

DHV11-M

DHV11-M FUNC TST PART 1
CVDHAD0

AH-T653D-MC
1 OF 1 OCT 1985
COPYRIGHT © 1983-85

digital
MADE IN USA

The main body of the document is a large grid of 144 small, illegible data tables or charts, arranged in 12 rows and 12 columns. Each cell contains a small table with text and numbers, likely representing test results or functional data for the DHV11-M device. The text within these cells is too small to be read, but the overall layout suggests a comprehensive set of test data or a functional test matrix.

.REM &

IDENTIFICATION

PRODUCT CODE: AC-T652D-MC
PRODUCT NAME: CVDHADO DHV11-M FUNC TST PART 1
PRODUCT DATE: MAY 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.
- Version D0 8-May-85 Howard Marshall
Found timing problems when running under the extended
DRS monitor on a PDP 11/23+ in tests: #10, Skip selftest
test and #13, Selftest fail test because extra under slower
CPUs, DRS calls consume too much time and interfere with
the critical timing loops in these tests. Substituted
SETPRI + GETPRI with MTPS + MFPS respectively.

ATTENTION: /NO RESTORE (of DRS) IS ASSEMBLED INTO THIS DIAG. !!!

)
V

 * Found additional timing problems when running under the *
 * extended DRS monitor on a PDP 11/23+ in test #3, Master reset *
 * skip selftest test. The DRS calls are consuming too much time *
 * and adversely affecting the timing of the critical loops in test *
 * #3. Assembled in "/NORESTORE" into this source code by adding *
 * an argument (a 1) to the end of the HEADER macro. *

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 HIERARCHY 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>)	
2.4.1.2	Pass Switch (/PASS:<PASS-CNT>)	
2.4.1.3	Flags Switch (/FLAGS:<FLAG-LIST>)	
2.4.1.4	End Of Pass Switch (/EOP:<INCR>)	
2.4.1.5	Effect Of Start Command	
2.4.2	Restart Command	11
2.4.2.1	Tests, Pass, And Flags Switches	
2.4.2.2	Units Switch (/UNITS:<UNIT-LIST>)	
2.4.2.3	Effect Of Restart Command	
2.4.3	Continue Command	11
2.4.3.1	Flag Switch (/FLAGS:<FLAG-LIST>)	
2.4.3.2	Effect Of Continue Command	
2.4.4	Proceed Command	12
2.4.4.1	Flags Switch (/FLAGS:<FLAG-LIST>)	
2.4.4.2	Effect Of Proceed Command	
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	
2.4.10	Display Command	13
2.4.10.1	Effect Of Display Command	
2.4.11	Flags Command	13
2.4.11.1	Effect Of Flags Command	
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

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.

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 is 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
 /FLAGS:FLGS
 /EOP:DDDDD
 /UNITS:LIST

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

PROGRAM DOCUMENT

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>

PROGRAM DOCUMENT

SEQ 0018

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 (OVERRRUN, 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.

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 CVDHADO
CVDHADO.BIC

DRS
CVDHA-D-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

```

1092      &
1093      .LIST SEQ,LOC,BIN,MEB
1094      .NLIST CND
1095      ;*****
1096      ;
1097      ;           VDHA.PHD
1098      ;
1099      ;*****
1100
1101
1102
1103      .SBTTL Program Header
1104
1105
1106      .MCALL SVC
1107 000000 SVC ; INITIALIZE SUPERVISOR MACROS
1108
1109      ;*****
1110      ; IF STRUCTURED MACROS ARE TO BE USED, ADD ".MCALL STRUCT" AND "STRUCT"
1111      ; TO INITIALIZE THE STRUCTURED MACROS.
1112
1113      000001 SVCINS= 1 ; LIST INSTRUCTIONS, SHIFTED RIGHT
1114      000001 SVCTST= 1 ; LIST TEST TAGS, SHIFTED RIGHT
1115      000001 SVCSUB= 1 ; LIST SUBTEST TAGS, SHIFTED RIGHT
1116      000001 SVCGBL= 1 ; LIST GLOBAL TAGS, SHIFTED RIGHT
1117      000001 SVCTAG= 1 ; LIST OTHER TAGS, SHIFTED RIGHT
1118
1119      ; CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
1120      ; TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
1121      ; SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
1122      ; CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.
1123      ;*****
1124
1125 000000 .ENABL ABS
1126      ;.ENABL AMA
1127      = 2000
1128
1129 002000 BGNMOD
1130
1131      ;++
1132      ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
1133      ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
1134      ;--
1135
1136 002000 POINTER BGNRPT,BGNSW,BGNSFT,BGNDU,ERRTBL
1137
1138
1139
1140
1141
1142
1143
1144
1145 002000 HEADER CVDHA,D,0,16,0,PRI07,1
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155 002000
002000 103
002001 126
002002 104
002003 110
002004 101
002005 000

```

```

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

```

Program Header

```

002006      000
002007      000
002010      104
002011      060
002012     000000
002014     000016
002016     036310
002020     036502
002022     002214
002024     002226
002026     037020
002030     000000
002032     000000
002034     000000
002036     000000
002040     002124
002042     000340
002044     000000
002046     000000
002050      004
002051      000
002052     000000
002054     000000
002056     000000
002060     004036
002062     024220
002064     000000
002066     000000
002070     000000
002072     025134
002074
    
```

```

      .BYTE 0
      .BYTE 0
L$REV:: .ASCII /D/
L$DEPO:: .ASCII /O/
L$UNIT:: .WORD 0
L$TIML:: .WORD 16
L$HPCP:: .WORD L$HARD
L$SPCP:: .WORD L$SOFT
L$HPTP:: .WORD L$HW
L$SPTP:: .WORD L$SW
L$LADP:: .WORD L$LAST
L$STA:: .WORD 0
L$CO:: .WORD 0
L$DTYP:: .WORD 0
L$APT:: .WORD 0
L$DTP:: .WORD L$DISPATCH
L$PRIO:: .WORD PRI07
L$ENVI:: .WORD 0
L$EXP1:: .WORD 0
L$MREV:: .BYTE C$REVISION
      .BYTE C$EDIT
L$EF:: .WORD 0
      .WORD 0
L$SPC:: .WORD 0
L$DEVP:: .WORD L$DVTYP
L$REPP:: .WORD L$RPT
L$EXP4:: .WORD 0
L$EXP5:: .WORD 0
L$AUT:: .WORD 0
L$DUT:: .WORD L$DU
L$LUN::
    
```

Program Header

002074 000000
002076 004046
002100 104035
002102 003766
002104 024234
002106 025100
002110 025076
002112 024226
002114 000001
002116 000000
002120 000000

1156

L\$DESP:: .WORD 0
L\$LOAD:: .WORD L\$DESC
 EMT E\$LOAD
L\$ETP:: .WORD L\$ERRTBL
L\$ICP:: .WORD L\$INIT
L\$CCP:: .WORD L\$CLEAN
L\$ACP:: .WORD L\$AUTO
L\$PRT:: .WORD L\$PROT
L\$TEST:: .WORD 1
L\$DLY:: .WORD 0
L\$HIME:: .WORD 0

DISPATCH TABLE

1168
1169
1170
1171
1172
1173
1174
1175

.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

```

002122 000033
002124
002124 025252
002126 025542
002130 025756
002132 026206
002134 026370
002136 026546
002140 026750
002142 027142
002144 027334
002146 027506
002150 027706
002152 030120
002154 030366
002156 030730
002160 031314
002162 031524
002164 031740
002166 032204
002170 032454
002172 032562
002174 032772
002176 033226
002200 033526
002202 034064
002204 035076
002206 035750
002210 036226
    
```

```

        .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
    
```

1176

DISPATCH TABLE

```

1184
1185 :*****
1186 :
1187 :          VDHA.DHT
1188 :
1189 :*****
*****
1190
1191
1192
1193 .SBTTL  DEFAULT HARDWARE P-TABLE
1194
1195 :**
1196 : THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
1197 : THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
1198 : IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES.
1199 : AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
1200 :--
1201
1202          BGNHW  DFPTBL
1203          002212  000004
1204          002212  000004
1205          002214
1206          002214
1207          002214
1208
1209
1210          002224          .WORD  L10000-L$HW/2
1211          002224          DFPTBL::
1212
1213          .WORD  160460  ;Default CSR Address
1214          .WORD  300    ;Default Vector Address
1215          .WORD  177777 ;Default Active lines bit map
1216          .BYTE  4      ;Default BR Level
1217          .EVEN
1218
1219          ENDPHW
1220
1221          L10000:

```

DEFAULT HARDWARE P-TABLE

1212
1213

1214
1215
1216
1217
1218
1219
1220

:
: VDHA.SWT
:

1221
1222
1223
1224
1225
1226
1227
1228
1229

.SBTTL SOFTWARE P-TABLE
:
:++
: THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
: PROGRAM AS OPERATIONAL PARAMETERS. THESE PARAMETERS ARE
: SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
: AT RUN TIME.
:--

1230 002224
002224 000002
002226
002226

BGNSW SFPTBL
L\$SW:: .WORD L10001-L\$SW/2
SFPTBL::

1231
1232 002226 000021
1233 002230 000000

OPTION:: .WORD 21 ;bit map of program control flags
NDERPT:: .WORD 0 ;Default number of individual data errors to rpt.

1234
1235 002232
002232

ENDSW
L10001:

SOFTWARE P-TABLE

```

1237
1238      ;*****
1239      ;
1240      ;           VDHA.EQU
1241      ;
1242      ;*****
1243
1244
1245      .SBTTL GLOBAL EQUATES SECTION
1246
1247
1248
1249
1250
1251      ;++
1252      ; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
1253      ; ARE USED IN MORE THAN ONE TEST.
1254      ;--
1255
1256      000010      NUMLNS==10      ;NUMBER OF LINES ON DHV11 IS 8.
1257      000377      MAPLNS==377    ;BIT MAP OF LINES ON DHV11.
1258
1259      ;***** DEVICE REGISTER OFFSETS FROM THE CSR'S ADDRESS *****
1260      000000      CSRO==0        ;CSR REGISTER OFFSET FROM THE CSR ADDRESS
1261      000002      RBUFO==2       ;RECEIVE REGISTER OFFSET FROM THE CSR ADDRESS
1262      000002      TXCHRO==2      ;TRANSMIT REGISTER OFFSET FROM THE CSR ADDRESS
1263      000004      LPRO==4        ;LINE PARAMETER REGISTER OFFSET FROM THE CSR ADDRESS
1264      000006      STATO==6       ;STATUS REGISTER OFFSET FROM THE CSR ADDRESS
1265      000010      LNCTRO==10     ;LINE CONTROL REGISTER OFFSET FROM THE CSR ADDRESS
1266      000012      TXAD10==12    ;TRANSMIT ADDRESS 1 REGISTER OFFSET FROM THE CSR ADDRESS
1267      000014      TXAD20==14    ;TRANSMIT ADDRESS 2 REGISTER OFFSET FROM THE CSR ADDRESS
1268      000016      TXBFCO==16    ;TRANSMIT COUNT REGISTER OFFSET FROM THE CSR ADDRESS
1269
1270      ;***** EQUATES USED WITH RESPECT TO THE RX BUFFER *****
1271      000020      RXBETX==16.    ;LEVEL OF RX BUFFER AT WHICH TO RE-ENABLE TRANSMISSION.
1272      000030      RXBDTX==24.   ;LEVEL OF RX BUFFER AT WHICH TO DISABLE TRANSMISSION.
1273      000100      RXBFUL==64.   ;TOTAL CHARACTER CAPACITY OF THE RX BUFFER.
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1298 002232      EQUALS
1299
1300      ;
1301      ; BIT DIFINITIONS
1302      ;
1303      100000      BIT15== 100000
1304      040000      BIT14== 40000
1305      020000      BIT13== 20000
1306      010000      BIT12== 10000
1307      004000      BIT11== 4000
1308      002000      BIT10== 2000
1309      001000      BIT09== 1000
1310      000400      BIT08== 400
1311      000200      BIT07== 200
1312      000100      BIT06== 100
1313      000040      BIT05== 40
1314      000020      BIT04== 20
1315      000010      BIT03== 10
1316      000004      BIT02== 4
1317      000002      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
                ;
                ;
000040          EF.START==      32.          ; BIT POSITION IN SECOND STATUS WORD
000037          EF.RESTART==    31.          ; (100000) START COMMAND WAS ISSUED
000036          EF.CONTINUE==   30.          ; (040000) RESTART COMMAND WAS ISSUED
000035          EF.NEW==        29.          ; (020000) CONTINUE COMMAND WAS ISSUED
000034          EF.PWR==        28.          ; (010000) A NEW PASS HAS BEEN STARTED
                ;                               ; (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

1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321 002232 000300
1322 002234 000304
1323 002236 000377
1324 0C2240 000000
1325 002242 004
1326
1327
1328
1329
1330
1331 002244
1332 002244 160020
1333 002246 160022
1334 002250 160024
1335 002252 160026
1336 002254 160030
1337 002256 160032
1338 002260 160034
1339 002262 160036
1340
1341
1342
1343
1344
1345 002264 137660
1346 002266 177777
1347 002270 000007
1348 002272 177777
1349 002274 166051
1350 002276 000000
1351 002300 077700
1352 002302 000000
1353
1354
1355
1356
1357 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::
TXCHA:: CSRA:: .WORD 160020 ;DHV11-M CSR ADDRESS
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.
;*****
UNBITB:: .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

