MOV     #200,R0      ;PASS INERT STATE, INTERNAL LOOPBACK.
6883 034740 004767 166566          JSR     PC,WTWLNC    ;DISABLE RECPTION AND DMA, ETC. ON DUT.
6884 034744 012700 156430          MOV     #156430,R0   ;SPECIFY 9600BPS,1STOP,NO PARITY,8BITS/CHAR.
6885 034750 004767 166632          JSR     PC,WTWLPR    ;WRITE TO ALL LPR REGISTERS.
6886
6887          ;+
6888          ; Send a null char to each line.
6889 034754 016701 145266          MOV     TXCHA,R1     ;SET UP TXCHAR REGISTER ADDRESS.
6890 034760 012702 100000          MOV     #100000,R2   ;SET CHARACTER TO BE TRANSMITTED = NULL.
6891 034764 004767 166572          JSR     PC,WTWLNS    ;SEND NULL CHAR TO EACH LINE.
6892
6893          ;+
6894          ; Delay 250 milli-seconds to allow interrupts to occur.
6895 034770 012704 000372          MOV     #250.,R4    ;SET UP FOR 250 MS DELAY.
6896 034774 004767 163300          JSR     PC,DELAY     ;WAIT 250 MS.
6897
6898          ;+
6899          ; Verify that TX interrupts occurred.
6900 035000 005767 145334          TST     TXINTC      ;CHECK THE TX INTERRUPT COUNTER.
6901 035004 001007          BNE     26$         ;SKIP THE FOLLOWING ERROR IF WE GOT TX INTS.
6902
6903          ;+
        ; Determine the reason that we received no interrupts.

```

HARDWARE TEST

- INTA -

```

6904
6905 035006 012701 014424      ;-   MOV    #EM2610,R1      ;SET UP MSG IN CASE "TX_ACTION IS SET".
6906 035012 005777 145226      TST    @CSRA             ;CHECK THE DUT CSR.
6907 035016 100407             BMI    28$              ;GO TO REPORT ERROR IF TX_ACTION IS SET.
6908 035020 012701 014516      MOV    #EM2611,R1      ;SET UP "TX_ACTION NOT SET" MESSAGE.
6909
6910      ;+
6911      ; Check to verify that TX_ACTION was set for each interrupt.
6912 035024 005767 145312      26$:  TST    TXINTF        ;CHECK THE TX INTERRUPT FLAGS.
6913 035030 100006             BPL    30$              ;SKIP ERROR IF TX_ACTION CLR FLAG IS CLEAR.
6914 035032 012701 014345      MOV    #EM2609,R1      ;SET UP TX INT WITH "TX_ACTION CLR" MSG.
6915
6916      ;+
6917      ; Report "TRANSMIT INTERRUPT TEST ERROR:...."
6918 035036 28$:  ERRDF  2607,EM2606,ER0503;      >>>> ERROR #2607 <<<<<.
        035036 104455             TRAP   C$ERDF
        035040 005057             .WORD 2607
        035042 014141             .WORD EM2606
        035044 016470             .WORD ER0503
6919
6920      ;+
6921      ; Verify that no TX interrupts have been generated so far in this test.
6922 035046 016702 145262      30$:  MOV    RXINTC,R2    ;LOAD # OF RX INTERRUPTS FOR ER0504 RTN.
6923 035052 001406             BEQ    32$              ;SKIP ERROR IF NO RX INTERRUPTS.
6924 035054 012701 007713      MOV    #EM0525,R1      ;SET UP MESSAGE ADR FOR INDIRECT PRINT.
6925      ;REPORT "RX INTERRUPTS(S) RECEIVED WITH RX INTERRUPTS DISABLED."
6926 035060      ERRDF  2608,EM2606,ER0504;      >>>> ERROR #2608 <<<<<.
        035060 104455             TRAP   C$ERDF
        035062 005060             .WORD 2608
        035064 014141             .WORD EM2606
        035066 016514             .WORD ER0504
6927
6928      ;+
6929      ; Disable interrupts and clean out the interrupt vectors used in this test.
6930 035070 32$:  CLR    R1              ;CLEAR BOTH TRANSMITTER
6931 035072 004767 165670      JSR    PC,TXIEO        ; INTERRUPT ENABLE AND RECEIVER
6932 035076 004767 165102      JSR    PC,RXIEO        ; INTERRUPT ENABLE BITS IN THE DUT CSR.
6933 035102      SETPRI #PRI05        ;DISABLE DEVICE INTERRUPTS.
        035102 012700 000240             MOV    #PRI05,RO
        035106 104441             TRAP  C$SPRI
6934 035110      CLRVEC RXVECA        ;RETURN RX INT VECTOR TO UNUSED POOL.
        035110 016700 145116             MOV    RXVECA,RO
        035114 104436             TRAP  C$CVEC
6935 035116      CLRVEC TXVECA        ;RETURN TX INT VECTOR TO UNUSED POOL.
        035116 016700 145112             MOV    TXVECA,RO
        035122 104436             TRAP  C$CVEC
6936
6937 035124 005067 145222      60$:  CLR    CTRLCF        ;INDICATE THAT WE ARE NOT WITHIN A TEST.
6938
6939 035130      ENDTST
        035130             L10055:
        035130 104401             TRAP  C$ETST
6940

```

HARDWARE TEST

- BRLEVA -

```

6942 .SBTTL HARDWARE TEST - BRLEVA -
6943 ;* *****
6944 ;* - BR Level Test B -
6945 ;* This test verifies that the Device Under Test (DUT) will generate
6946 ;* reception and transmission interrupts at the correct BR level.
6947 ;* This test does not depend on the use of the serial line transmission
6948 ;* or reception capabilities of the DUT. The lines are put in internal
6949 ;* loopback to minimize any external effects that could be caused on
6950 ;* devices attached to the serial lines.
6951 ;*
6952 ;*
6953 ;* - *****
6954 035132 BGNTST
035132
6955 000031 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6956 035132 012767 000031 145166 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (30)
6957 035140 012767 177777 145204 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
6958 035146 012767 000001 146612 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
6959 035154 012767 005671 146606 MOV #3001.,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
6960 035162 012767 014601 146602 MOV #EM3001,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRtbl.
6961 035170 005067 145266 CLR ERSMRF ;INITIALIZE THE "REPORT ERROR SUMMARY" FLAGS.
6962 ;*
6963 ;* Reset the DUT to a known state, do not remove the status codes from the fifo.
6964 ;* Clear TX and RX interrupt enable bits in the CSR.
6965 ;* This subroutine reports errors from >>>> 3001 thru 3002 <<<<<.
6966 ;
6967 035174 004767 164256 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6968 035200 103402 BCS 2$ ;SKIP AROUND ABORTING TEST IF NO ERROR FOUND.
6969 035202 000167 000572 JMP 60$ ;ABORT TEST IF FATAL ERROR FOUND DURING RESET.
6970 035206 012767 005673 146554 2$: MOV #3003.,ERRNBR ;SET THE ERROR REPORT NUMBER TO 3003.
6971 ;*
6972 ;* Enable transmitters on all lines.
6973 ;*
6974 035214 012705 000377 4$: MOV #MAPLNS,R5 ;PASS ACTIVE LINE BIT MAP.
6975 035220 004767 165446 JSR PC,TXENBL ;ENABLE TRANSMISSION ON ALL LINES.
6976 ;*
6977 ;* Generate a transmission interrupt request.
6978 ;* Processor priority should be at 7 disabling ints.
6979 ;*
6980 035224 SETPRI #PRI07 ;DISABLE ALL INTERRUPTS.
035224 012700 000340 MOV #PRI07,R0
035230 104441 TRAP C$SPRI
6981 035232 SETVEC TXVECA,#TXINTR,#PRI07 ;SET UP INTERRUPT VECTOR TO CATCH TX INT.
035232 012746 000340 MOV #PRI07,-(SP)
035236 012746 024160 MOV #TXINTR,-(SP)
035242 016746 144766 MOV TXVECA,-(SP)
035246 012746 000003 MOV #3,-(SP)
035252 104437 TRAP C$SVEC
035254 062706 000010 ADD #10,SP
6982 ;*
6983 ;* Set up DUT for transmission interrupts:
6984 ;* Set up internal loopback.
6985 ;* Set up line parameters for transmission.
6986 ;*
6987 035260 012705 000377 MOV #MAPLNS,R5 ;PASS ACTIVE LINES BIT MASK.
6988 035264 012700 000200 MOV #200,R0 ;PASS INERT STATE, INTERNAL LOOPBACK.
6989 035270 004767 166236 JSR PC,WTWLCN ;DISABLE RECPTION AND DMA, ETC. ON DUT.

```

HARDWARE TEST

- BRLEVA -