```

1358 002306 006010          .WORD  DR02MG  ;ADDRESS OF "RBUF" MESSAGE.
1359 002310 006015          .WORD  DR04MG  ;ADDRESS OF "LPR" MESSAGE.
1360 002312 006021          .WORD  DR06MG  ;ADDRESS OF "STAT" MESSAGE.
1361 002314 006026          .WORD  DR10MG  ;ADDRESS OF "LNCTRL" MESSAGE.
1362 002316 006035          .WORD  DR12MG  ;ADDRESS OF "TBUFFAD1" MESSAGE.
1363 002320 006046          .WORD  DR14MG  ;ADDRESS OF "TBUFAD2" MESSAGE.
1364 002322 006057          .WORD  DR16MG  ;ADDRESS OF "TBUFFCT" MESSAGE.
1365
1366
1367
1368
1369 002324 000000          .WORD  DR02MG  ;ADDRESS OF "RBUF" MESSAGE.
1370 002326 000001          .WORD  DR04MG  ;ADDRESS OF "LPR" MESSAGE.
1371 002330 000000          .WORD  DR06MG  ;ADDRESS OF "STAT" MESSAGE.
1372 002332 000000          .WORD  DR10MG  ;ADDRESS OF "LNCTRL" MESSAGE.
1373 002334 000000          .WORD  DR12MG  ;ADDRESS OF "TBUFFAD1" MESSAGE.
1374 002336 000000          .WORD  DR14MG  ;ADDRESS OF "TBUFAD2" MESSAGE.
1375 002340 000000          .WORD  DR16MG  ;ADDRESS OF "TBUFFCT" MESSAGE.
1376 002342 000000
1377 002344 000000
1378 002346 000000
1379 002350 000000
1380 002352 000000
1381
1382
1383
1384
1385 002354 177546          .WORD  DR02MG  ;ADDRESS OF "RBUF" MESSAGE.
1386 002356 000300          .WORD  DR04MG  ;ADDRESS OF "LPR" MESSAGE.
1387 002360 000100          .WORD  DR06MG  ;ADDRESS OF "STAT" MESSAGE.
1388 002362 000074          .WORD  DR10MG  ;ADDRESS OF "LNCTRL" MESSAGE.
1389 002364 000000          .WORD  DR12MG  ;ADDRESS OF "TBUFFAD1" MESSAGE.
1390 002366 000000          .WORD  DR14MG  ;ADDRESS OF "TBUFAD2" MESSAGE.
1391 002370 000170          .WORD  DR16MG  ;ADDRESS OF "TBUFFCT" MESSAGE.
1392 002372 000170
1393 002374 000021
1394 002376 000062
1395
1396
1397
1398
1399 002400 177572          .WORD  DR02MG  ;ADDRESS OF "RBUF" MESSAGE.
1400 002402 000000          .WORD  DR04MG  ;ADDRESS OF "LPR" MESSAGE.
1401 002404 000000          .WORD  DR06MG  ;ADDRESS OF "STAT" MESSAGE.
1402 002406 172340          .WORD  DR10MG  ;ADDRESS OF "LNCTRL" MESSAGE.
1403
1404
1405
1406
1407 002410 000001          .WORD  DR02MG  ;ADDRESS OF "RBUF" MESSAGE.
1408 002412 000002          .WORD  DR04MG  ;ADDRESS OF "LPR" MESSAGE.
1409 002414 000004          .WORD  DR06MG  ;ADDRESS OF "STAT" MESSAGE.
1410 002416 000010          .WORD  DR10MG  ;ADDRESS OF "LNCTRL" MESSAGE.
1411 002420 000020          .WORD  DR12MG  ;ADDRESS OF "TBUFFAD1" MESSAGE.
1412 002422 000040          .WORD  DR14MG  ;ADDRESS OF "TBUFAD2" MESSAGE.
1413 002424 000100          .WORD  DR16MG  ;ADDRESS OF "TBUFFCT" MESSAGE.
1414 002426 000200

```

; 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.

```

; Memory Management Variables and Flags.

```

MMSRO:: .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

```

1415 002430 000400      .WORD 400      ;BIT 8 SET.
1416 002432 001000      .WORD 1000     ;BIT 9 SET.
1417 002434 002000      .WORD 2000     ;BIT 10 SET.
1418 002436 004000      .WORD 4000     ;BIT 11 SET.
1419 002440 010000      .WORD 10000    ;BIT 12 SET.
1420 002442 020000      .WORD 20000    ;BIT 13 SET.
1421 002444 040000      .WORD 40000    ;BIT 14 SET.
1422 002446 100000      .WORD 100000   ;BIT 15 SET.
1423
1424
1425      ;*****
1426      ;*      GPR Save Area Zero.
1427      ;*****
1427 002450      GPRS0B::      ;BASE OF GPR SAVE AREA NUMBER ZERO.
1428 002450 000000      .WORD 0        ;WORD 1, STORAGE FOR R1.
1429 002452 000000      .WORD 0        ;WORD 2, STORAGE FOR R2.
1430 002454 000000      .WORD 0        ;WORD 3, STORAGE FOR R3.
1431 002456 000000      .WORD 0        ;WORD 4, STORAGE FOR R4.
1432 002460 000000      .WORD 0        ;WORD 5, STORAGE FOR R5.
1433
1434      ;*****
1435      ;*      Transmission and Reception Variables, Pointers, and Flags.
1436      ;*****
1437 002462 000000      ERSMRF:: .WORD 0      ;ERROR SUMMARY REPORT FLAGS.
1438 002464      ERCNTB:: .BLKW 16.    ;TABLE OF ERROR COUNTERS.
1439
1440      ;*****
1441      ;      Storage area for the BMP code queue.
1442      ;*****
1443 002524 000000      BMPCQP::      .WORD 0        ;POINTER USED TO ACCESS THE NEXT CELL IN QUE.
1444 002526      BMPCQB::      .BLKW 64.      ;STORAGE FOR 32 CELLS, TEST# PLUS BMP CODE.
1445 002726      BMPCQE::      ;LAST ADDRESS PLUS 2 OF THE BMP CODE QUEUE.
1446      ;*****
1447      ;      General table and buffer area--513 words.
1448      ;*****
1449 002726      BUFBAS::      ;BASE OF MEMORY BUFFER.
1450 002726      ERLTBL::      .BLKW 128.     ;FIRST HALF OF GENERAL TABLE OR BUFFER.
1451 003326      BUFMID::      .BLKW 64.      ;SECOND HALF OF GENERAL TABLE OR BUFFER.
1452 003526      BUF3QT::      .BLKW 64.      ;LAST QUARTER OF THE BUFFER AREA.
1453 003726      BUFEND::      ;END OF GENERAL PURPOSE MEMORY BUFFER.
1454 003726      ENDETB::      .BLKW 16.      ;BUFFER OVERFLOW SPACE.
1455
1468      ERRTBL
1468 003766      ERRTYP::      .WORD 0
1468 003766 000000      ERRNBR::      .WORD 0
1468 003770 000000      ERRMSG::      .WORD 0
1468 003772 000000      ERRBLK::      .WORD 0
1469
1470      .EVEN

```

GPR HANDLING ROUTINES FOR SUBROUTINE CALLS.

```

1472 .SBTTL GPR HANDLING ROUTINES FOR SUBROUTINE CALLS.
1473 ;*****
1474 ;*
1475 ;*   There are 4 routines and macro definitions used for the handling of
1476 ;*   GPR values during subroutine calls within this program. The four
1477 ;*   routines/macro calls have the following names:
1478 ;*
1479 ;*   SAVE - Macro definition used at the beginning of a subroutine to
1480 ;*         save the GPR contents for later restoration.
1481 ;*   PASS - Macro definition used at the end of a subroutine to restore
1482 ;*         the previously saved GPR contents and to leave the contents
1483 ;*         of the specified GPR(s) intact (NOT restored).
1484 ;*   PREG05 - Subroutine which is called from the SAVE and PASS macro
1485 ;*           expansions which actually performs the actions on the GPRs.
1486 ;*
1487 ;*   During a subroutine which uses these GPR save routines the values
1488 ;*   of the GPRs are stored on the stack in the following stack frame:
1489 ;*
1490 ;*       SP      -> RET PC INTO PREG05 ROUTINE.
1491 ;*       SP+2    -> GPR R0 CONTENTS.
1492 ;*       SP+4    -> GPR R1 CONTENTS.
1493 ;*       SP+6    -> GPR R2 CONTENTS.
1494 ;*       SP+8    -> GPR R3 CONTENTS.
1495 ;*       SP+10   -> GPR R4 CONTENTS.
1496 ;*       SP+12   -> GPR R5 CONTENTS.
1497 ;*       SP+14   -> RET PC INTO CALLER OF SUB'TNE WHICH CALLED PREG05.
1498 ;*
1499 ;*   Each level of sub'tne calling uses 8 words of stack overhead.
1500 ;*   The SAVE and PASS macros can also be used in "straight line code"
1501 ;*   to save and restore the GPR values. In any case, after the
1502 ;*   issuing of a PASS call the GPRs will be restored to the values
1503 ;*   they had prior to the last SAVE call (except for the excepted,
1504 ;*   or passed intact, GPRs specified as parameters to the PASS call)
1505 ;*   and the SP will also be restored to its condition before the last
1506 ;*   SAVE call. The programmer must be sure that the SP has the same
1507 ;*   value when the PASS macro is called as it had immediately after
1508 ;*   the SAVE macro was called.
;*****

```

GPR FRAME ACCESS EQUATES

```
1510          .SBTTL GPR FRAME ACCESS EQUATES
1511          ;+++
1512          ;Equates that allow access to the stack frame. These are the
1513          ;offsets into the stack for registers saved during the PREG05
1514          ;routine.
1515          ;---
1516
1517          000036          LPCSLT==          36          ;Offset for last return PC.
1518          000016          PCSLOT==          16          ;Offset for return PC.
1519          000014          R5SLOT==          14          ;Offset for R5.
1520          000012          R4SLOT==          12          ;Offset for R4.
1521          000010          R3SLOT==          10          ;Offset for R3.
1522          000006          R2SLOT==          6           ;Offset for R2.
1523          000004          R1SLOT==          4           ;Offset for R1.
1524          000002          ROSLOT==          2           ;Offset for R0.
```

GLOBAL MACRO DEFINITION - SAVE -

```

1526 .SBTTL GLOBAL MACRO DEFINITION - SAVE -
1527 ;*****
1528 ;* This macro is used at the beginning of a subroutine to save the
1529 ;* contents of the GPRs R0 thru R5.
1530 ;*
1531 ;* INPUTS: SP - Unchanged since subroutine was entered
1532 ;* R5SLOT - Offset to stack slot for R5 (Equated to 14 Octal)
1533 ;*
1534 ;* OUTPUTS: GPR save area on the stack is loaded with the contents of GPRs
1535 ;* TOP OF STACK - Loaded with the return address into PREG05
1536 ;*
1537 ;* CALLING SEQUENCE: SAVE
1538 ;*
1539 ;* COMMENTS: No arguments are allowed.
1540 ;* The PASS macro should be called to restore the GPR values.
1541 ;*
1542 ;* SUBORDINATE ROUTINES CALLED: PREG05.
1543 ;*****
1544
1545 .MACRO SAVE
1546 .LIST JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
1547
1548 .NLIST
1549 .ENDM SAVE
    
```

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
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598

```
.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 -

```

1600 .SBTTL GLOBAL SUBROUTINE - PREG05 -
1601 ;*****
1602 ;* Preserve Registers R0 through R5 for subroutine calls.
1603 ;*
1604 ;* INPUTS: The return address back into the calling routine must be in
1605 ;* GPR R5. (i.e.- Macros use "JSR R5,PREG05".)
1606 ;*
1607 ;* OUTPUTS: Registers R0 through R5 are saved on the stack.
1608 ;*
1609 ;*CALLING SEQUENCE: SAVE ;Macro expansion calls PREG05.
1610 ;* [Subroutine code]...
1611 ;* PASS ;Macro expansion recalls PREG05.
1612 ;*
1613 ;*COMMENTS: This routine is re-entrant.
1614 ;*
1615 ;* Parameters may be passed out of a subroutine by modifying the
1616 ;* register save area on the stack. Use the PASS GPRn macro
1617 ;* to return GPR values intact.
1618 ;* Use the RnSLOT offsets from the SP to pass other parameters.
1619 ;* [Example: MOV VALUE,ROSLOT(SP) ]
1620 ;* Make sure the SP is at its original value when you do this.
1621 ;*
1622 ;*SUBORDINATE ROUTINES CALLED: None.
1623 ;*****
1624
1625 003776 PREG05: ;R5 HAS BEEN LOADED ON THE STACK BY THE SUBROUTINE CALL
1626 003776 010446 MOV R4,-(SP) ;SAVE R4
1627 004000 010346 MOV R3,-(SP) ;SAVE R3
1628 004002 010246 MOV R2,-(SP) ;SAVE R2
1629 004004 010146 MOV R1,-(SP) ;SAVE R1
1630 004006 010046 MOV R0,-(SP) ;SAVE R0
1631 004010 010546 MOV R5,-(SP) ;PUSH RETURN PC ON TOP OF STACK
1632 004012 016605 000014 MOV R5SLOT(SP),R5 ;RESTORE R5 TO VALUE IT HAD BEFORE CALLS
1633
1634 004016 004736 JSR PC,@(SP)+ ;Call the subroutine at the return address
1635 ;from the PREG05 call, putting the present
1636 ;PC on the stack as a return address into
1637 ;this (PREG05) routine.
1638
1639 ;***
1640 ;The following code is executed when the calling routine does a
1641 ;"return" [JSR PC,@(SP)+] using the PC deposited on the stack above.
1642 ;---
1643
1644 004020 012605 PREGRT:: MOV (SP)+,R5 ;Put return PC in R5.
1645 004022 012600 MOV (SP)+,R0 ;Restore R0.
1646 004024 012601 MOV (SP)+,R1 ;Restore R1.
1647 004026 012602 MOV (SP)+,R2 ;Restore R2.
1648 004030 012603 MOV (SP)+,R3 ;Restore R3.
1649 004032 012604 MOV (SP)+,R4 ;Restore R4.
1650
1651 004034 000205 RTS R5 ;Return to the subroutine which called PREG05.
1652 ;restoring R5 in the process.

```


GLOBAL TEXT SECTION

```

1654 .SBTTL GLOBAL TEXT SECTION
1656 ;*****
1657 ;
1658 ; FVTSKL1.P11
1659 ;
1660 ;*****
1662
1663
1664 ;**
1665 ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
1666 ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
1667 ; MORE THAN ONE TEST.
1668 ;--
1669
1670 ;
1671 ; NAMES OF DEVICES SUPPORTED BY PROGRAM
1672 ;
1673 ; DEVTYP <DHV11-M>
1674 004036
1675 004036 104 110 126 L$DVTYP::
1676 004036 061 061 055 .ASCIZ <DHV11-M>
1677 004041 061 061 055
1678 004044 115 000 .EVEN
1679
1680
1681 ; TEST DESCRIPTION
1682 ;
1683 ; DESCRIPT <DHV11-M FUNC TST PART 1>
1684 004046
1685 004046 104 110 126 L$DESC::
1686 004046 061 061 055 .ASCIZ /DHV11-M FUNC TST PA
1687 004051 115 040 106
1688 004054 125 116 103
1689 004057 040 124 123
1690 004062 124 040 120
1691 004065 101 122 124
1692 004070 040 061 000 .EVEN
1693
1684 .EVEN
1685
1692

```

GLOBAL TEXT SECTION

```
1694
1695 ;*****
1696 ;
1697 ;           VDHA.FMT
1698 ;
1699 ;*****
1700
1701
1702
1703 ;
1704 ; FORMAT STATEMENTS USED IN PRINT CALLS
1705 ;
1706
1717
1718
```

GLOBAL TEXT SECTION

```

1727
1728
1729
1730
1731
1732
:*****
:
:          VDHA.MSG
:
:*****
**
1733
1734
1735      .NLIST BIN
1736      .SBTTL  GLOBAL MESSAGE AREA
1737      : ***** FORMAT STATEMENTS *****
1738 004076 MFUNIT:: .ASCIZ /%N%A TESTING UNIT :%D4%A(D)%N/
1739 004134 EF0503:: .ASCIZ /%T%N/
1740 004141 EF0505:: .ASCIZ /%A      %D5%A ILLEGAL INTERRUPTS RECEIVED.%N/
1741 004214 EF1401:: .ASCIZ /%N%A ROM VERSION NUMBERS: PROC_1 = %D2%A(D)  PROC_2 = %D2%A(D)%N/
1742 004316 EF1402:: .ASCIZ /%T%A ROM VERSION NUMBER %T%N/
1743 004353 EF1601:: .ASCIZ /%A      %T%A ABORTED %N/
1744 004377 EF1602:: .ASCIZ /%A      EXPECTED DATA: %06%A (0).%N/
1745 004441 EF1603:: .ASCIZ /%A      ACTUAL DATA:   %06%A (0).%N/
1746 004503 EF1604:: .ASCIZ /%A      BAD BIT(S) IN DEVICE %T%A REGISTER FOR LINE %D2%A (D).%N/
1747 004600 EF3001:: .ASCIZ /%A      EXPECTED OR CORRECT VALUE: %03%N/
1748 004647 EF3002:: .ASCIZ /%A      ACTUAL OR MEASURED VALUE: %03%N/
1749 004716 EF9001:: .ASCIZ /%A      UNEXPECTED %T%A FOUND IN RECEIVE CHAR FIFO:%N/
1750 005000 EF9002:: .ASCIZ /%A      CODE IS ASSOCIATED WITH LINE: %D2%A(D)%N/
1751 005057 EF9003:: .ASCIZ /%A      CODE IS: %03%A(0)%N/
1752 005113 EF9004:: .ASCIZ /%A      %T%A VALUE: %03%A(0)%N/
1753 005150 EF9005:: .ASCIZ /%A      %T%A VALUE: NONE%N/
1754 005201 EF9006:: .ASCIZ /%A      %T%A %D2%A(D)%N/
1755 005225 EF9010:: .ASCIZ /%A      NUMBER OF ERRORS DETECTED ON LINE %D2%A(D) IS %D5%A(D)%N/
1756 005324 EF9016:: .ASCIZ /%A      UNEXPECTED %T%A FOR LINE %D2%A(D) IN FIFO AFTER RESET:%N/
1757 005421 EF9017:: .ASCIZ /%A      %T%A (WITH ERROR FLAGS) IS %06%A(0)%N/
1758 005475 EF9018:: .ASCII /%A      %T%A IN SELFTEST CODE FIFO SLOT FOR LINE %D2/
1759 005555      .ASCIZ /%A(D) AFTER RESET.%N/
1760 005602 EF9019:: .ASCIZ /%A      %T%A %06%A(0)%N/
1761 005626 EF9301:: .ASCIZ /%A      %T%D2%A(D), BMP CODE REPORTED :%03%A(0)%N/
1762 005704 EF9302:: .ASCIZ /%A      OVERFLOW OCCURRED (MORE THAN 31 BMP CODES FOUND IN QUEUE)%N/
1763      :***** ERROR MESSAGES *****
1764 006004 DR00MG:: .ASCIZ /CSR/
1765 006010 DR02MG:: .ASCIZ /RBUF/
1766 006015 DR04MG:: .ASCIZ /LPR/
1767 006021 DR06MG:: .ASCIZ /STAT/
1768 006026 DR10MG:: .ASCIZ /LNCTRL/
1769 006035 DR12MG:: .ASCIZ /TBUFFAD1/
1770 006046 DR14MG:: .ASCIZ /TBUFFAD2/
1771 006057 DR16MG:: .ASCIZ /TBUFFCT/
1772 006067 EM0103:: .ASCIZ /DEVICE REGISTER ACCESS ERRORS/
1773 006125 EM0201:: .ASCIZ /MASTER RESET (PERFORM SELFTEST) TEST /
1774 006173 EM0202:: .ASCIZ / MASTER RESET BIT DID NOT CLEAR AFTER BOARD RESET./
1775 006257      .ASCIZ / WAITED 5 SECONDS. BIT DEFECTIVE OR FIRMWARE HUNG./
1776 006346 EM0203:: .ASCIZ / MASTER RESET BIT CLEAR IMMEDIATELY AFTER BOARD RESET./
1777 006436      .ASCIZ / BIT DEFECTIVE OR BOARD FIRMWARE ERROR./
1778 006511 EM0204:: .ASCIZ \ MR BIT WENT CLEAR WITHIN 1/2 SECOND OF BOARD RESET.\
1779 006577      .ASCIZ / BIT DEFECTIVE OR SELFTEST WAS (INCORRECTLY) SKIPPED./
1780 006670 EM0301:: .ASCIZ /MASTER RESET (SKIP SELFTEST) TEST /
1781 006733 EM0302:: .ASCIZ / MR BIT CLR WITHIN 10 MILISECOND AFTER BOARD RESET./
1782 007020      .ASCIZ / BIT DEFECTIVE OR BOARD FIRMWARE ERROR./
1783 007073 EM0303:: .ASCIZ \ MR BIT WENT CLEAR 1/5 TO 5 SECONDS AFTER RESET.\

```

GLOBAL MESSAGE AREA

```

1784 007155      .ASCIZ / SELFTEST DID NOT GET SKIPPED (SHOULD HAVE BEEN SKIPPED)./
1785 007252 EM0401:: .ASCIZ /RBUF REGISTER RX CHARACTER FIELD TEST /
1786 007321 EM0402:: .ASCIZ / IMPROPER CODE FOUND IN RX FIFO AFTER DUT RESET./
1787 007403      .ASCIZ / EXPECTED: SELFTEST CODE, ACTUAL: IMPROPER CODE./
1788 007471 EM0501:: .ASCIZ /RBUF REGISTER ERROR FLAGS FIELD TEST /
1789 007537 EM0502:: .ASCIZ / RX ERROR FLAG(S) FOUND CLEAR ON SELFTEST CODE./
1790 007620      .ASCIZ / EXPECTED: ALL ERROR FLAGS SET, ACTUAL: FLAG(S) CLEAR./
1791 007713 EM0525:: .ASCIZ / RX INTERRUPT(S) RECEIVED WITH RX INTERRUPTS DISABLED./
1792 010003 EM0526:: .ASCIZ / TX INTERRUPT(S) RECEIVED WITH TX INTERRUPTS DISABLED./
1793 010073 EM0601:: .ASCIZ /CSR RX.DATA.AVAIL BIT TEST /
1794 010127 EM0602:: .ASCIZ / RX.DATA.AVAIL BIT FOUND CLEAR AFTER RESET COMPLETION./
1795 010217      .ASCIZ / EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO./
1796 010307 EM0603:: .ASCIZ / RX.DATA.AVAIL BIT COULD NOT BE CLEARED BY PURGING FIFO./
1797 010401      .ASCIZ / 600 CHARS READ FROM FIFO WITHOUT R.D.A BIT CLEARING./
1798 010472 EM0701:: .ASCIZ /RBUF RX.DATA.VALID BIT TEST /
1799 010527 EM0702:: .ASCIZ / RX.DATA.VALID BIT FOUND CLEAR AFTER RESET COMPLETION./
1800 010617      .ASCIZ / EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO./
1801 010707 EM0703:: .ASCIZ / RX.DATA.VALID BIT COULD NOT BE CLEARED BY PURGING FIFO./
1802 011001      .ASCIZ / 600 CHARS READ FROM FIFO WITHOUT R.D.V BIT CLEARING./
1803 011072 EM0801:: .ASCIZ /RBUF RX.LINE.NUMBER FIELD TEST /
1804 011132 EM0802:: .ASCIZ / LINE NUMBER IS WRONG ON A SELFTEST CODE./
1805 011205 EM0901:: .ASCIZ /CHECK FOR BMP_CODES TEST/
1806 011236 EM0902:: .ASCIZ /UNEXPECTED BMP_CODES FOUND./
1807 011272 EM1001:: .ASCIZ /DIAGNOSTIC FAIL (SKP SELFTEST) TEST/
1808 011336 EM1002:: .ASCIZ / SKIP SELF-TEST TOOK TOO LONG TO COMPLETE, > 50 MS./
1809 011423 EM1003:: .ASCIZ / SKIP SELF-TEST COMPLETED TOO SOON, < 10 MS./
1810 011501 EM1101:: .ASCIZ /SKIP SELF-TEST TEST/
1811 011525 EM1201:: .ASCIZ /SELF-TEST TEST/
1812 011544 EM1202:: .ASCIZ / SELF-TEST TOOK TOO LONG TO COMPLETE, > 3 SECONDS./
1813 011630 EM1203:: .ASCIZ \ SELF-TEST COMPLETED TOO SOON, < 1/2 SECOND.\
1814 011706 EM1204:: .ASCIZ / SELF-TEST DID NOT EXECUTE/
1815 011742 EM1205:: .ASCIZ / DIAG_FAIL BIT BAD/
1816 011766 EM1301:: .ASCIZ /FAIL SELF-TEST TEST/
1817 012012 EM1302:: .ASCIZ / SELF-TEST ERROR REPORTING BAD/
1818 012051 EM1401:: .ASCIZ /ROM VERSION_NUMBER TEST/
1819 012101 EM1402:: .ASCIZ / FIFO EMPTY, ONE OR MORE ROM VERSION_NUMBERS MISSING/
1820 012167 EM1403:: .ASCIZ / ROM VERSION_NUMBER FOUND OUT OF SEQUENCE/
1821 012242 EM1404:: .ASCIZ / ONE OR MORE ROM VERSION_NUMBERS MISSING/
1822 012314 EM1405:: .ASCIZ / PROC_1/
1823 012327 EM1406:: .ASCIZ / PROC_2/
1824 012342 EM1407:: .ASCIZ /NOT FOUND/
1825 012354 EM1408:: .ASCIZ /FOUND/
1826 012362 EM1601:: .ASCIZ /TIMEOUT OCCURRED WAITING FOR MASTER RESET TO CLEAR/
1827 012445 EM1604:: .ASCIZ \DEVICE REGISTER WORD READ/WRITE TEST \
1828 012513 EM1701:: .ASCIZ \DEVICE REGISTER WORD READ/MODIFY/WRITE TEST \
1829 012570 EM1801:: .ASCIZ \DEVICE REGISTER BYTE READ/WRITE TEST \
1830 012636 EM1901:: .ASCIZ \DEVICE REGISTER BYTE READ/MODIFY/WRITE TEST \
1831 012713 EM2001:: .ASCIZ /DEVICE STAT REGISTER ID BIT TEST /
1832 012755 EM2002:: .ASCIZ /ID BIT BAD. EXPECTED: CLEAR, ACTUAL: SET./
1833 013030 EM2101:: .ASCIZ \NO TX_DATA_VALID/NO TX_ACTION TEST\
1834 013073 EM2102:: .ASCIZ / TX_ACTION FOUND AFTER INVALID DATA WORD WRITTEN TO LINE: /
1835 013167 EM2201:: .ASCIZ \TX_DATA_VALID/TX_ACTION TEST\
1836 013224 EM2202:: .ASCIZ / NO TX_ACTION FOUND AFTER VALID DATA WORD TX'D ON LINE: /
1837 013316 EM2203:: .ASCIZ / INCORRECT LINE NUMBER FOUND WITH TX_ACT AFTER DATA TX'D ON LINE : /
1838 013423 EM2301:: .ASCIZ /TX_ENABLE (INACTIVE) BIT TEST/
1839 013461 EM2302:: .ASCIZ / TX_ENABLE BIT BAD ON LINE: /
1840 013517 EM2401:: .ASCIZ /TX_ENABLE (ACTIVE) BIT TEST/