```

6990 035274 012700 156430      MOV    #156430,R0      ;SPECIFY 9600BPS,1STOP,NO PARITY,8BITS/CHAR.
6991 035300 004767 166302      JSR    PC,WTWLP      ;WRITE INTO ALL LPR REGISTERS.
6992
6993      ;+
6994      ; Send a null char to each line.
6995 035304 016701 144736      MOV    TXCHA,R1      ;SET UP TXCHAR REGISTER ADDRESS.
6996 035310 012702 100000      MOV    #100000,R2    ;SET CHARACTER TO BE TRANSMITTED = NULL.
6997 035314 004767 166242      JSR    PC,WTWLN      ;SEND NULL CHAR TO EACH LINE.
6998
6999      ;+
7000      ; Delay 50 ms to allow time for the interrupt to be generated.
7001 035320 012704 000062      MOV    #50.,R4      ;PASS 50 MS TIME TO THE DELAY ROUTINE.
7002 035324 004767 162750      JSR    PC,DELAY      ;DELAY 50 MILLI-SECONDS.
7003
7004      ;+
7005      ; Generate a reception interrupt request.
7006 035330      SETVEC  RXVECA,#RXBRRT,#PRI07 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
        MOV    #PRI07,-(SP)
        MOV    #RXBRRT,-(SP)
        MOV    RXVECA,-(SP)
        MOV    #3,-(SP)
        TRAP   C$SVEC
        ADD    #10,SP
7007
7008      ;+
7009      ; Set up for the loop which tests the interrupt BR levels.
7010 035356 012705 000340      MOV    #340,R5      ;SET UP THE PRIORITY LEVEL TO 7.
7011 035362 005003      CLR    R3           ;CLEAR THE RX PRIORITY STORE AND FLAGS.
7012 035364 005002      CLR    R2           ;CLEAR THE TX PRIORITY STORE AND FLAGS.
7013
7014      ;+
7015      ; Enable TX and RX interrupts.
7016      ; Processor priority should be at 7 disabling the interrupts.
7017 035366 004767 164652      JSR    PC,RXIE1     ;ENABLE RECEIVER INTERRUPTS.
7018 035372 004767 165430      JSR    PC,TXIE1     ;ENABLE TRANSMITTER INTERRUPTS.
7019
7020      ;+
7021      ; Loop, lowering the processor priority until the DUT interrupts on RX and TX.
7022 035376 005067 144736      6$:   CLR    TXINTC      ;CLEAR THE TX INTERRUPT COUNTER.
7023 035402 005067 144734      CLR    TXINTF      ;CLEAR THE TX INTERRUPT FLAGS.
7024 035406 005067 144722      CLR    RXINTC      ;CLEAR THE RX INTERRUPT COUNTER.
7025 035412 005067 144720      CLR    RXINTF      ;CLEAR THE RX INTERRUPT FLAGS.
7026 035416      SETPRI  R5         ;SET PROCESSOR PRIORITY TO THE SELECTED VALUE.
        MOV    R5,R0
        TRAP   C$SPRI
7027 035422 012704 000001      MOV    #1,R4        ;PASS 1 MS COUNT TO THE DELAY ROUTINE.
7028 035426 004767 162646      JSR    PC,DELAY     ;DELAY 1 MS TO ALLOW INTERRUPTS TO OCCUR.
7029
7030      ;+
7031      ; Determine if any RX DUT interrupts occurred.
7032      ; Log the processor priority for the RX interrupt if first RX int.
7033 035432 005767 144676      TST    RXINTC      ;CHECK THE RECEIVE INTERRUPT COUNTER.
7034 035436 001412      BEQ    8$          ;SKIP THE PRIORITY LOG IF NO RX INT OCCURRED.
7035
7036      ;+
7037      ; If this is the first RX interrupt, log the priority.
7038 035440 005703      TST    R3          ;CHECK THE RX PRIORITY STORE AND FLAGS.

```

HARDWARE TEST

- BRLEVA -