```

GLOBAL MESSAGE AREA

```

1841 013553 EM2601:: .ASCIZ /RECEIVE INTERRUPT TEST /
1842 013603 EM2602:: .ASCIZ / NO RX INT GENERATED (DATA_VALID SET, RX INTS ENABLED)./
1843 013674 EM2603:: .ASCIZ / NO RX INT GENERATED (NO CODES IN FIFO AFTER RESET)./
1844 013762 EM2604:: .ASCIZ / NO RX INT GENERATED (RX_DATA_AVAIL CLR, RX INTS ENABLED)./
1845 014056 EM2605:: .ASCIZ / RX INTERRUPT GENERATED WITH RX_DATA_AVAIL CLEAR./
1846 014141 EM2606:: .ASCIZ /TRANSMIT INTERRUPT TEST ERROR:/
1847 014200 EM2607:: .ASCIZ / TX_ACTION SET REPEATEDLY AFTER BOARD RESET, NO DATA SENT./
1848 014274 EM2608:: .ASCIZ / TX_ACTION STUCK SET AFTER BOARD RESET./
1849 014345 EM2609:: .ASCIZ / TX INTERRUPT GENERATED WITH TX_ACTION CLEAR./
1850 014424 EM2610:: .ASCIZ / NO TX INTERRUPT WITH TX_ACTION SET AND TX INTS ENABLED./
1851 014516 EM2611:: .ASCIZ / TX_ACTION NOT SET AFTER CHARS SENT ON ALL LINES./
1852 014601 EM3001:: .ASCIZ /INTERRUPT BR LEVEL TEST /
1853 014632 EM3002:: .ASCIZ / NO RX_DATA_AVAIL FROM SELFTTEST CODES IN FIFO AFTER RESET./
1854 014726 EM3003:: .ASCIZ / TX INTERRUPT GENERATED AT WRONG BR LEVEL:/
1855 015002 EM3004:: .ASCIZ / RX INTERRUPT GENERATED AT WRONG BR LEVEL:/
1856 015056 EM3005:: .ASCIZ / TX INTERRUPT GIVEN PRECEDENCE OVER SIMULTANEOUS RX INT./
1857 015150 EM3101:: .ASCIZ /DIAGNOSTIC FIELD (BMP) TEST/
1858 015204 EM3102:: .ASCIZ / DIAGNOSTIC FIELD BAD ON LINE: /
1859 015245 EM9009:: .ASCIZ /EXPECTED OR CORRECT/
1860 015271 EM9010:: .ASCIZ /ACTUAL OR MEASURED /
1861 015315 EM9014:: .ASCIZ /SUMMARY REPORTS FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:/
1862 015411 EM9017:: .ASCIZ / FIFO WILL NOT PURGE (DATA.VALID STUCK SET),/
1863 015466 .ASCIZ / REMAINDER OF TEST SKIPPED./
1864 015522 EM9018:: .ASCIZ /NO CODE/
1865 015532 EM9019:: .ASCIZ /NON-SELFTTEST/
1866 015547 EM9020:: .ASCIZ /SELFTTEST ERROR CODE/
1867 015573 EM9022:: .ASCIZ /DATA CHARACTER/
1868 015612 EM9023:: .ASCIZ /MODEM STATUS CODE/
1869 015634 EM9024:: .ASCIZ /SELFTTEST CODE/
1870 015652 EM9026:: .ASCIZ / LPR CONTENTS: /
1871 015676 EM9301:: .ASCIZ /BMP CODE REPORT/
1872 015716 EM9302:: .ASCIZ /BMP CODE FOUND IN TEST /
1873 015746 EM9303:: .ASCIZ /THE LAST BMP CODE WAS FOUND IN TEST /
1874 016013 EM9304:: .ASCIZ /UNEXPECTED BMP CODES FOUND DURING THIS PASS/
1875 .EVEN
1876 .LIST BIN

```

GLOBAL MESSAGE AREA

1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894

```
*****  
:                                     :  
:               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 -

```

1896 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0101 -
1897 ;*****
1898 ;* This is an error reporting subroutine which prints additional error
1899 ;* information if an error is detected in TEST 1 (Register Address
1900 ;* Access Test). This subroutine reports the type of access (Read or
1901 ;* Write or both) which caused a bus time-out trap (004 trap).
1902 ;* A message indicating that the DHV may be at the wrong Q-bus address
1903 ;* is also printed.
1904 ;*
1905 ;* INPUTS: R5 - Error flag word.
1906 ;* If bit 0 is set, a read error occurred.
1907 ;* If bit 1 is set, a write error occurred.
1908 ;*
1909 ;* OUTPUTS: Messages are printed at the operator console.
1910 ;*
1911 ;* CALLING SEQUENCE: Include the label "ER0101" as the message pointer
1912 ;* parameter in the DRS error report macro call.
1913 ;*
1914 ;* COMMENTS:
1915 ;*
1916 ;* SUBORDINATE ROUTINES USED: None.
1917 ;*****
1918
1919 016070 BGNMSG ER0101
1920 016070 ER0101::
1920 016070 004537 003776 SAVE JSR ;SAVE THE GPR CONTENTS.
1921 ;CALL REGISTER SAVE SUBRT.
1922 016074 032705 000001 BIT #BIT0,R5 ;TEST FOR READ ERROR.
1923 016100 001410 BEQ 2$ ;SKIP READ ERROR MSG IF NO READ ERROR.
1924 016102 PRINTB #MSG1 ;PRINT READ ERROR MESSAGE.
1924 016102 012746 016174 MOV #MSG1,-(SP)
1924 016106 012746 000001 MOV #1,-(SP)
1924 016112 010600 MOV SP,R0
1924 016114 104414 TRAP C$PNTB
1924 016116 062706 000004 ADD #4,SP
1925 016122 032705 000002 2$: BIT #BIT1,R5 ;TEST FOR WRITE ERROR.
1926 016126 001410 BEQ 4$ ;SKIP WRITE ERROR MSG IF NO WRITE ERROR.
1927 016130 PRINTB #MSG2 ;PRINT WRITE ERROR MESSAGE.
1927 016130 012746 016252 MOV #MSG2,-(SP)
1927 016134 012746 000001 MOV #1,-(SP)
1927 016140 010600 MOV SP,R0
1927 016142 104414 TRAP C$PNTB
1927 016144 062706 000004 ADD #4,SP
1928 016150 4$: PRINTX #MSG3 ;SUGGEST THAT DHV MAY BE AT WRONG ADDRESS.
1928 016150 012746 016331 MOV #MSG3,-(SP)
1928 016154 012746 000001 MOV #1,-(SP)
1928 016160 010600 MOV SP,R0
1928 016162 104415 TRAP C$PNTX
1928 016164 062706 000004 ADD #4,SP
1929 016170 PASS ;RESTORE THE GPR CONTENTS.
1929 016170 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
1930 016172 ENDMSG
1930 016172 104423 L10002: TRAP C$MSG
1931
1932 016174 045 101 102 MSG1:: .ASCIZ /*ABUS TIME-OUT TRAP CAUSED BY READ ATTEMPT.*/

```

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			
1933	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		
1934	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	

1935
1936

.EVEN

GLOBAL ERROR REPORTING ROUTINE

- ER0201 -

```

1938 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0201 -
1939 ;*****
1940 ;* This is an error reporting subroutine which prints 2 contiguous
1941 ;* ASCII error messages. The address of the first message is passed
1942 ;* as an input parameter and the address of the second is found by
1943 ;* searching for the end of the first message.
1944 ;*
1945 ;* INPUTS: R1 - Address of the first message to print.
1946 ;*
1947 ;* OUTPUTS: A messages is printed at the operator console.
1948 ;*
1949 ;* CALLING SEQUENCE: Load the address of the first message in R1.
1950 ;* Include the label "ER0201" as the message pointer
1951 ;* parameter in the Diag Super error report macro call.
1952 ;*
1953 ;* COMMENTS: The message is printed as Basic error information.
1954 ;* The second message should follow the first one in the program
1955 ;* memory. Each message should be defined using .ASCIZ
1956 ;*
1957 ;* SUBORDINATE ROUTINES USED: None.
1958 ;*****
1959
1960 016406 BGNMSG ER0201
1961 016406 004537 003776 ER0201::
1962 016406 ;SAVE THE GPR CONTENTS.
1963 016412 010102 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
1964 016414 105722
1965 016416 001376
1966
1967 016420 PRINTB #EF0503,R1 ;PRINT THE FIRST MESSAGE.
1968 016420 010146 MOV R1,-(SP)
1969 016422 012746 004134 MOV #EF0503,-(SP)
1970 016426 012746 000002 MOV #2,-(SP)
1971 016432 010600 MOV SP,R0
1972 016434 104414 TRAP C#PNTB
1973 016436 062706 000006 ADD #6,SP
1974 016442 PRINTB #EF0503,R2 ;PRINT THE SECOND MESSAGE.
1975 016442 010246 MOV R2,-(SP)
1976 016444 012746 004134 MOV #EF0503,-(SP)
1977 016450 012746 000002 MOV #2,-(SP)
1978 016454 010600 MOV SP,R0
1979 016456 104414 TRAP C#PNTB
1980 016460 062706 000006 ADD #6,SP
1981
1982 016464 PASS ;RESTORE THE GPR CONTENTS.
1983 016464 004736 JSR PC,#(SP)+ ;RETURN TO PREG05 SUBRT.
1984
1985 ENDMSG
1986
1987 L10003: TRAP C#MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER0503 -

1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991

```
.SBTTL GLOBAL ERROR REPORTING ROUTINE          - ER0503 -
;*****
;*      This is an error reporting subroutine which prints an additional error
;*      message whose address is passed as an input parameter.
;*
;* INPUTS:      R1 - Address of the message to print.
;*
;* OUTPUTS:     A messages is printed at the operator console.
;*
;* CALLING SEQUENCE:  Load the address of the message in R1.
;*                   Include the label "ER0503" as the message pointer
;*                   parameter in the Diag Super error report macro call.
;*
;* COMMENTS:     The message is printed as Basic error information.
;*
;* SUBORDINATE ROUTINES USED: None.
;*****
```

1992 016470
016470

BGNMSG ER0503

ER0503::

1993

1994 016470
016470 010146
016472 012746 004134
016476 012746 000002
016502 010600
016504 104414
016506 062706 000006

PRINTB #EF0503,R1 ;PRINT THE MESSAGE.

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

1995

1996 016512
016512
016512 104423

ENDMSG

```
L10004:
TRAP C#MSG
```

GLOBAL ERROR REPORTING ROUTINE

- ER0504 -

1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

```
.SBTTL GLOBAL ERROR REPORTING ROUTINE          - ER0504 -
;*****
;*      This is an error reporting subroutine which prints additional error
;*      messages when illegal interrupts are received.
;*
;* INPUTS:      R1 - Address of the message to print.
;*              R2 - Number of illegal interrupts received.
;*
;* OUTPUTS:     Messages are printed at the operator console.
;*
;* CALLING SEQUENCE:  Load the address of the message in R1.
;*                   Load the number of illegal ints in R2.
;*                   Include the label "ER0504" as the message pointer
;*                   parameter in the Diag Super error report macro call.
;*
;* COMMENTS:
;*
;* SUBORDINATE ROUTINES USED: None.
;*****
```

2018 016514
016514
2019
2020 016514
016514 010146
016516 012746 004134
016522 012746 000002
016526 010600
016530 104414
016532 062706 000006
2021 016536
016536 010246
016540 012746 004141
016544 012746 000002
016550 010600
016552 104415
016554 062706 000006
2022
2023 016560
016560
016560 104423

```
BGNMSG ER0504
ER0504::
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
PRINTX #EF0505,R2 ;PRINT THE NUMBER OF INTS RECEIVED.
MOV R2,-(SP)
MOV #EF0505,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C#PNTX
ADD #6,SP
ENDMSG
L10005: TRAP C#MSG
```

GLOBAL ERROR REPORTING ROUTINE

- ER1401 -

```

2025 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER1401 -
2026 ;*****
2027 ;* This is an error reporting subroutine which prints additional error
2028 ;* information if an error is detected in the Rom version test.
2029 ;* This subroutine analyses the input parameters which contain the
2030 ;* ROM version numbers for PROC_1 and PROC_2 and reports the appropriate
2031 ;* error message to the operator.
2032 ;*
2033 ;* INPUTS: R1 - Contains the address of the first message to be reported.
2034 ;* R3 - Contains the ROM version number of PROC_1.
2035 ;* R4 - Contains the ROM version number of PROC_2.
2036 ;*
2037 ;* OUTPUTS: Basic and extended error messages are reported at the
2038 ;* operators console.
2039 ;*
2040 ;* CALLING SEQUENCE: Include the label "ER1401" as the message pointer
2041 ;* parameter in the DRS error report macro call.
2042 ;*
2043 ;* COMMENTS:
2044 ;*
2045 ;* SUBORDINATE ROUTINES USED: None.
2046 ;*****
2047
2048 016562 BGNMSG ER1401
2049 016562 ER1401::
2050 016562 PRINTB #EF0503,R1 ;REPORT THE ERROR MESSAGE PASSED IN.
2051 016562 010146 MOV R1,-(SP)
2052 016564 012746 004134 MOV #EF0503,-(SP)
2053 016570 012746 000002 MOV #2,-(SP)
2054 016574 010600 MOV SP,R0
2055 016576 104414 TRAP C:PNTB
2056 016600 062706 000006 ADD #6,SP
2057
2058 ;*
2059 ;* Determine which Rom version number(s) are missing.
2060 ;*
2061
2062 016604 012705 000143 MOV #99.,R5 ;GET INVALID ROM NUMBER.
2063 016610 012701 012314 MOV #EM1405,R1 ;SELECT PROC_1 MESSAGE.
2064 016614 012702 012342 MOV #EM1407,R2 ;SELECT THE "NOT FOUND" MESSAGE.
2065 016620 120305 CMPB R3,R5 ;CHECK PROC_1 ROM VERSION NUMBER.
2066 016622 001402 BEQ 2# ;GO REPORT PROC_1 CODE NOT FOUND.
2067 016624 012702 012354 MOV #EM1408,R2 ;SELECT "FOUND" MESSAGE.
2068 016630 004737 016662 2#: JSR PC,50# ;GO REPORT MESSAGE.
2069
2070 016634 012701 012327 MOV #EM1406,R1 ;SELECT PROC_2 MESSAGE.
2071 016640 012702 012342 MOV #EM1407,R2 ;SELECT THE "NOT FOUND" MESSAGE.
2072 016644 120405 CMPB R4,R5 ;CHECK PROC_2 ROM VERSION NUMBER.
2073 016646 001402 BEQ 4# ;GO REPORT PROC_2 CODE NOT FOUND.
2074 016650 012702 012354 MOV #EM1408,R2 ;SELECT "FOUND" MESSAGE.
2075 016654 004737 016662 4#: JSR PC,50# ;GO REPORT THE MESSAGE.
2076 016660 000413 BR 60# ;EXIT.
2077
2078 016662 50#: PRINTX #EF1402,R1,R2 ;REPORT THE MESSAGE.
2079 016662 010246 MOV R2,-(SP)
2080 016664 010146 MOV R1,-(SP)

```

GLOBAL ERROR REPORTING ROUTINE

- ER1401 -

016666 012746 004316
016672 012746 000003
016676 010600
016700 104415
016702 062706 000010
2073 016706 000207
2074 016710
016710
016710 104423

604: RTS PC
ENDMSG

;RETURN.

L10006:

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

TRAP C#MSG

GLOBAL ERROR REPORTING ROUTINE

- ER1601 -

2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099 016712
016712
2100
2101 016712 016304 002304
2102
2103 016716
016716 010546
016720 010446
016722 012746 004503
016726 012746 000003
016732 010600
016734 104414
016736 062706 000010
2104 016742
016742 010246
016744 012746 004377
016750 012746 000002
016754 010600
016756 104415
016760 062706 000006
2105 016764
016764 010146
016766 012746 004441
016772 012746 000002
016776 010600
017000 104415
017002 062706 000006
2106 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 the device register
;* access tests.
;* This subroutine reports the actual and expected from 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.
;*              R4MATBB - Label at base of register message address table.
;*
;* OUTPUTS:     Messages are printed at the operators console.
;*
;* CALLING SEQUENCE: Include the lable "ER1601" as the message pointer
;*                  parameter in the DRS error report macro call.
;*
;* COMMENTS:
;*
;* SUBORDINATE ROUTINES CALLED: NONE
;*****
          BGNMSG ER1601
                                ER1601::
2101          MOV    R4MATBB(R3),R4 ;FETCH ADDRESS OF REGISTER NAME MESSAGE.
2103          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$PNTB
                                ADD    #10,SP
2104          PRINTX #EF1602,R2 ;PRINT THE EXPECTED DATA.
                                MOV    R2,-(SP)
                                MOV    #EF1602,-(SP)
                                MOV    #2,-(SP)
                                MOV    SP,R0
                                TRAP   C$PNTX
                                ADD    #6,SP
2105          PRINTX #EF1603,R1 ;PRINT THE ACTUAL DATA.
                                MOV    R1,-(SP)
                                MOV    #EF1603,-(SP)
                                MOV    #2,-(SP)
                                MOV    SP,R0
                                TRAP   C$PNTX
                                ADD    #6,SP
2106          ENDMSG
                                L10007:
                                TRAP   C$MSG
```

GLOBAL ERROR REPORTING ROUTINE

- ER1603 -

```

2108 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER1603 -
2109 ;*****
2110 ;* This error reporting routine is used to print out a basic error
2111 ;* message, along with a message informing the operator which test is
2112 ;* about to be aborted.
2113 ;*
2114 ;* INPUTS: R1 - Contains the address of the message to be printed.
2115 ;* ERRMSG - Contains the address of the message that indicates
2116 ;* the test that is being performed, eg DMA, BREAK etc.
2117 ;*
2118 ;* OUTPUTS: Messages are printed at the operators console.
2119 ;* "testname TEST ABORTED"
2120 ;*
2121 ;* CALLING SEQUENCE: Include the lable "ER1603" as the message pointer
2122 ;* parameter in the DRS error report macro call.
2123 ;*
2124 ;* COMMENTS:
2125 ;*
2126 ;*
2127 ;* SUBORDINATE ROUTINES CALLED: None.
2128 ;*****
2129 017010 BGNMSG ER1603
2130 017010 ER1603::
017010 004537 003776 SAVE ;SAVE THE CONTENTS OF THE GPRS.
;SAVE THE CONTENTS OF THE GPRS.
017010 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2131
2132 017014 PRINTB #EF0503,R1 ;PRINT BASIC MESSAGE ON OPERATORS CONSOLE.
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
2133
2134 017036 013702 003772 MOV ERRMSG,R2 ;GET THE "TEST MESSAGE".
2135 017042 PRINTB #EF1601,R2 ;PRINT "TEST ABORTED" MESSAGE.
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
2136
2137 017064 PASS ;RESTORE THE CONTENTS OF THE GPRS.
017064 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2138 017066 ENDMSG
017066 L10010:
017066 104423 TRAP C#MSG
    
```

GLOBAL ERROR REPORTING ROUTINE

- ER3001 -

```

2140 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER3001 -
2141 ;*****
2142 ;* This is an error reporting subroutine which is intended for use in the
2143 ;* Interrupt BR Level Test. It reports additional information when an
2144 ;* interrupt has occurred at the wrong BR level.
2145 ;*
2146 ;* INPUTS: R1 - Address of message to print first.
2147 ;* R4 - BR level at which the int request occurred.
2148 ;* R5 - Expected or correct BR level for the DUT.
2149 ;*
2150 ;* OUTPUTS: A messages is printed at the operator console.
2151 ;*
2152 ;* CALLING SEQUENCE: Include the label "ER3001" as the message pointer
2153 ;* parameter in the Diag Super error report macro call.
2154 ;*
2155 ;* COMMENTS: The message is printed as Basic and Extended error information.
2156 ;*
2157 ;* SUBORDINATE ROUTINES USED: None.
2158 ;*****
2159
2160 017070 BGNMSG ER3001
2161 017070 ER3001::
2162 017070 PRINTB #EF0503,R1 ;PRINT THE FIRST LINE OF THE MESSAGE.
2163 017070 010146 MOV R1,-(SP)
2164 017072 012746 004134 MOV #EF0503,-(SP)
2165 017076 012746 000002 MOV #2,-(SP)
2166 017102 010600 MOV SP,R0
2167 017104 104414 TRAP C#PNTB
2168 017106 062706 000006 ADD #6,SP
2169 017112 PRINTX #EF3001,R5 ;REPORT EXPECTED BR LEVEL.
2170 017112 010546 MOV R5,-(SF)
2171 017114 012746 004600 MOV #EF3001,-(SP)
2172 017120 012746 000002 MOV #2,-(SP)
2173 017124 010600 MOV SP,R0
2174 017126 104415 TRAP C#PNTX
2175 017130 062706 000006 ADD #6,SP
2176 017134 PRINTX #EF3002,R4 ;REPORT ACTUAL BR LEVEL.
2177 017134 010446 MOV R4,-(SP)
2178 017136 012746 004647 MOV #EF3002,-(SP)
2179 017142 012746 000002 MOV #2,-(SP)
2180 017146 010600 MOV SP,R0
2181 017150 104415 TRAP C#PNTX
2182 017152 062706 000006 ADD #6,SP
2183
2184 017156 ENDMSG
2185 017156 L10011:
2186 017156 104423 TRAP C#MSG

```


GLOBAL ERROR REPORTING ROUTINE

- ER9004 -

2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189 017160
017160
2190
2191 017160
017160 012746 015315
017164 012746 004134
017170 012746 000002
017174 010600
017176 104414
017200 062706 000006
2192 017204 005002
2193 017206 013703 002462
2194 017212 005004
2195 017214 000241
2196 017216 006003
2197 017220 103013
2198 017222
017222 016446 002464
017226 010246
017230 012746 005225
017234 012746 000003
017240 010600
017242 104415
017244 062706 000010
2199 017250 012405
2200 017252 005202
2201 017254 005703
2202 017256 001356
2203
2204 017260
017260
017260 104423

.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.
;*****

BGNMSG ER9004

ER9004::

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)

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

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

ENDMSG

L10012:
TRAP C#MSG

GLOBAL ERROR REPORTING ROUTINE

- ER9007 -

2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227 017262
017262
2228
2229 017262 042703 177760
2230 017266
017266 010346
017270 010146
017272 012746 005475
017276 012746 000003
017302 010600
017304 104414
017306 062706 000010
2231 017312
017312 010246
017314 010146
017316 012746 005421
017322 012746 000003
017326 010600
017330 104415
017332 062706 000010
2232
2233 017336
017336
017336 104423

.SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9007 -
;*****
;* This is an error reporting subroutine which is used to report that
;* something other than a selftest code was found in a selftest code
;* FIFO slot during the removal of the selftest codes from the FIFO.
;* This routine is used by the RSTRPT routine.
;*
;* INPUTS: R1 - Address of error message qualifier string.
;* R2 - Incorrect code as read from the selftest code FIFO slot.
;* R3 - Line number associated with the selftest FIFO slot.
;*
;* OUTPUTS: A message is printed at the operator console.
;*
;* CALLING SEQUENCE: Include the label "ER9007" 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 ER9007

ER9007::

BIC #177760,R3 ;REMOVE ALL BUT LINE # BITS FROM LINE # WORD.
PRINTB #EF9018,R1,R3 ;REPORT SECONDARY ERROR MESSAGE.

MOV R3,-(SP)
MOV R1,-(SP)
MOV #EF9018,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C#PNTB
ADD #10,SP
MOV R2,-(SP)
MOV R1,-(SP)
MOV #EF9017,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C#PNTX
ADD #10,SP

PRINTX #EF9017,R1,R2 ;REPORT THE ACTUAL INCORRECT CODE.

ENDMSG

L10013:
TRAP C#MSG

GLOBAL ERROR REPORTING ROUTINE

- ER9008 -

2235
 2236
 2237
 2238
 2239
 2240
 2241
 2242
 2243
 2244
 2245
 2246
 2247
 2248
 2249
 2250
 2251
 2252
 2253
 2254 017340
 017340
 2255
 2256
 2257
 2258
 2259
 2260 017340 010203
 2261 017342 000303
 2262 017344 042703 177760
 2263 017350
 017350 010346
 017352 010146
 017354 012746 005324
 017360 012746 000003
 017364 010600
 017366 104414
 017370 062706 000010
 2264 017374
 017374 010246
 017376 010146
 017400 012746 005421
 017404 012746 000003
 017410 010600
 017412 104415
 017414 062706 000010
 2265
 2266 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 -

```

2268 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9101 -
2269 ;*****
2270 ;* This is a general error reporting subroutine which reports a message
2271 ;* which takes a single, 2 digit decimal argument after the end of an
2272 ;* ASCII message.
2273 ;*
2274 ;* INPUTS: R1 - Value to be printed after msg as 2 decimal digits.
2275 ;* R2 - Address of message to print first.
2276 ;*
2277 ;* OUTPUTS: A messages is printed at the operator console.
2278 ;*
2279 ;* CALLING SEQUENCE: Include the label "ER9101" as the message pointer
2280 ;* parameter in the Diag Super error report macro call.
2281 ;*
2282 ;* COMMENTS: The message is printed as Basic error information.
2283 ;*
2284 ;* SUBORDINATE ROUTINES USED: None.
2285 ;*****
2286
2287 017422 BGNMSG ER9101
2288 017422 ER9101::
2289 017422 PRINTB #EF9006,R2,R1 ;REPORT THE STRING FOLLOWED BY THE NUMBER.
2290 017422 010146 MOV R1,-(SP)
2291 017424 010246 MOV R2,-(SP)
2292 017426 012746 005201 MOV #EF9006,-(SP)
2293 017432 012746 000003 MOV #3,-(SP)
2294 017436 010600 MOV SP,R0
2295 017440 104414 TRAP C#PNTB
2296 017442 062706 000010 ADD #10,SP
2297
2298
2299
2300
2301 017446 ENDMSG
2302 017446 L10015:
2303 017446 104423 TRAP C#MSG
    
```

GLOBAL ERROR REPORTING ROUTINE

- ER9301 -

2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313 017450
017450
2314 017450
017450 004537 003776
2315
2316 017454
017454 010146
017456 012746 004134
017462 012746 000002
017466 010600
017470 104414
017472 062706 000006
2317 017476 012703 002526
2318 017502 012705 015716
2319 017506 012301
2320 017510 012304
2321 017512 004737 017574
2322 017516 020302
2323 017520 103772
2324
2325
2326
2327
2328
2329
2330 017522 020227 002722
2331 017526 001036
2332 017530 005762 000002
2333 017534 001433
2334 017536 012301
2335 017540 011304
2336 017542 012705 015746
2337 017546
017546 012746 005704
017552 012746 000001
017556 010600
017560 104415

```
.SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9301 -
;*****
;* This is an error reporting subroutine which prints any BMP codes
;* that are found in the BMP code queue, together with the the number of
;* the test that was executing at the time the BMP code was logged.
;*
;* INPUTS: R1 - The address of the first message to be reported.
;* R2 - The address of the next empty cell in the queue.
;*
;* OUTPUTS: The test number followed by the BMP code are printed at the
;* operator console.
;*
;* CALLING SEQUENCE: Include the label "ER9301" as the message pointer
;* parameter in the Diag Super error report macro call.
;*
;* COMMENTS: The message is printed as Basic error information.
;*
;* SUBORDINATE ROUTINES USED: None.
;*****

BGNMSG ER9301

ER9301::
SAVE ;SAVE THE GPRS ON THE STACK.
JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.

PRINTB #EF0503,R1 ;REPORT UNEXPECTED BMP CODES FOUND.
MOV R1,-(SP)
MOV #EF0503,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C#PNTB
ADD #6,SP

2$: MOV #BMPCQB,R3 ;GET THE START ADDRESS OF THE BMP CODE QUEUE.
MOV #EM9302,R5 ;GET THE MESSAGE TO BE REPORTED.
MOV (R3)+,R1 ;GET THE NUMBER OF THE TEST THAT WAS EXECUTING.
MOV (R3)+,R4 ;GET BMP CODE THAT WAS REPORTED OFF THE QUEUE.
JSR PC,50$ ;GO REPORT THE BMP CODE.
CMP R3,R2 ;CHECK IF ALL CODES HAVE BEEN REPORTED.
BLO 2$ ;IF IT IS NOT THE LAST BMP CODE THEN LOOP.