```

7039 035442 001010          BNE      8$          ;GOTO TEST FOR TX INTS IF NOT THE FIRST RX INT.
7040 035444 010503          MOV      R5,R3       ;LOG THE PRESENT PRIORITY IN THE RX PRIO STORE.
7041 035446 052703 100000   BIS      #BIT15,R3   ;SET THE RX INT HAS OCCURRED FLAG.
7042 035452 016700 144660   MOV      RXINTF,R0   ;GET THE RX INTERRUPT ROUTINE FLAGS.
7043 035456 042700 137777   BIC      #137777,R0  ;CLEAR ALL BUT THE TX INT ERROR FLAG.
7044 035462 050003          BIS      R0,R3       ;IF TX INT ERROR, SET BIT 14 OF THE PRIO FLAGS.
7045
7046          ;+
7046          ; Determine if any TX DUT interrupts have occurred.
7047          ; Log the present processor priority if this is the first TX interrupt.
7048          ;-
7049 035464 005767 144650   8$:      TST      TXINTC          ;CHECK THE TRANSMIT INTERRUPT COUNTER.
7050 035470 001405          BEQ      10$          ;SKIP THE PRIORITY LOG IF NO TX INT OCCURRED.
7051          ;+
7052          ; If this is the first TX interrupt, log the priority.
7053          ;-
7054 035472 005702          TST      R2           ;CHECK THE TX PRIORITY STORE AND FLAGS.
7055 035474 100403          BMI      10$          ;SKIP THE LOGGING IF NOT FIRST TX INTERRUPT.
7056 035476 010502          MOV      R5,R2       ;LOG THE PRESENT PRIORITY IN THE TX PRIO STORE.
7057 035500 052702 100000   BIS      #BIT15,R2   ;SET THE TX INT HAS OCCURRED FLAG.
7058
7059          ;+
7059          ; Select next processor priority.
7060          ; Test for both RX and TX interrupts having occurred, loop if not.
7061          ;-
7062 035504 162705 000040   10$:     SUB      #40,R5          ;DECREMENT PRIORITY LEVEL BY ONE.
7063 035510 002402          BLT      12$          ;GOTO CHECK FOR ERRORS IF BELOW PRIORITY ZERO.
7064 035512 030203          BIT      R2,R3       ;AND PRIO FLAGS TOGETHER, ALTER NONE OF THEM.
7065 035514 100330          BPL      6$           ;LOOP IF RX AND TX INTS HAVEN'T BOTH OCCURRED.
7066          ;+
7067          ; Disable interrupts and clear interrupt vectors.
7068          ;-
7069 035516          12$:     SETPRI  #PRI07          ;DISABLE ALL INTERRUPTS.
7069 035516 012700 000340          MOV      #PRI07,R0
7069 035522 104441          TRAP    C$SPRI
7070 035524          CLRVEC  RXVECA          ;RETURN RX INT VECTOR TO UNUSED POOL.
7070 035524 016700 144502          MOV      RXVECA,R0
7070 035530 104436          TRAP    C$CVEC
7071 035532          CLRVEC  TXVECA          ;RETURN TX INT VECTOR TO UNUSED POOL.
7071 035532 016700 144476          MOV      TXVECA,R0
7071 035536 104436          TRAP    C$CVEC
7072
7073          ;+
7073          ; Verify that RX and TX interrupts occurred,
7074          ; at the proper BR level, and
7075          ; in the proper order.
7076          ; Determine if TX interrupt occurred.
7077          ;-
7078 035540 005702          TST      R2           ;DETERMINE WHETHER TX INT OCCURRED OR NOT.
7079 035542 100414          BMI      16$          ;SKIP THESE ERRORS IF TX INT OCCURRED.
7080          ;+
7081          ; Determine reason that no TX int occurred.
7082          ;-
7083 035544 012701 014424          MOV      #EM2610,R1  ;SELECT "NO TX INT FROM TX.ACTION" MESSAGE.
7084 035550 005777 144470          TST      @CSRA       ;CHECK THE TX.ACTION BIT OF THE DUT CSR.
7085 035554 100402          BMI      14$          ;SKIP TX.ACTION CLR MSG SELECTION IF IT IS SET.
7086 035556 012701 014516          MOV      #EM2611,R1  ;SELECT "TX.ACTION CLEAR AFTER CHARS SENT" MSG.
7087          ;REPORT "INTERRUPT BR LEVEL TEST ERROR:"
7088 035562          14$:     ERRDF   3003,EM3001,ER0503; >>>> ERROR #3003 <<<<<.
7088 035562 104455          TRAP    C$ERDF

```

HARDWARE TEST

- BRLEVA -

```

035564 005673                                .WORD 3003
035566 014601                                .WORD EM3001
035570 016470                                .WORD ER0503
7089 035572 000423                BR      18$          ;SKIP THE BR LEVEL CHECK, NO TX INT OCCURRED.
7090
7091                ;+
7092                ; Verify that the TX interrupt was at the proper BR level.
7093 035574 010204                16$:  MOV      R2,R4          ;CALCULATE THE BR LEVEL
7094 035576 042704 177400        BIC      #177400,R4      ; THAT THE TRANSMIT
7095 035602 006204                ASR      R4              ; INTERRUPT WAS
7096 035604 006204                ASR      R4              ; REQUESTED AT, WHICH
7097 035606 006204                ASR      R4              ; IS ONE GREATER THAN
7098 035610 006204                ASR      R4              ; THE PROCESSOR PRIORITY
7099 035612 006204                ASR      R4              ; LEVEL AT WHICH THE
7100 035614 005204                INC      R4              ; TRANSMIT INTERRUPT OCCURRED.
7101 035616 116705 144420        MOVB     BRLEVL,R5       ;GET THE EXPECTED INTERRUPT BR LEVEL.
7102 035622 120405                CMPB     R4,R5          ;COMARE THE INTERRUPT BR LEVEL WITH EXPECTED.
7103 035624 001406                BEQ      18$            ;SKIP THE ERROR IF BR LEVEL IS CORRECT.
7104                ;REPORT "TX INTERRUPT GENERATED AT WRONG BR LEVEL: ..."
7105 035626 012701 014726        MOV      #EM3003,R1     ;SELECT THE ERROR MESSAGE FOR THE ERROR CALL.
7106 035632                ERRDF    3004,EM3001,ER3001; >>>> ERROR #3004 <<<<<.
035632 104455                                TRAP     C$ERDF
035634 005674                                .WORD 3004
035636 014601                                .WORD EM3001
035640 017070                                .WORD ER3001
7107
7108                ;+
7109                ; Determine if RX interrupt occurred.
7110 035642 005703                18$:  TST      R3          ;CHECK THE RX INT OCCURRED FLAG.
7111 035644 100415                BMI      22$            ;SKIP THESE ERRORS IF RX INT OCCURRED.
7112
7113                ;+
7114                ; Determine reason that no RX int occurred.
7115 035646 012701 013603        MOV      #EM2602,R1     ;SELECT "NO RX INT FROM TX ACTION" MSG.
7116 035652 032777 000200 144364  BIT      #BIT7,@CSRA    ;CHECK THE RX.DATA.AVAIL BIT OF THE DUT CSR.
7117 035660 001002                BNE      20$            ;SKIP RX.DATA.AVAIL CLR MSG IF BIT IS SET.
7118 035662 012701 014632        MOV      #EM3002,R1     ;SELECT "NO RX.DATA.AVAIL AFTER RESET" MSG.
7119                ;REPORT "INTERRUPT BR LEVEL TEST ERROR:"
7120 035666                20$:  ERRDF    3005,EM3001,ER0503; >>>> ERROR #3005 <<<<<.
035666 104455                                TRAP     C$ERDF
035670 005675                                .WORD 3005
035672 014601                                .WORD EM3001
035674 016470                                .WORD ER0503
7121 035676 000423                BR      24$          ;SKIP THE BR CHECK IF NO RX INT OCCURRED.
7122
7123                ;+
7124                ; Verify that the RX interrupt was at the proper BR level.
7125 035700 010304                22$:  MOV      R3,R4          ;CALCULATE THE BR LEVEL
7126 035702 042704 177400        BIC      #177400,R4      ; THAT THE RECEIVE
7127 035706 006204                ASR      R4              ; INTERRUPT WAS
7128 035710 006204                ASR      R4              ; REQUESTED AT, WHICH
7129 035712 006204                ASR      R4              ; IS ONE GREATER THAN
7130 035714 006204                ASR      R4              ; THE PROCESSOR PRIORITY
7131 035716 006204                ASR      R4              ; LEVEL AT WHICH THE
7132 035720 005204                INC      R4              ; RECEIVE INTERRUPT OCCURRED.
7133 035722 116705 144314        MOVB     BRLEVL,R5       ;GET THE EXPECTED INTERRUPT BR LEVEL.
7134 035726 120405                CMPB     R4,R5          ;COMARE THE INTERRUPT BR LEVEL WITH EXPECTED.

```

HARDWARE TEST

- BRLEVA -

```

7135 035730 001406          BEQ      24$          ;SKIP THE ERROR IF BR LEVEL IS CORRECT.
7136                          ;REPORT "RX INTERRUPT GENERATED AT WRONG BR LEVEL: ..."
7137 035732 012701 015002    MOV      #EM3004,R1      ;SELECT ERROR MESSAGE FOR THE ERROR CALL.
7138 035736          ERRDF  3006,EM3001,ER3001; >>>> ERROR #3006 <<<<<.
                                TRAP      C$ERDF
                                .WORD    3006
                                .WORD    EM3001
                                .WORD    ER3001
7139                          ;+
7140                          ; Test for interrupts occuring in the proper order.
7141                          ;-
7142                          ;-
7143 035746 032703 040000    24$:    BIT      #BIT14,R3      ;CHECK THE IMPROPER INT ORDER ERROR FLAG.
7144 035752 001406          BEQ      26$          ;SKIP ERROR REPORT IF ERROR DID NOT OCCUR.
7145                          ;REPORT "TX INTERRUPT GIVEN PRECEDENCE OVER SIMULTANEOUS RX INT."
7146 035754 012701 015056    MOV      #EM3005,R1      ;SELECT THE ERROR MESSAGE FOR INDIRECT PRINT.
7147 035760          ERRDF  3007,EM3001,ER0503; >>>> ERROR #3007 <<<<<.
                                TRAP      C$ERDF
                                .WORD    3007
                                .WORD    EM3001
                                .WORD    ER0503
7148                          ;+
7149                          ; Clean up, exit the test.
7150                          ;-
7151 035770 004767 164772    26$:    JSR      PC,TXIE0      ;CLEAR TRANSMITTER INTERRUPTS.
7152 035774 004767 164204    JSR      PC,RXIE0      ;CLEAR RECEIVER INTERRUPTS.
7153 036000 005067 144346    60$:    CLR      CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
7154 036004          SETPRI  #PRI05      ;DISABLE ALL BUT LTC INTERRUPTS.
                                MOV      #PRI05,R0
                                TRAP      C$SPRI
7155 036012          ENDTST
                                L10056:
036012 104401          TRAP      C$ETST
7156

```

HARDWARE TEST

- DIABMP -

```

7158
7159
7160
7161
7162
7163
7164
7165
7166 036014
      036014
7167      000032
7168 036014 012767 000032 144304
7169 036022 012767 177777 144322
7170 036030 012767 000001 145730
7171 036036 012767 006035 145724
7172 036044 012767 015150 145720
7173 036052 012767 017422 145714
7174
7175
7176
7177
7178
7179 036060 004767 162072
7180 036064 103077
7181
7182
7183
7184
7185
7186 036066 016705 144144
7187 036072 005004
7188 036074 016703 144144
7189 036100 000241
7190 036102 006005
7191 036104 103064
7192
7193
7194
7195
7196 036106 012767 006036 145654
7197 036114 010413
7198 036116 052777 000002 144124
7199
7200
7201
7202
7203 036124 012701 010764
7204 036130 016702 144114
7205 036134 004767 165046
7206 036140 103042
7207
7208
7209
7210
7211 036142 005267 145622
7212 036146 012701 070012
7213 036152 016702 144066

```

```

.SBTTL HARDWARE TEST - DIABMP -
; * *****
; * - Diagnostic field (BMP) Test -
; * This test verifies that a request to the DUT to report BMP status
; * codes is complied with, within the specified time.
; * All active lines are tested.
; * *****

BGNTST

T26::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (31)
      MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
      MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV #3101,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
      MOV #EM3101,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRtbl.
      MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.

; *
; * Reset the DUT to a known state, remove the status codes from the fifo.
; * Clear TX and RX interrupt enable bits in the CSR.
; * This subroutine reports error >>>> 3101 <<<<.
; *
      JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.

; *
; * Test all active lines individually.
; * Write the request code to the diagnostic field in the LPR register.
; * Verify that a BMP code is returned within the correct time.
; *
2$:   MOV ACTLNS,R5 ;GET THE ACTIVE LINE BIT MAP.
      CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
      MOV CSRA,R3 ;GET THE ADDRESS OF THE DUT'S CSR.
      CLC ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
      ROR R5 ;SHIFT THE BIT MAP INTO THE CARRY BIT.
      BCC 8$ ;DO NOT TEST THE LINE IF IT IS INACTIVE.

; *
; * Select the line under test.
; * Write the BMP request code to the DIAG field in the LPR register.
; *
      MOV #3102,ERRNBR ;SET THE ERROR NUMBER TO 3102.
      MOV R4,(R3) ;SELECT THE LINE CURRENTLY UNDER TEST.
      BIS #2,@LPRA ;WRITE THE BMP REQUEST CODE TO THE LPR.

; *
; * Wait for BMP request code to be cleared, report error if time-out
; * occurs.
; *
      MOV #10764,R1 ;TEST BIT 1, TIMEOUT OF 500 MILLI SECS.
      MOV LPRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.
      JSR PC,WAIBIC ;WAIT FOR REQUEST CODE TO CLEAR.
      BCC 6$ ;GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.

; *
; * Wait for BMP code to appear in the fifo, report error if time out
; * occurs.
; *
      INC ERRNBR ;SET ERROR NUMBER TO 3103.
      MOV #70012,R1 ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
      MOV CSRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.

```


HARDWARE TEST

- DIABMP -