;+
; Check if overflow has occurred.
; The conditions for overflow are: the pointer contains the address of the
; last cell in the queue, and a bmp code has already been written into that
; cell.
;-
CMP R2,#BMPCQE-4 ;CHECK IF THE POINTER IS AT THE LAST LOCATION.
BNE 60$ ;EXIT IF NOT AT THE LAST LOCATION.
TST 2(R2) ;CHECK FOR A BMP CODE IN THE LAST CELL
BEQ 60$ ;EXIT IF NO OVERFLOW HAS OCCURED, CELL EMPTY.
MOV (R3)+,R1 ;GET THE TEST NUMBER OFF THE QUEUE.
MOV (R3),R4 ;GET THE BMP CODE OFF THE QUEUE.
MOV #EM9303,R5 ;SELECT THE MESSAGE TO BE REPORTED.
PRINTX #EF9302 ;REPORT OVERFLOW CONDITION.
MOV #EF9302,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C#PNTX
```

GLOBAL ERROR REPORTING ROUTINE

- ER9301 -

```

017562 062706 000004
2338 017566 004737 017574      JSR    PC,50$      ;REPORT THE LAST BMP CODE PLACED ON THE QUEUE.
2339 017572 000414      BR     60$         ;EXIT.
2340
2341 017574      50$: PRINTX $EF9301,R5,R1,R4 ;PRINT THE MESSAGE.
      017574 010446
      017576 010146
      017600 010546
      017602 012746 005626
      017606 012746 000004
      017612 010600
      017614 104415
      017616 062706 000012
2342 017622 000207
2343 017624      60$: RTS    PC      ;RETURN.
      017624 004736      PASS   ;RESTORE THE GPR CONTENTS.
2344      JSR    PC,$(SP)+ ;RETURN TO PREG05 SUBRT.
2345 017626      ENDMSG
      017626
      017626 104423
L10016: TRAP    C$MSG
    
```

GLOBAL SUBROUTINES SECTION

```
2347 .SBTTL GLOBAL SUBROUTINES SECTION
2349 ;*****
2350 ;
2351 ;           FVTSKL3.P11
2352 ;
2353 ;*****
2355
2356
2357 ;**
2358 ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
2359 ; THAT ARE USED IN MORE THAN ONE TEST.
2360 ;--
```

GLOBAL SUBROUTINE

- ALTFLD -

```

2362 .SBTTL GLOBAL SUBROUTINE - ALTFLD -
2363 ;+ *****
2364 ;* - Alter Device Register Fields Routine -
2365 ;* This subroutine alters the specified field of the specified device
2366 ;* register for the specified lines. This routine can be used to set
2367 ;* or clear bits within selected fields of selected registers.
2368 ;* Use examples: Set RX.BAUD.RATE fields on lines 3 and 6.
2369 ;* Clear TX.DMA bits on all lines.
2370 ;*
2371 ;* INPUTS: R1 - Address of the registers to alter.
2372 ;* R2 - Bit fields set to desired states.
2373 ;* R3 - Bit map of lines for which to alter register.
2374 ;* R4 - Mask of bits to alter (1 indicates change bit).
2375 ;* CSRA - Contains the address of the device CSR.
2376 ;* IESTAT - Saved states of the interrupt enable bits.
2377 ;*
2378 ;* OUTPUTS: DEVICE REGISTERS - Specified register fields altered.
2379 ;* CSR IND.ADR.REG field - Destroyed.
2380 ;*
2381 ;* CALLING SEQUENCE: JSR PC,ALTFLD
2382 ;*
2383 ;* COMMENTS: This routine reads the specified registers for all lines
2384 ;* with numbers lower than the highest specified line.
2385 ;* This routine does not read the CSR.
2386 ;*
2387 ;* SUBROUTINES CALLED: None.
2388 ;-- *****
2389
2390 017630 017630 004537 003776 ALTFLD:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
; JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2391
2392 ;+
2393 ; Set up to loop for each line:
2394 ; Prepare the word to be ORed into the register contents.
2395 ; Set up the word to write into the IND.ADR.REG field of the CSR.
2396 ;-
2397 017634 010400 MOV R4,R0 ;CALCULATE THE NEW CONTENTS OF THE
2398 017636 005100 COM R0 ; REGISTER FIELDS WHICH ARE TO BE
2399 017640 040002 BIC R0,R2 ; ALTERED BY THIS ROUTINE.
2400 017642 013705 002330 MOV IESTAT,R5 ;SET UP TO WRITE IND.ADR.REG FIELD TO 0.
2401 ;+
2402 ; Loop once for each line, altering the specified field in the specified
2403 ; register if the line has been selected for altering.
2404 ; Exit the loop if no more lines to alter, or if we have altered the max
2405 ; allowable number of lines (as specified by NUMLNS).
2406 ;-
2407 017646 000241 CLC ;PREPARE FOR ROTATE, "TST R5" DOES THIS BELOW.
2408 017650 006003 2#: ROR R3 ;GET THE LINE SELECT BIT FOR THIS LINE.
2409 017652 103006 BCC 4# ;SKIP SETUP IF LINE IS NOT SELECTED.
2410 017654 010577 162364 MOV R5,@CSRA ;SET DUT CSR IND.ADR.REG FIELD TO THIS LINE.
2411 017660 011100 MOV (R1),R0 ;GET THE PRESENT CONTENTS OF THE REG TO ALTER.
2412 017662 040400 BIC R4,R0 ;CLEAR THE BIT FIELDS WE ARE TO ALTER.
2413 017664 050200 BIS R2,R0 ;OR IN THE NEW STATES OF THE FIELDS.
2414 017666 010011 MOV R0,(R1) ;WRITE THE NEW REGISTER CONTENTS TO THE REG.
2415 017670 005205 4#: INC R5 ;SET LINE NUMBER TO THE NEXT LINE.
2416 017672 005703 TST R3 ;CHECK FOR UNHANDLED LINES, CLEAR CARRY FLAG.
2417 017674 001365 BNE 2# ;LOOP IF SELECTED LINE(S) IS NOT HANDLED.

```


GLOBAL SUBROUTINE

- ALTFLD -

```
2418
2419 017676          604:  PASS          ;RESTORE GPRS.
      017676 004736          PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2420 017700 000207          RTS    PC    JSR    ;RETURN TO CALLING ROUTNE.
```

GLOBAL SUBROUTINE

- CALMSL -

```

2422 .SBTTL GLOBAL SUBROUTINE - CALMSL -
2423 ;** *****
2424 ;* - Calibrate Milli Second Loop count subroutine -
2425 ;* This subroutine calibrates the timing loop which is used in the MSLOOP
2426 ;* routine. This subroutine calculates a value for the MSLCNT variable
2427 ;* which is the number of software loops which takes 1 ms to execute in
2428 ;* the MSLOOP routine. This routine calibrates the count by using the
2429 ;* Line Time Clock (LTC), so if no LTC is available the default value for
2430 ;* the delay count must be used.
2431 ;*
2432 ;*
2433 ;* INPUTS: MSLCNT - Default 1 ms delay loop count value, or
2434 ;* value from previous calibration.
2435 ;* MSTICK - Number of MS per LTC clock tick.
2436 ;* TIMER1 - Timer counter changed by LTC interrupt service rtn.
2437 ;* CLKHRZ - Number of LTC clicks per second (50 or 60).
2438 ;*
2439 ;* OUTPUTS: CARRY - Set if LTC is available, and new calibration performed.
2440 ;* MSLCNT - New 1 ms delay loop count value if LTC available, or
2441 ;* unchanged if no LTC is available.
2442 ;*
2443 ;* CALLING SEQUENCE: JSR PC,CALMSL
2444 ;*
2445 ;* COMMENTS:
2446 ;*
2447 ;* SUBORDINATE ROUTINES CALLED: UNSDIV,OOPS.
2448 ;-- *****
2449
2450 CALMSL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
017702 017702 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2451 017706 005037 020122 CLR 62# ;CLEAR THE 2ND TIME FLAG.
2452 ;*
2453 ; Synchronize with the LTC.
2454 ;-
2455 017712 012705 000001 2# : MOV #1,R5 ;SET OUTER LOOP COUNTER TO 1 LOOP.
2456 ;INCREASE THE VALUE LOADED INTO THIS COUNTER IF THE <*&
2457 ;FOLLOWING LOOP FAILS ON FUTURE, FASTER PROCESSORS. <*&
2458 017716 005000 CLR R0 ;CLEAR THE WAIT FOR CLOCK INT COUNTER.
2459 017720 012737 000001 002364 MOV #1,TIMER1 ;SET UP COUNT OF 1 TO SYNCH WITH LTC.
2460 017726 005737 002364 4# : TST TIMER1 ;CHECK FOR COUNTER HAVING GONE TO ZERO.
2461 017732 001410 BEQ 6# ;JUMP OUT OF LOOP IF LTC HAS INTERRUPTED.
2462 017734 005200 INC R0 ;COUNT THIS ITERATION OF THE INNER LOOP.
2463 017736 001373 BNE 4# ;LOOP IF COUNTER HAS NOT TURNED OVER.
2464 017740 005305 DEC R5 ;DECREMENT THE INNER LOOP COUNTER.
2465 017742 003371 BGT 4# ;LOOP IF OUTER LOOP COUNT NOT UP.
2466 ;*
2467 ; If we got no LTC interrupt, indicate that there is no LTC available.
2468 ; LTC must be flekey, or not really an LTC at all.
2469 ;-
2470 017744 005037 002362 CLR CLKHRZ ;CLEAR LTC FREQUENCY WORD TO INDICATE NO LTC.
2471 017750 000241 CLC ;INDICATE FAILURE FOR RETURN.
2472 017752 000461 BR 60# ;BYPASS THE FOLLOWING CALIBRATION PROCEDURES.
2473 ;*
2474 ; We are now synchronized with the LTC.
2475 ; Set up for the calibration loop.
2476 ;-
2477 017754 012704 002364 6# : MOV #TIMER1,R4 ;WILL TEST TIMER1 IN THE LOOP BELOW.

```

GLOBAL SUBROUTINE

- CALMSL -

```

2478 017760 005001          CLR    R1          ;CLEAR THE OUTER LOOP COUNTER.
2479 017762 005002          CLR    R2          ;INDICATE TO CHECK ALL BITS OF TIMER1.
2480 017764 005003          CLR    R3          ;INDICATE TO CHECK FOR TIMER1 CLEAR.
2481 017766 012714 000001    MOV    #1,(R4)      ;LOAD TIMER1 WITH COUNT OF 1.
2482
2483 017772 013705 002376    8$:    MOV    MSLCNT,R5 ;LOAD MS LOOP COUNT.
2484 017776 011400          10$:   MOV    (R4),R0      ;GET THE TIMER1 VALUE.
2485 020000 010037 020124    MOV    R0,64$      ;SAVE WORD (LIKE IN THE REAL LOOP).
2486 020004 040200          BIC    R2,R0        ;LEAVE ALL THE BITS.
2487 020006 020003          CMP    R0,R3        ;COMPARE AGAINST ZERO.
2488 020010 000261          SEC                    ;SET CARRY IN CASE OF SUCCESS.
2489 020012 001406          BEQ    12$          ;EXIT LOOP IF TIMER1 HAS CLEARED.
2490 020014 005305          DEC    R5          ;COUNT DOWN THE INSIDE MS LOOP COUNT.
2491 020016 001367          BNE    10$          ;LOOP IF MS NOT UP.
2492 020020 005301          DEC    R1          ;DECREMENT THE MS TIME COUNT.
2493 020022 001363          BNE    8$          ;KEEP LOOPING.
2494 020024 004737 020470    JSR    PC,OOPS      ;WE OVERFLOWED, SOMETHING IS WRONG, ABORT.
2495
2496          ;+
2497          ; We have now have loop count information for one clock tick.
2498          ; We have negative of number of outer loops in R1, each is MSLCNT inner loops.
2499          ; We have the portion of the last outer loop not executed, in R5.
2500          ; Now we calculate the total number of inner loops executed.
2501          ;-
2501 020030 005401          12$:   NEG    R1          ;GET NUMBER OF OUTER LOOPS.
2502 020032 013702 002376    MOV    MSLCNT,R2    ;GET THE NUMBER OF INNER LOOPS PER OUTER LOOP.
2503 020036 010203          MOV    R2,R3        ;COPY NUMBER OF LOOPS FOR MULTIPLY.
2504 020040 160502          SUB    R5,R2        ;CALC # OF INNER LOOPS DONE IN LAST OUTER LOOP
2505 020042 010204          MOV    R2,R4        ; AND ADD TO ACCUMULATOR LSWORD.
2506 020044 005005          CLR    R5          ;CLEAR ACCUMULATOR MSWORD.
2507 020046 005301          14$:   DEC    R1          ;CHECK R1 FOR 0 CONDITION
2508 020050 100403          BMI    16$          ; SKIP MULTIPLICATION IF ZERO
2509 020052 060304          ADD    R3,R4        ;MULTIPLY NUMBER OF INNER
2510 020054 005505          ADC    R5          ; LOOPS PER OUTER LOOP BY
2511 020056 000773          BR    14$          ;NUMBER OF OUTER LOOPS PERFORMED.
2512
2513          ;+
2514          ; Divide the total number of inner loops by the number of MS per LTC tick.
2515          ;-
2515 020060 013701 002374    16$:   MOV    MSTICK,R1    ;# OF MS PER LTC TICK IS DIVISOR.
2516 020064 010403          MOV    R4,R3        ;LSWORD OF LOOP COUNT IS LSWORD OF DIVIDEND.
2517 020066 010502          MOV    R5,R2        ;MSWORD OF LOOP COUNT IS MSWORD OF DIVIDEND.
2518 020070 004737 023036    JSR    PC,UNSDIV    ;DIVIDE NUMBER OF LOOPS BY MS PER LTC TICK.
2519 020074 103402          BCS    18$          ;BYPASS OOPS IF WE'RE OK.
2520 020076 004737 020470    JSR    PC,OOPS      ;CLOCK ROUTINES ARE NOT LONG ENOUGH, OR BUG.
2521 020102 010137 002376    18$:   MOV    R1,MSLCNT    ;SET NEW VALUE FOR MS LOOP COUNT.
2522 020106 005137 020122    COM    62$          ;SET THE 2ND ITERATION FLAGS IF 1ST ITERATION.
2523 020112 001277          BNE    2$          ;BRANCH IF ONLY ONE ITERATION DONE.
2524 020114 000261          SEC                    ;SET THE SUCCESS FLAG FOR EXIT.
2525
2526 020116          60$:   PASS                    ;RESTORE GPRS.
2527 020120 000207          RTS    PC          JSR    PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2528          ; CARRY - SUCCESS FLAG. SET IF SUCCESS.
2529 020122 000000          62$:   .WORD    0          ;2ND CALIBRATION ITERATION FLAGS.
2530 020124 000000          64$:   .WORD    0          ;DUMMY WORD FOR STORAGE OF THE READ WORD.

```

GLOBAL SUBROUTINE

- CKTRAP -