```

7214 036156 004767 165100      JSR    PC,WAIBIS      ;WAIT FOR RX_DATA_AVAILABLE TO SET.
7215 036162 103031              BCC    6$             ;GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.
7216                          ;*
7217                          ; Read the BMP code (if it is there) from the RBUF register.
7218                          ; Determine if it is a valid BMP code.
7219                          ; Verify the BMP code was received from the correct channel.
7220                          ; If the BMP code does not indicate DUT running ok, then save it on
7221                          ; the queue to be reported in a later test.
7222                          ;-
7223 036164 005267 145600      INC    ERRNBR         ;SET ERROR NUMBER TO 3104.
7224 036170 017702 144052      MOV    @RBUFA,R2     ;GET THE BMP CODE FROM THE FIFO.
7225 036174 100024              BPL    6$             ;GO REPORT ERROR IF NO BMP CODE FOUND.
7226 036176 005267 145566      INC    ERRNBR         ;SET ERROR NUMBER TO 3105.
7227 036202 012700 170301      MOV    #170301,R0    ;SET-UP A BMP CODE MASK.
7228 036206 040200              BIC    R2,R0         ;TRY TO CLEAR THE BMP MASK.
7229 036210 001016              BNE    6$             ;GO REPORT ERROR IF IT IS NOT A VALID BMP CODE.
7230 036212 005267 145552      INC    ERRNBR         ;SET THE ERROR NUMBER TO 3106.
7231 036216 010200              MOV    R2,R0         ;COPY THE BMP CODE.
7232 036220 000300              SWAB   R0            ;PUT THE LINE NUMBER IN THE LOW BYTE.
7233 036222 042700 177760      BIC    #177760,R0    ;CLEAR THE UNWANTED BITS.
7234 036226 120400              CMPB   R4,R0         ;DID THE BMP CODE COME FROM THE CORRECT LINE?.
7235 036230 001006              BNE    6$             ;NO; GO REPORT ERROR.
7236 036232 120227 000305      CMPB   R2,#305      ;IS THE BMP CODE A "GOOD ONE"?.
7237 036236 001407              BEQ    8$             ;YES; SKIP SAVING THE BMP CODE ON THE QUEUE.
7238 036240 004767 164024      JSR    PC,SAVBMP     ;SAVE THE BMP CODE ON THE QUEUE.
7239 036244 000404              BR     8$             ;GO SEE IF THERE ARE ANY MORE LINE TO TEST.
7240
7241 036246 010401              6$:    MOV    R4,R1     ;PASS THE LINE NUMBER TO BE REPORTED.
7242 036250 012702 015204      MOV    #EM3102,R2    ;PASS THE ERROR MESSAGE TO BE REPORTED.
7243
7244 036254              ERROR          ;"BMP REQUEST BIT BAD ON LINE:"
7244 036254 104460              ;          >>>> ERROR <<<<<<
7245
7246                          ;*
7247                          ; Verify all active lines have been tested.
7248 036256 005204              8$:    INC    R4         ;INCREMENT THE LINE NUMBER COUNTER.
7249 036260 005705              TST    R5            ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
7250 036262 001306              BNE    2$             ;YES; BRANCH TO TEST THE NEXT LINE.
7251 036264 005067 144062      60$:   CLR    CTRLCF    ;INDICATE THAT WE ARE NOT WITHIN A TEST.
7252 036270
7252 036270 104401              L10057:
7252 036270              TRAP    C$ETST
    
```

HARDWARE TEST

- DIABMP -

7254
7255
7256
7257
7258
7259
7260
7261
7262
7263
7264
7265 036272
036272
7266 000033
7267 036272 012767 000033 144026
7268 036300 012767 177777 144044
7269 036306 016702 144212
7270 036312 012703 002526
7271 036316 020203
7272 036320 001411
7273
7274
7275
7276
7277
7278 036322 012701 016013
7279 036326
036326 104455
036330 022125
036332 015676
036334 017450
7280
7281 036336 012767 002526 144160
7282
7283 036344 005067 144002 60\$:
7284 036350
036350
036350 104401

```

.SBTTL HARDWARE TEST - REPBM -
;+ *****
;* - Report any BMP codes in the queue -
;* This is a pseudo-test used to report any BMP codes that were found
;* in the DUT's FIFO during previous test, and logged in the BMP code
;* queue.
;* It is unlikely that running this pseudo test alone will produce any
;* error reports.
;+ *****
;-- BGNTST
;+
;T27::
;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (93)
MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
MOV BMPCQP,R2 ;GET THE CONTENTS OF THE POINTER.
MOV #BMPCQB,R3 ;GET THE START ADDRESS OF THE QUEUE.
CMP R2,R3 ;SEE IF THE POINTER HAS MOVED FROM THE BASE.
BEQ 60$ ;EXIT NO CODES IN THE QUEUE.
;+
; There is at least one BMP code in the queue. Report the error.
;--
;Report error BMP CODE FOUND IN TEST nn, BMP CODE:nnnnnn"
MOV #EM9304,R1 ;PASS THE FIRST MESSAGE TO BE REORTED.
ERRDF 9301,EM9301,ER9301 ; >>>> ERROR #9301 <<<<<.
TRAP C$ERDF
.WCRD 9301
.WORD EM9301
.WORD ER9301
MOV #BMPCQB,BMPCQP ;SET POINTER BACK TO THE BEGINING OF THE QUE.
CLR CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.
ENDTST
L10060:
TRAP C$ETST

```

HARDWARE TEST

- REPBMF -

7287
7288
7289
7290
7291
7293
7294
7295
7296
7297
7298
7299
7300
7301
7302
7303
7304
7305
7306
7307

```
*****
:
:                               VDHA.HWQ
:
*****
```

.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.
:--
```

7308 036352
036352 000022
036354

BGNHRD

.WORD L10061 L\$HARD/2
L\$HARD::

7309
7319

```
:DEVICE CSR ADDRESS QUESTION:
GPRMA HWPTQ1,0,0,160000,177776,YES
```

7320 036354
036354 000031
036356 036420
036360 160000
036362 177776

.WORD T\$CODE
HWPTQ1
.WORD T\$LOLIM
T\$HILIM

7321
7322

```
:DEVICE INTERRUPT VECTOR QUESTION:
GPRMA HWPTQ2,2,0,40,776,YES
```

036364 001031
036366 036436
036370 000040
036372 000776

.WORD T\$CODE
HWPTQ2
.WORD T\$LOLIM
T\$HILIM

7323
7324

```
:ACTIVE LINES BIT MAP QUESTION:
GPRMD HWPTQ3,4,0,MAPLNS,0,177777,YES
```

036374 002032
036376 036471
036400 000377
036402 000000
036404 177777

.WORD T\$CODE
HWPTQ3
.WORD MAPLNS
T\$LOLIM
T\$HILIM

7325
7326

```
:INTERRUPT BR LEVEL QUESTION:
GPRMD HWPTQ5,6,0,377,0,6,YES
```

036406 003032
036410 036517
036412 000377
036414 000000
036416 000006

.WORD T\$CODE
HWPTQ5
.WORD 377
T\$LOLIM
T\$HILIM

7327
7328

7329 036420

ENDHRD

.EVEN

036420

L10061:

7330
7337

HARDWARE PARAMETER CODING SECTION

7338	036420	103	123	122	HWPTQ1: .ASCIZ /CSR ADDRESS: /
	036423	040	101	104	
	036426	104	122	105	
	036431	123	123	072	
	036434	040	000		
7339	036436	111	116	124	HWPTQ2: .ASCIZ /INTERRUPT VECTOR ADDRESS: /
	036441	105	122	122	
	036444	125	120	124	
	036447	040	126	105	
	036452	103	124	117	
	036455	122	040	101	
	036460	104	104	122	
	036463	105	123	123	
	036466	072	040	000	
7340	036471	101	103	124	HWPTQ3: .ASCIZ /ACTIVE LINE BIT MAP: /
	036474	111	126	105	
	036477	040	114	111	
	036502	116	105	040	
	036505	102	111	124	
	036510	040	115	101	
	036513	120	072	040	
	036516	000			
7341	036517	111	116	124	HWPTQ5: .ASCIZ /INTERRUPT BR LEV L: /
	036522	105	122	122	
	036525	125	120	124	
	036530	040	102	122	
	036533	040	114	105	
	036536	126	105	114	
	036541	072	040	000	