```

2532 .SBTTL GLOBAL SUBROUTINE - CKTRAP -
2533 ;*****
2534 ;* Check Trap Routine -
2535 ;* This subroutine is used to check for a bus time-out trap (004 trap)
2536 ;* which is caused by an access to a non-existent memory or I/O location.
2537 ;* If the trap does not occur, this routine returns a success indication.
2538 ;*
2539 ;* INPUTS: R0 - Source address for move.
2540 ;* R1 - Destination address for move.
2541 ;* (R0) - Source for the move.
2542 ;*
2543 ;* OUTPUTS: (R1) - Written to the contents of (R0).
2544 ;* Carry flag - Set on return if no 004 trap detected.
2545 ;* TP4FLG - Nonzero if trap occurred, cleared otherwise.
2546 ;*
2547 ;* CALLING SEQUENCE: JSR PC,CKTRAP
2548 ;*
2549 ;* COMMENTS: If this subroutine causes a trap, either the address which
2550 ;* is labeled ADRPTR will be the trap PC address on the stack.
2551 ;*
2552 ;* SUBORDINATE ROUTINES CALLED: None.
2553 ;*****
2554
2555 020126 CKTRAP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020126 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2556 020132 005037 002346 CLR TP4FLG ;CLEAR THE 004 TRAP FLAGS.
2557 020136 011011 MOV (R0),(R1) ;PERFORM THE MOVE IN QUESTION.
2558 020140 005737 002346 ADRPTR:: TST TP4FLG ;CHECK FOR OCCURENCE OF TRAP.
2559 020144 000261 SEC ;INDICATE SUCCESS.
2560 020146 001401 BEQ 60$ ;EXIT WITH SUCCESS IF TRAP DID NOT OCCUR.
2561 020150 000241 CLC ;INDICATE FAILURE.
2562 020152 60$: PASS ;RESTORE GPRS.
020152 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2563 020154 000207 RTS PC

```

GLOBAL SUBROUTINE

- CLNRST -

SEQ 0068

2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608

020156
020156 004537 003776

020162 004737 021456
020166 103002

020170 004737 020716

020174
020174 004736

020176 000207

```
.SBTTL GLOBAL SUBROUTINE - CLNRST -
;*****
;* - Clean Reset of the Device Under Test -
;* This subroutine is used to reset the DUT to a known state.
;* The DUT's self-test is skipped, and the fifo is purged of any error
;* codes, etc.
;* If the reset does not successfully complete, then the carry bit is
;* passed back to the calling routine (clear).
;*
;* INPUTS: CSRA - Contains the address of the CSR
;* TXBFCA - Contains address of DUT DMA Buffer Count register.
;* ERRNBR - Error number for possible error report.
;* ERRTBL- ERRTYP,ERNBR,and ERRMSG set up correctly.
;*
;* OUTPUTS: The DUT performs its reset function into a known state.
;* CARRY - Clear indicates the test is to be aborted.
;* ERRBLK - value may be destroyed.
;* IESTAT - TX and RX interrupt flags are cleared.
;* TX and RX interrupt enable bits in the DUT's CSR are cleared.
;*
;* CALLING SEQUENCE: JSR PC,CLNRST
;*
;* COMMENTS: This subroutine can report errors with numbers ERRNBR.
;* This routine does not destroy the value of ERRNBR.
;*
;* SUBORDINATE ROUTINES CALLED: DELAY,MSLGET,PUFIFO,RESETT.
;*****
CLNRST:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
; R5,PREG05 ;CALL REGISTER SAVE SUBRT.
;+
; Reset the DUT.
; This routine reports errors with numbers from ERRNBR thru ERRNBR+2.
;-
JSR PC,RESETT ;RESET THE DUT TO A KNOWN STATE.
BCC 60$ ;EXIT ROUTINE WITH ABORT TEST INDICATOR.
;+
; Purge the FIFO of error codes, save any BMP codes found.
;-
JSR PC,PUFIFO ;PURGE THE FIFO.
60$:
PASS ;EXIT THE TEST USING RESETT OR PUFIFO STATUS.
;RESTORE GPRS, PASS THE FOLLOWING INTACT:
; PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
;CARRY BIT:IF CLEAR, THEN ABORT THE TEST.
RTS PC
```

GLOBAL SUBROUTINE

- CLR16W -

```

2610 .SBTTL GLOBAL SUBROUTINE - CLR16W -
2611 ;* *****
2612 ;* - Clear Sixteen Words Routine -
2613 ;* This subroutine clears 16 words starting with the specified word.
2614 ;*
2615 ;* INPUTS: R0 - Address of the first word to clear.
2616 ;*
2617 ;* OUTPUTS: (R0) to (R0+15) - 16 words of memory are cleared to 0.
2618 ;*
2619 ;* CALLING SEQUENCE: JSR PC,CLR16W
2620 ;*
2621 ;* COMMENTS:
2622 ;*
2623 ;* SUBORDINATE ROUTINES CALLED: None.
2624 ;*
2625 ;* *****
2626 CLR16W:: SAVE JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
2627 020200 004537 003776 R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2628 020204 012701 000020 ;SET THE LOOP COUNTER TO 16.
2629 020210 005020 2$: CLR (R0)+ ;CLEAR A WORD OF MEMORY.
2630 020212 005301 DEC R1 ;COUNT THIS LOOP.
2631 020214 001375 BNE 2$ ;LOOP IF NOT 16 WORD CLEARED.
2632 020216 004736 PASS ;RESTORE GPRS.
020220 000207 RTS PC JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
    
```

GLOBAL SUBROUTINE

- CNTERR -

```

2634 .SBTTL GLOBAL SUBROUTINE - CNTERR -
2635 ;++ *****
2636 ;* - Count Error Routine -
2637 ;* This subroutine is used to count a "data" error on the specified
2638 ;* line. It checks whether error summary reporting is active, or should
2639 ;* be made active on this line, and activates it if necessary.
2640 ;*
2641 ;* INPUTS: R5 - Line number of line under consideration.
2642 ;* ERCNTB - Label at base of error counters table.
2643 ;* ERSMRF - Error summary flags (bit set if line in summary mode).
2644 ;* NDERPT - Number of individual data errors to report on a line.
2645 ;*
2646 ;* OUTPUTS: CARRY - Set if line is in error summary mode.
2647 ;* ERCNT - Error counter incremented for specified line.
2648 ;* ERSMRF - Bit set if line should be in summary mode.
2649 ;*
2650 ;* CALLING SEQUENCE: JSR PC,CNTERR
2651 ;*
2652 ;* COMMENTS:
2653 ;*
2654 ;* SUBORDINATE ROUTINES CALLED: None.
2655 ;-- *****
2656
2657 020222 CNTERR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020222 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2658
2659 ;+
2660 ; Count the error on the counter for the specified line.
2661 ;-
2661 020226 006305 ASL R5 ;FORM WORD OFFSET FROM LINE NUMBER.
2662 020230 016501 002464 MOV ERCNTB(R5),R1 ;GET THE PRESENT ERROR COUNT FOR THIS LINE.
2663 020234 005201 INC R1 ;COUNT ERROR.
2664 020236 103402 BCS 2$ ;OVERFLOW? YES, DON'T UPDATE COUNTER IN TABLE.
2665 020240 010165 002464 MOV R1,ERCNTB(R5) ;UPDATE ERROR COUNTER TABLE ENTRY.
2666 020244 005737 002230 2$: TST NDERPT
2667 020250 001411 BEQ 60$ ;SUMMARYS DISABLED? YES, EXIT WITH CARRY 0.
2668 020252 020137 002230 CMP R1,NDERPT ;NO, CHECK FOR ENOUGH ERRORS FOR SUMMARY USE.
2669 020256 101002 BHI 4$ ;ENOUGH ERRORS TO USE SUMMARY? YES, GO HANDLE.
2670 020260 000241 CLC ;INDICATE NOT TO USE SUMMARY REPORT YET.
2671 020262 000404 BR 60$ ;EXIT WITH CARRY 0.
2672 020264 056537 002410 002462 4$: BIS BITTBL(R5),ERSMRF ;SET THE ERROR SUMMARY FLAG FOR LINE.
2673 020272 000261 SEC ;INDICATE TO USE SUMMARY REPORT.
2674 020274 60$: PASS ;RESTORE GPRS.
020274 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2675 020276 000207 RTS PC

```

GLOBAL SUBROUTINE

- DELAY -

```

2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695 020300
      020300 004537 003776
2696 020304 010401
2697 020306 012702 177777
2698 020312 005003
2699 020314 012704 020336
2700 020320 004737 020454
2701 020324 103002
2702 020326 004737 020470
2703 020332
      020332 004736
2704 020334 000207
2705
2706 020336 177777
    
```

```

.SBTTL GLOBAL SUBROUTINE          - DELAY -
;*****
;*                                - DELAY SUBROUTINE -
;*      This subroutine is used to delay a variable number of milli-seconds.
;*
;* INPUTS:      R4 - Contains the number of ms to delay.
;*              MSLCNT.
;*
;* OUTPUTS:     None.
;*
;* CALLING SEQUENCE:  .JSR    PC,DELAY
;*
;* COMMENTS:     If no hardware clock interrupts are occurring, control-Cs will
;*              not be honored for the duration of the delay.
;*
;* SUBORDINATE ROUTINES CALLED: None.
;*****
DELAY:: SAVE
      MOV     R4,R1      .JSR
      MOV     #-1,R2    ;SAVE CONTENTS OF GPRS R0 THRU R5.
      CLR     R3        ;R5,PREG05      ;CALL REGISTER SAVE SUBRT.
      MOV     #62$,R4   ;PASS NUMBER OF MS DELAY AS TIME-OUT VALUE.
      JSR     PC,MSLOOP ;TELL MSLOOP ROUTINE TO CHECK ALL BITS.
      BCC    60$       ;TELL MSLOOP RTN TO CHECK FOR ALL BITS CLEAR.
      JSR     PC,00PS   ;TELL MSLOOP TO CHECK DUMMY NON-ZERO WORD.
      PASS    60$      ;DELAY THE REQUESTED # OF MS.
      JSR     PC,00PS   ;EXIT ROUTINE IF WE TIMED-OUT.]
      RTS    PC        ;IF NO TIME-OUT, BAD PROGRAM OR HOST MACHINE.
      JSR     PC,00PS   ;RESTORE GPRS.
      RTS    PC        ;RETURN TO PREG05 SUBRT.
      JSR     PC,00PS   ;DUMMY, NON-ZERO WORD.
      .WORD  -1
    
```


GLOBAL SUBROUTINE

- MSLGET -

2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746 020340
020340 004537 003776
2747
2748
2749
2750
2751 020344 005102
2752 020346 040203
2753
2754
2755
2756 020350 005701
2757 020352 001011
2758 020354 011400
2759 020356 010037 020452
2760 020362 040200
2761 020364 020003
2762 020366 000261
2763 020370 001420

```
.SBTTL GLOBAL SUBROUTINE - MSLGET -
;*****
;* - Milli Seconds Loop which returns read word and remaining time -
;* This subroutine is a general purpose test loop subroutine. It is used
;* to verify that a certain action occurs before a time-out period. The
;* calling routine passes in which bits should be set and cleared for the
;* desired condition and the time-out value in milli-seconds.
;* This routine checks for the desired condition upon entrance into the
;* routine and then once each milli-second thereafter.
;* Upon return, the last word which was read to check for the condition
;* is returned by this subroutine.
;*
;* INPUTS: R1 - Time-out value in milli-seconds (up to 64K ms).
;* R2 - Bit map of bits to test (1 indicates to test the bit).
;* R3 - Desired states of the indicated fields in R2.
;* R4 - Address of the word to test.
;* MSLCNT - Milli second software loop count.
;*
;* OUTPUTS: R0 - The last word which was read to check for the condition.
;* R1 - Remaining number of ms in time-out time.
;* CARRY - Success flag (set if condition is met before time-out).
;*
;* CALLING SEQUENCE: JSR PC,MSLGET
;*
;* COMMENTS: This routine works with or without a hardware clock, but the
;* calibration is only guaranteed when a line clock is available
;* on the system.
;* This routine can be used as a delay routine, by specifying the
;* desired delay as the time-out and specifying a condition to
;* look for which will not be met during the delay.
;* If a time-out value of 0 is specified, this routine checks for
;* the desired condition before returning. It indicates success
;* if the condition is met, failure otherwise.
;*
;* SUBORDINATE ROUTINES CALLED: None.
;*****
MSLGET:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
; JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
;+
; Set up mask for removing unused bits in the test word, and clear unused
; bits in the desired state word to allow direct comparison.
;-
COM R2 ;GET MASK OF UNUSED BITS.
BIC R2,R3 ;MASK OUT UNUSED BITS IN DESIRED STATE WORD.
;+
; Handle the test and exit if we have a 0 time-out value.
;-
TST R1 ;TEST THE TIME-OUT VALUE FOR ZERO.
BNE 2$ ;IF NON-ZERO TIME-OUT, GO LOOP AND TEST.
MOV (R4),R0 ;GET THE WORD TO TEST BEFORE EXITING.
MOV R0,62$ ;SAVE VALUE SO WE CAN RETURN IT.
BIC R2,R0 ;MASK OUT UNTESTED BITS OF WORD.
CMP R0,R3 ;COMPARE AGAINST DESIRED STATE WORD.
SEC ;INDICATE SUCCESS IN CASE WORDS ARE EQUAL.
BEQ 6$ ;EXIT WITH SUCCESS IF WORDS ARE EQUAL.
```

GLOBAL SUBROUTINE

- MSLGET -