7342
7343

.EVEN

HARDWARE PARAMETER CODING SECTION

```

7346 ;*****
* 7347 ;
7348 ;           VDHA.SWQ
7349 ;
7350 ;*****

```

```

7352
7353
7354 .SBTTL SOFTWARE PARAMETER CODING SECTION
7355

```

```

7356 ;**
7357 ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
7358 ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
7359 ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7360 ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
7361 ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7362 ; WITH THE OPERATOR.
7363 ;-

```

```

7364
7365 036544          BGNSFT
      036544 000013
      036546
                                .WORD L10062-L$SOFT/2
                                L$SOFT::

```

```

7366
7375 ;UNIT NUMBER PRINTOUT QUESTION:
7376 036546          GPRML SWPTQ1,0,20,YES
      036546 000130
      036550 036574
      036552 000020
                                .WORD T$CODE
                                .WORD SWPTQ1
                                .WORD 20

```

```

7377 ;NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE QUESTION:
7378 036554          GPRMD SWPTQ2,2,D,177777,0,177777,YES
      036554 001052
      036556 036650
      036560 177777
      036562 000000
      036564 177777
                                .WORD T$CODE
                                .WORD SWPTQ2
                                .WORD 177777
                                .WORD T$LOLIM
                                .WORD T$HILIM

```

```

7379 ;ROM VERSION PRINTOUT ON FIRST PASS QUESTION:
7380 036566          GPRML SWPTQ3,0,1,YES
      036566 000130
      036570 036737
      036572 000001
                                .WORD T$CODE
                                .WORD SWPTQ3
                                .WORD 1

```

```

7381
7382          .EVEN
7383
7384 036574          ENDSFT
                                .EVEN
                                L10062:

```

```

7385
7386
7393 036574          122      105      120
      036577          117      122      124
      036602          040      125      116
      036605          111      124      040
      036610          116      125      115
      036613          102      105      122
      036616          040      101      123
      036621          040      105      101
      036624          103      110      040
      036627          125      116      111
      SWPTQ1: .ASCIZ /REPORT UNIT NUMBER AS EACH UNIT IS TESTED: /

```

SOFTWARE PARAMETER CODING SECTION

	036632	124	040	111	
	036635	123	040	124	
	036640	105	123	124	
	036643	105	104	072	
	036646	040	000		
7394	036650	116	125	115	SWPTQ2: .ASCIZ /NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE: /
	036653	102	105	122	
	036656	040	117	106	
	036661	040	111	116	
	036664	104	111	126	
	036667	111	104	125	
	036672	101	114	040	
	036675	104	101	124	
	036700	101	040	105	
	036703	122	122	117	
	036706	122	123	040	
	036711	124	117	040	
	036714	122	105	120	
	036717	117	122	124	
	036722	040	117	116	
	036725	040	101	040	
	036730	114	111	116	
	036733	105	072	040	
	036736	000			
7395	036737	122	117	115	SWPTQ3: .ASCIZ /ROM VERSION PRINTOUT ON THE FIRST PASS: /
	036742	040	126	105	
	036745	122	123	111	
	036750	117	116	040	
	036753	120	122	111	
	036756	116	124	117	
	036761	125	124	040	
	036764	117	116	040	
	036767	124	110	105	
	036772	040	106	111	
	036775	122	123	124	
	037000	040	120	101	
	037003	123	123	072	
	037006	040	000		
7396					.EVEN

SOFTWARE PARAMETER CODING SECTION

```

7398
7399
7400      ;*****
7401      ;
7402      ;           FVTSKL6.P11
7403      ;
7404      ;*****
7405
7406
7407 037010  $PATCH:
7408 037010      .BLKW  24
7409
7416
7417
7418
7419
7420 037060      LASTAD
                                .EVEN
                                .WORD  0
                                .WORD  0
7421 037064      L$LAST:
7422      .ENDMOD
7423
7424
7425
7426
7427
7428
7429      000001      .END

```

Symbol table

ACTLNS	002236	G	CTRLCF	002352	G	C\$RPT =	000025	EM0401	007252	G	EM2608	014274	G
ADR =	000020	G	C\$AU =	000052		C\$SEFG=	000046	EM0402	007321	G	EM2609	014345	G
ADRPTR	020140	G	C\$AUTO=	000061		C\$SPRI=	000041	EM0501	007471	G	EM2610	014424	G
ALTFLD	017630	G	C\$BRK =	000022		C\$SVEC=	000037	EM0502	007537	G	EM2611	014516	G
ASSEMB=	000010		C\$BSEG=	000004		C\$TOME=	000076	EM0525	007713	G	EM3001	014601	G
BCOUNT	002372	G	C\$BSUB=	000002		DELAY	020300	EM0526	010003	G	EM3002	014632	G
BITTBL	002410	G	C\$CLCK=	000062		DFPTBL	002214	EM0601	010073	G	EM3003	014726	G
BIT0 =	000001	G	C\$CLEA=	000012		DIAGMC=	000000	EM0602	010127	G	EM3004	015002	G
BIT00 =	000001	G	C\$CLOS=	000035		DRADRT	002244	EM0603	010307	G	EM3005	015056	G
BIT01 =	000002	G	C\$CLP1=	000006		DROP	025202	EM0701	010472	G	EM3101	015150	G
BIT02 =	000004	G	C\$CPBF=	000074		DR00MG	006004	EM0702	010527	G	EM3102	015204	G
BIT03 =	000010	G	C\$CPME=	000075		DR02MG	006010	EM0703	010707	G	EM9009	015245	G
BIT04 =	000020	G	C\$CVEC=	000036		DR04MG	006015	EM0801	011072	G	EM9010	015271	G
BIT05 =	000040	G	C\$DCLN=	000044		DR06MG	006021	EM0802	011132	G	EM9014	015315	G
BIT06 =	000100	G	C\$DODU=	000051		DR10MG	006026	EM0901	011205	G	EM9017	015411	G
BIT07 =	000200	G	C\$DRPT=	000024		DR12MG	006035	EM0902	011236	G	EM9018	015522	G
BIT08 =	000400	G	C\$DU =	000053		DR14MG	006046	EM1001	011272	G	EM9019	015532	G
BIT09 =	001000	G	C\$EDIT=	000000		DR16MG	006057	EM1002	011336	G	EM9020	015547	G
BIT1 =	000002	G	C\$ERDF=	000055		EDROP	025260	EM1003	011423	G	EM9022	015573	G
BIT10 =	002000	G	C\$ERHR=	000056		EF.CON=	000036	EM1101	011501	G	EM9023	015612	G
BIT11 =	004000	G	C\$ERRO=	000060		EF.NEW=	000035	EM1201	011525	G	EM9024	015634	G
BIT12 =	010000	G	C\$ERSF=	000054		EF.PWR=	000034	EM1202	011544	G	EM9026	015652	G
BIT13 =	020000	G	C\$ERSO=	000057		EF.RES=	000037	EM1203	011630	G	EM9301	015676	G
BIT14 =	040000	G	C\$ESCA=	000010		EF.STA=	000040	EM1204	011706	G	EM9302	015716	G
BIT15 =	100000	G	C\$ESEG=	000005		EF0503	004134	EM1205	011742	G	EM9303	015746	G
BIT2 =	000004	G	C\$ESUB=	000003		EF0505	004141	EM1301	011766	G	EM9304	016013	G
BIT3 =	000010	G	C\$ETST=	000001		EF1401	004214	EM1302	012012	G	ENDETB	003726	G
BIT4 =	000020	G	C\$EXIT=	000032		EF1402	004316	EM1401	012051	G	ENDIT	025066	
BIT5 =	000040	G	C\$FREQ=	000101		EF1601	004353	EM1402	012101	G	ERCNTB	002464	G
BIT6 =	000100	G	C\$FRME=	000100		EF1602	004377	EM1403	012167	G	ERLTBL	002726	G
BIT7 =	000200	G	C\$GETB=	000026		EF1603	004441	EM1404	012242	G	ERRBLK	003774	G
BIT8 =	000400	G	C\$GETW=	000027		EF1604	004503	EM1405	012314	G	ERRMSG	003772	G
BIT9 =	001000	G	C\$GMAN=	000043		EF3001	004600	EM1406	012327	G	ERRNBR	003770	G
BMPCQB	002526	G	C\$GPHR=	000042		EF3002	004647	EM1407	012342	G	ERRTYP	003766	G
BMPCQE	002726	G	C\$GPRI=	000040		EF9001	004716	EM1408	012354	G	ERSMRF	002462	G
BMPCQP	002524	G	C\$INIT=	000011		EF9002	005000	EM1601	012362	G	ER0101	016070	G
BOE =	000400	G	C\$INLP=	000020		EF9003	005057	EM1604	012445	G	ER0201	016406	G
BRLEVL	002242	G	C\$MANI=	000050		EF9004	005113	EM1701	012513	G	ER0503	016470	G
BUFBAS	002726	G	C\$MAP =	000102		EF9005	005150	EM1801	012570	G	ER0504	016514	G
BUFEND	003726	G	C\$MEM =	000031		EF9006	005201	EM1901	012636	G	ER1401	016562	G
BUF MID	003326	G	C\$MMU =	000103		EF9010	005225	EM2001	012713	G	ER1601	016712	G
BUF PTR	002324	G	C\$MSG =	000023		EF9016	005324	EM2002	012755	G	ER1603	017010	G
BUF3QT	003526	G	C\$OPNR=	000034		EF9017	005421	EM2101	013030	G	ER3001	017070	G
CACHRX	023636	G	C\$OPNW=	000104		EF9018	005475	EM2102	013073	G	ER9004	017160	G
CACHTX	023664	G	C\$PNTB=	000014		EF9019	005602	EM2201	013167	G	ER9007	017262	G
CALMSL	017702	G	C\$PNTF=	000017		EF9301	005626	EM2202	013224	G	ER9008	017340	G
CKTRAP	020126	G	C\$PNTS=	000016		EF9302	005704	EM2203	013316	G	ER9101	017422	G
CLKBRL	002356	G	C\$PNTX=	000015		EM0101	020534	EM2301	013423	G	ER9301	017450	G
CLKCSR	002354	G	C\$PUTB=	000072		EM0102	020620	EM2302	013461	G	EVL =	000004	G
CLKHRZ	002362	G	C\$PUTW=	000073		EM0103	006067	EM2401	013517	G	E\$END =	002100	
CLKINT	023712	G	C\$QIO =	000377		EM0201	006125	EM2601	013553	G	E\$LOAD=	000035	
CLKVEC	002360	G	C\$RDBU=	000007		EM0202	006173	EM2602	013603	G	F\$AU =	000015	
CLNRST	020156	G	C\$REFG=	000047		EM0203	006346	EM2603	013674	G	F\$AUTO=	000020	
CLR16W	020200	G	C\$REL =	000077		EM0204	006511	EM2604	013762	G	F\$BGN =	000040	
CNTERR	020222	G	C\$RESE=	000033		EM0301	006670	EM2605	014056	G	F\$CLEA=	000007	
CSRA	002244	G	C\$REVI=	000004		EM0302	006733	EM2606	014141	G	F\$DU =	000016	
CSRO =	000000	G	C\$RFLA=	000021		EM0303	007073	EM2607	014200	G	F\$END =	000041	

Symbol table

F\$HARD= 000004	I\$PTAB= 000041	L\$PROT 024242 G	L10054 034104	RMATBB 002304 G
F\$HW = 000013	I\$PWR = 000041	L\$PRT 002112 G	L10055 035130	ROLDAP 021570 G
F\$INIT= 000006	I\$RPT = 000041	L\$REPP 002062 G	L10056 036012	RSTRPT 021622 G
F\$JMP = 000050	I\$SEG = 000041	L\$REV 002010 G	L10057 036270	RXBCTX= 000030 G
F\$MOD = 000000	I\$SETU= 000041	L\$RPT 024234 G	L10060 036350	RXBCTX= 000020 G
F\$MSG = 000011	I\$SFT = 000041	L\$SOFT 036546 G	L10061 036420	RXBFUL= 000100 G
F\$PROT= 000021	I\$SRV = 000041	L\$SPC 002056 G	L10062 036574	RXBRRT 023762 G
F\$PWR = 000017	I\$SUB = 000041	L\$SPCP 002020 G	MAPLNS= 000377 G	RXIE0 022204 G
F\$RPT = 000012	I\$TST = 000041	L\$SPTP 002024 G	MFUNIT 004076 G	RXIE1 022244 G
F\$SEG = 000003	J\$JMP = 000167	L\$STA 002030 G	MMENAB 002404 G	RXINPT 024052 G
F\$SOFT= 000005	LNCTRA 002254 G	L\$SW 002226 G	MMPRES 002402 G	RXINTC 002334 G
F\$SRV = 000010	LNCTRO= 000010 G	L\$TEST 002114 G	MMSRO 002400 G	RXINTF 002336 G
F\$SUB = 000002	LOE = 040000 G	L\$TIML 002014 G	MSG1 016174 G	RXVECA 002232 G
F\$SW = 000014	LOT = 000010 G	L\$UNIT 002012 G	MSG2 016252 G	R0SLOT= 000002 G
F\$TEST= 000001	LPCSLT= 000036 G	L10000 002224	MSG3 016331 G	R1SLOT= 000004 G
GETPRM 024652	LPRA = 002250 G	L10001 002232	MSLCNT 002376 G	R2SLOT= 000006 G
GPRSOB 002450 G	LPRO = 000004 G	L10002 016172	MSLGET 020340 G	R3SLOT= 000010 G
G\$CNT0= 000200	L\$ACP 002110 G	L10003 016466	MSLOOP 020454 G	R4SLOT= 000012 G
G\$DELM= 000372	L\$APT 002036 G	L10004 016512	MSTICK 002374 G	R5SLOT= 000014 G
G\$DISP= 000003	L\$AU 025266 G	L10005 016560	NDERPT 002230 G	SAVBMP 022270 G
G\$EXCP= 000400	L\$AUT 002070 G	L10006 016710	NEWPAS 024632	SFPTBL 002226 G
G\$HILI= 000002	L\$AUTO 025116 G	L10007 017006	NEWRES 024624	SKPSTS 022336 G
G\$LOLI= 000001	L\$CCP 002106 G	L10010 017066	NEWSTA 024314	STATA 002252 G
G\$NO = 000000	L\$CLEA 025120 G	L10011 017156	NUMLNS= 000010 G	STATO = 000006 G
G\$OFFS= 000400	L\$CO 002032 G	L10012 017260	OOPS 020470 G	SVCGBL= 000000
G\$OFSI= 000376	L\$DEPO 002011 G	L10013 017336	OPTION 002226 G	SVCINS= 000001
G\$PRMA= 000001	L\$DESC 004046 G	L10014 017420	O\$APTS= 000000	SVCSUB= 000001
G\$PRMD= 000002	L\$DESP 002076 G	L10015 017446	O\$AU = 000000	SVCTAG= 000001
G\$PRML= 000000	L\$DEVP 002060 G	L10016 017626	O\$BGNR= 000001	SVCTST= 000001
G\$RADA= 000140	L\$DISP 002124 G	L10017 024240	O\$BGNS= 000001	SWAPO 022414 G
G\$RADB= 000000	L\$DLY 002116 G	L10021 025114	O\$DU = 000001	SWPTQ1 036574
G\$RADD= 000040	L\$DTP 002040 G	L10022 025116	O\$ERRT= 000001	SWPTQ2 036650
G\$RADL= 000120	L\$DTYP 002034 G	L10023 025154	O\$GNSW= 000001	SWPTQ3 036737
G\$RADO= 000020	L\$DU 025156 G	L10024 025264	O\$POIN= 000001	S\$LSYM= 010000
G\$XFER= 000004	L\$DUT 002072 G	L10025 025272	O\$SETU= 000000	TIMER1 002364 G
G\$YES = 000010	L\$DVTY 004036 G	L10026 025562	PAROA 002406 G	TIMER2 002366 G
HELP = 000000	L\$EF 002052 G	L10027 025776	PASCNT 002332 G	TIMER3 002370 G
HOE = 100000 G	L\$ENVI 002044 G	L10030 026226	PCSLT= 000016 G	TNUM = 000033 G
HWPTQ1 036420	L\$ERRT 003766 G	L10031 026410	PNT = 001000 G	TP4FLG 002346 G
HWPTQ2 036436	L\$ETP 002102 G	L10032 026566	PREGRT 004020 G	TP4RTN 024136 G
HWPTQ3 036471	L\$EXP1 002046 G	L10033 026770	PREG05 003776	TP4VEC 002344 G
HWPTQ5 036517	L\$EXP4 002064 G	L10034 027162	PRI = 002000 G	TSABRT 022464 G
IBE = 010000 G	L\$EXP5 002066 G	L10035 027354	PRI00 = 000000 G	TSTNUM 002326 G
IDU = 000040 G	L\$HARD 036354 G	L10036 027526	PRI01 = 000040 G	TXAD1A 002256 G
IER = 020000 G	L\$HIME 002120 G	L10037 027726	PRI02 = 000100 G	TXAD10= 000012 G
IESTAT 002330 G	L\$HPCP 002016 G	L10040 030140	PRI03 = 000140 G	TXAD2A 002260 G
ISR = 000100 G	L\$HPTP 002022 G	L10041 030406	PRI04 = 000200 G	TXAD20= 000014 G
IXE = 004000 G	L\$HW 002214 G	L10042 030750	PRI05 = 000240 G	TXBFCA 002262 G
I\$AU = 000041	L\$ICP 002104 G	L10043 031334	PRI06 = 000300 G	TXBFCO= 000016 G
I\$AUTO= 000041	L\$INIT 024250 G	L10044 031544	PRI07 = 000340 G	TXCHA 002246 G
I\$CLN = 000041	L\$LADP 002026 G	L10045 031760	PUFIFO 020716 G	TXCHRO= 000002 G
I\$DU = 000041	L\$LAST 037064 G	L10046 032224	RBUFA 002246 G	TXDSBL 022576 G
I\$HRD = 000041	L\$LOAD 002100 G	L10047 032474	RBUFO = 000002 G	TXENBL 022672 G
I\$INIT= 000041	L\$LUN 002074 G	L10050 032602	RDPDR 021000 G	TXIE0 022766 G
I\$MOD = 000041	L\$MREV 002050 G	L10051 033012	REGTST 021220 G	TXIE1 023026 G
I\$MSG = 000041	L\$NAME 002000 G	L10052 033246	REPSMR 021430 C	TXINTC 002340 G
I\$PROT= 000040	L\$PRIO 002042 G	L10053 033546	RESETT 021456 G	TXINTF 002342 G

Symbol table

TXINTR 024160 G	T\$SAVL= 177777	T\$\$PRO= 010020	T2 025564 G	UAM = 000200 G
TXVECA 002234 G	T\$SEGL= 177777	T\$\$RPT= 010017	T20 032604 G	UNBTB 002264 G
T\$ARGC= 000003	T\$SUBN= 000000	T\$\$SOF= 010062	T21 033014 G	UNITN 002240 G
T\$CODE= 000130	T\$TAGL= 177777	T\$\$SW = 010001	T22 033250 G	UNSDIV 023052 G
T\$ERRN= 022125	T\$TAGN= 010063	T\$\$TES= 010060	T23 033550 G	WAIBIC 023206 G
T\$EXCP= 000000	T\$TEMP= 000000	T1 025274 G	T24 034106 G	WAIBIS 023262 G
T\$FLAG= 000050	T\$TEST= 000033	T10 027530 G	T25 035132 G	WDPDR 023336 G
T\$GMAN= 000000	T\$TSTM= 177777	T11 027730 G	T 036272 G	WORD1 002350 G
T\$HILI= 177777	T\$TSTS= 000001	T12 030142 G	T5 026000 G	WTWLNC 023532 G
T\$LAST= 000001	T\$\$AU = 010025	T13 030410 G	T4 026230 G	WTWLNS 023562 G
T\$LOLI= 000000	T\$\$AUT= 010022	T14 030752 G	T5 026412 G	WTWLPR 023606 G
T\$LSYM= 010000	T\$\$CLE= 010023	T15 031336 G	T6 026570 G	X\$ALWA= 000000
T\$LTNO= 000033	T\$\$DU = 010024	T16 031546 G	T7 026772 G	X\$FALS= 000040
T\$NEST= 177777	T\$\$HAR= 010061	T17 031762 G	T8 027164 G	X\$OFFS= 000400
T\$NSO = 000000	T\$\$HW = 010000	T18 032226 G	T9 027356 G	X\$TRUE= 000020
T\$NS1 = 000005	T\$\$INI= 010021	T19 032476 G		\$PATCH 037010 G
T\$PTNU= 000000	T\$\$MSG= 010016			

. ABS. 037064 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 275
 Work file writes: 284
 Size of work file: 35712 Words (140 Pages)
 Size of core pool: 19714 Words (75 Pages)
 Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:06:07.92
 CVDHAC.BIN,CVDHAC.LST/-SP=SVC40/ML,CVDHAC.P11