```

2764 020372 000241          CLC          ;INDICATE FAILURE (TIME-OUT).
2765 020374 000416          BR          6$          ;EXIT WITH FAILURE, WORDS AREN'T EQUAL.
2766                          ;+
2767                          ; Non-zero time-out value. Loop, waiting for condition or time-out.
2768                          ;-
2769 020376 013705 002376    2$:      MOV      MSLCNT,R5      ;LOAD MS LOOP COUNT.
2770 020402 011400          4$:      MOV      (R4),R0      ;GET THE WORD TO TEST.
2771 020404 010037 020452    MOV      R0,62$      ;SAVE WORD IN CASE THIS IS THE LAST.
2772 020410 040200          BIC      R2,R0      ;MASK OUT UNTESTED BITS OF WORD.
2773 020412 020003          CMP      R0,R3      ;COMPARE AGAINST DESIRED STATE WORD.
2774 020414 000261          SEC          ;SET CARRY IN CASE OF SUCCESS.
2775 020416 001405          BEQ      6$          ;EXIT WITH SUCCESS IF WORDS ARE EQUAL.
2776 020420 005305          DEC      R5      ;COUNT DOWN THE INSIDE MS LOOP COUNT.
2777 020422 001367          BNE      4$          ;LOOP IF MS NOT UP.
2778 020424 005301          DEC      R1      ;DECREMENT THE MS TIME COUNT.
2779 020426 001363          BNE      2$          ;IF TIME NOT UP, LOOP TO COUNT ANOTHER MS.
2780 020430 000241          CLC          ;CLEAR CARRY, WE TIMED-OUT.
2781                          ;+
2782                          ; Have either found condition, or timed-out (possibly from 0 time-out value).
2783                          ; Restore the last contents read from the test word. Exit routine.
2784                          ;-
2785 020432 013700 020452    6$:      MOV      62$,R0      ;PASS OUT THE LAST READ WORD.
2786 020436 010066 000002    60$:     PASS      R0,R1      ;RESTORE GPRS, EXCEPT THE FOLLOWING:
                                MOV      R0,R0SLOT(SP)      ;PUT R0 IN STACK SLOT.
                                MOV      R1,R1SLOT(SP)      ;PUT R1 IN STACK SLOT.
                                JSR      PC,@(SP)+          ;RETURN TO PREGO5 SUBRT.
2787                          ;R0 - LAST READ WORD CHECKED FOR CONDITION.
2788                          ;R1 - REMAINING TIME (0 IF TIME-OUT OCCURED).
2789 020450 000207          RTS      PC          ;CARRY - SET IF SUCCESS, CLEAR IF TIME-OUT.
2790                          ;+
2791                          ; Local storage.
2792                          ;-
2793 020452 000000          62$:     .WORD    0          ;STORAGE FOR THE LAST READ WORD.

```

GLOBAL SUBROUTINE

- MSLOOP -

```

2795 .SBTTL GLOBAL SUBROUTINE - MSLOOP -
2796 ;*****
2797 ;* - Test Loop subroutine -
2798 ;* This subroutine is a general purpose test loop subroutine. It is used
2799 ;* to verify that a certain action occurs before a time-out period. The
2800 ;* calling routine passes in which bits should be set and cleared for the
2801 ;* desired condition and the time-out value in milli-seconds.
2802 ;* This routine checks for the desired condition upon entrance into the
2803 ;* routine and then once each milli-second thereafter.
2804 ;*
2805 ;* INPUTS: R1 - Time-out value in milli-seconds (up to 64K ms).
2806 ;* R2 - Bit map of bits to test (1 indicates to test the bit).
2807 ;* R3 - Desired states of the indicated fields in R2.
2808 ;* R4 - Address of the word to test.
2809 ;* MSLCNT - Milli second software loop count.
2810 ;*
2811 ;* OUTPUTS: CARRY - Success flag (set if condition is met before time-out).
2812 ;*
2813 ;* CALLING SEQUENCE: JSR PC,MSLOOP
2814 ;*
2815 ;* COMMENTS: This routine works with or without a hardware clock, but the
2816 ;* calibration is only guaranteed when a line clock is available
2817 ;* on the system.
2818 ;* This routine can be used as a delay routine, by specifying the
2819 ;* desired delay as the time-out and specifying a condition to
2820 ;* look for which will not be met during the delay.
2821 ;* If a time-out value of 0 is specified, this routine checks for
2822 ;* the desired condition before returning. It indicates success
2823 ;* if the condition is met, failure otherwise.
2824 ;*
2825 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
2826 ;*****
2827
2828 020454 MSLOOP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
020454 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2829
2830 ;+
2831 ; Calling the MSLGET routine from the MSLOOP routine isolates the caller of
2832 ; MSLOOP from the returned test word and remaining time-out values.
2833 ;-
2834 020460 004737 020340 JSR PC,MSLGET ;CALL THE MULTI-PURPOSE MS LOOP AND SEARCH RTN.
2835
2836 020464 60$: PASS ;RESTORE GPRS,
020464 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2837 020466 000207 RTS PC ;CARRY - SET IF SUCCESS, CLEAR IF TIME-OUT.

```

GLOBAL SUBROUTINE

- OOPS -

```

2839 .SBTTL GLOBAL SUBROUTINE - OOPS -
2840 ;** *****
2841 ;* - Program abort subroutine -
2842 ;* This subroutine is used to abort the program when a fatal error is
2843 ;* detected in the program or the host system hardware. An error message
2844 ;* is printed giving some information about the nature of the abort.
2845 ;*
2846 ;* INPUTS: R1 - Error code giving reason for abort.
2847 ;*
2848 ;* OUTPUTS: An error message is printed.
2849 ;* A list of return PC values for all subroutine calls is printed.
2850 ;*
2851 ;* CALLING SEQUENCE: JSR PC,OOPS
2852 ;*
2853 ;* COMMENTS:
2854 ;*
2855 ;* SUBORDINATE ROUTINES CALLED: None.
2856 ;-- *****
2857
2858 020470 OOPS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
2859 020470 004537 003776 ;REPORT "HOST COMPUTER HARDWARE OR SOFTWARE BUG ENCOUNTERED." ERROR.
2860 020474 ERRSF 101,EM0101
2861 020474 104454 TRAP C$ERSF
2862 020476 000145 .WORD 101
2863 020500 020534 .WORD EM0101
2864 020502 000000 .WORD 0
2865 ; REPORT "PROGRAM HUNG, WAITING FOR A CONTROL-C."
2866 020504 PRINTF #EM0102
2867 020504 012746 020620 MOV #EM0102,-(SP)
2868 020510 012746 000001 MOV #1,-(SP)
2869 020514 010600 MOV SP,R0
2870 020516 104417 TRAP C$PNTF
2871 020520 062706 000004 ADD #4,SP
2872 2$: BREAK ;LOOK FOR OPERATOR CONTROL-C INPUT.
2873 TRAP C$BRK
2874 ;INFINITE LOOP.
2875 60$: BR 2$ ;DON'T NEED THIS, BUT SOMEBODY MAY CHANGE THIS
2876 ;ROUTINE IN THE FUTURE, SO BE CONSISTANT.
2877 020530 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2878 020532 000207 RTS PC
2879
2880 020534 110 117 123 EM0101:: .ASCIZ /HOST COMPUTER HARDWARE OR SOFTWARE BUG ENCOUNTERED./
2881 020537 124 040 103
2882 020542 117 115 120
2883 020545 125 124 105
2884 020550 122 040 110
2885 020553 101 122 104
2886 020556 127 101 122
2887 020561 105 040 117
2888 020564 122 040 123
2889 020567 117 106 124
2890 020572 127 101 122
2891 020575 105 040 102
2892 020600 125 107 040
2893 020603 105 116 103
2894 020606 117 125 116
2895 020611 124 105 122

```

GLOBAL SUBROUTINE

- OOPS -

	020614	105	104	056	
	020617	000			
2869	020620	045	116	045	EM0102:: .ASCIZ /%N%APROGRAM HUNG, WAITING FOR A CONTROL-C. <*****N%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	
2870	020714	116	000		

.EVEN

GLOBAL SUBROUTINE

- PUFIFO -

```

2872 .SBTTL GLOBAL SUBROUTINE - PUFIFO -
2873 ;*****
2874 ;* - PURGE THE FIFO
2875 ;* This routine tries to remove all the characters from the FIFO.
2876 ;* Any BMP codes that are found are saved on the BMP code queue.
2877 ;*
2878 ;* INPUTS: RBUFA- Contains the address of the Receiver.
2879 ;*
2880 ;*
2881 ;* OUTPUTS: Carry bit - Indicates the state of the fifo, set:= purged.
2882 ;* BMPCQ - The contents of the BMP code queue may be updated.
2883 ;*
2884 ;* CALLING SEQUENCE: JSR PC,PUFIFO
2885 ;*
2886 ;* COMMENTS:
2887 ;*
2888 ;* SUBORDINATE ROUTINES CALLED: SAVBMP.
2889 ;*****
2890
2891 020716 004537 003776 PUFIFO::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
2892 020716 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2893 020722 012701 001000 MOV #512.,R1 ;SET MAXIMUM TRY COUNT OF 512.
2894 020726 013704 002246 MOV RBUFA,R4 ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
2895 020732 011402 2$: MOV (R4),R2 ;GET THE CONTENTS OF THE RECEIVER BUFFER REG.
2896 020734 100016 BPL 6$ ;EXIT IF THE FIFO IS EMPTY, DATA_VALID CLR.
2897 ;*
2898 ; Check if the read character is actually a BMP code.
2899 ; If it is, then save it on the BMP code queue to be reported later.
2900 ;-
2901 020736 012700 070000 MOV #70000,R0 ;GENERATE A BIT MAP OF CHAR ERROR BITS
2902 020742 040200 BIC R2,R0 ; WHICH ARE NOT SET FOR CHAR.
2903 020744 001006 BNE 4$ ;THROW CHAR AWAY IF NOT BMP OR SELFTEST CODE.
2904 ;*
2905 ; Check if the read data is modem status , BMP or Selftest?.
2906 ;-
2907 020746 012700 000300 MOV #300,R0 ; CHECK IF BMP OR SELFTEST?.
2908 020752 040200 BIC R2,R0 ;TRY TO CLEAR BMP FLAGS IN THE READ DATA.
2909 020754 001002 BNE 4$ ;IF IT IS MODEM OR SELFTEST CODE THROW IT AWAY.
2910 020756 004737 022262 JSR PC,SAVBMP ;SAVE BMP CODE ON THE QUEUE.
2911 ;*
2912 020762 005301 4$: DEC R1 ;DECREMENT THE TRY COUNT.
2913 020764 001362 BNE 2$ ;LOOP TO TRY AGAIN.
2914 020766 000241 CLC ;CLEAR CARRY,TO INDICATE FIFO NOT PURGED.
2915 020770 000401 BR 60$ ;EXIT WITH CARRY CLEAR.
2916 020772 000261 6$: SEC ;SET CARRY, TO INDICATE FIFO PURGED.
2917 ;*
2918 020774 60$: PASS ;RESTORE GPRS,
2919 020774 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2920 020776 000207 RTS PC ;CARRY BIT, SET INDICATES FIFO PURGED.

```

GLOBAL SUBROUTINE

- RDPDR -

```

2922 .SBTTL GLOBAL SUBROUTINE - RDPDR -
2923 ;*****
2924 ;* - Read and verify Data Pattern from Device Registers Routine -
2925 ;* This routine reads and verifies the rotated data pattern which has
2926 ;* been written by the WDPDR subroutine.
2927 ;* Each active line's register's contents is read and compared with the
2928 ;* written data.
2929 ;* After the unused and Read Only (RO) bits are masked out, any errors are
2930 ;* reported from this routine.
2931 ;* This routine will take into account the type of write operation which
2932 ;* was performed by the WDPDR subroutine.
2933 ;*
2934 ;* INPUTS: R2 - Used to pass in the data pattern to be rotated & verified.
2935 ;* R3 - Byte indicator (- => lo byte, + => hi byte, 0 => both).
2936 ;* R4 - Operation type indicator (- => BIC, + => BIS, 0 => MOV).
2937 ;* ACTLNS - Bit map of active lines on the device under test.
2938 ;* CSRA - Contains the CSR address of the Device under test.
2939 ;* DRADRT - Base address of device register address table.
2940 ;* ERCNTB - Label at base of error counters table for lines.
2941 ;* ERRMSG - Set up with the proper error message for this test.
2942 ;* ERRNBR - Set up with the proper error number.
2943 ;* LPRO - Equated to LPR reg offset from device CSR address.
2944 ;* NUMLNS - Number of lines on the device under test.
2945 ;* NDERPT - Number of individual data errors to report on a line.
2946 ;* TXBFCO - Equated to TBUFFCT reg offset from device CSR address.
2947 ;* UNBTB - Base address of the unused bit table.
2948 ;*
2949 ;* OUTPUTS: Error messages may be printed at the operator's console.
2950 ;* ERCNT - Error counters table is updated for line under test.
2951 ;* ERRBLK - Contents destroyed.
2952 ;* ERSMRF - Error summary flags bit set if line in summary mode.
2953 ;* UUT CSR - All bits cleared, except IND.ADR.REG field destroyed.
2954 ;*
2955 ;* CALLING SEQUENCE: JSR PC,RDPDR
2956 ;*
2957 ;* COMMENTS: For byte accesses, only the specified byte is verified.
2958 ;*
2959 ;* SUBORDINATE ROUTINES CALLED: ER1601,ROLDAP.
2960 ;-- *****
2961
2962 021000 RDPDR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
2963 021000 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2964 021004 012737 016712 003774 MOV #ER1601,ERRBLK ;SET UP THE ADDRESS OF THE ERROR REPORT RTN.
2965 ;*
2966 ; Determine whether register data should be inverted from data pattern.
2967 ;--
2967 021012 005704 TST R4 ;CHECK THE OPERAND TYPE INDICATOR.
2968 021014 100001 BPL 2# ;BIC WRITE PERFORMED? NO, USE STANDARD DATA.
2969 021016 005102 COM R2 ;YES, INVERT THE DATA PATTERN.
2970 ;*
2971 ; Set up outer loop.
2972 ;--
2973 021020 005005 CLR R5 ;CLEAR LINE COUNTER TO SELECT LINE 0.
2974 ;*
2975 ; The outer loop follows. Each pass through this loop reads and compares data
2976 ; from all of the device registers for a particular line if the line is active.
2977 ;--

```

GLOBAL SUBROUTINE

- RDPDR -

```

2978 021022 010237 021216      4$:  MOV    R2,70$           ;SAVE THE OUTER LOOP DATA PATTERN.
2979 021026 010577 161212      MOV    R5,@CSRA        ;SET CSR IND.ADR.REG FIELD TO THIS LINE.
2980 021032 010500              MOV    R5,R0
2981 021034 006300              ASL    R0
2982 021036 036037 002410 002236  BIT    BITTBL(R0),ACTLNS
2983 021044 001452              BEQ    16$             ;IS THE LINE ACTIVE? NO, SKIP THE LINE.
2984 021046 012703 000004      MOV    @LPRO,R3        ;YES, INITIALIZE REGISTER OFFSET FOR LPR.
2985
2986      ;+
2987      ; The inner loop follows. Each pass through this loop reads and compares
2988      ; data from a device register.
2989 021052 010204      6$:  MOV    R2,R4           ;SAVE THE INNER LOOP DATA PATTERN.
2990 021054 046302 002264      BIC    UNBTTB(R3),R2   ;REMOVE UNUSED BITS FROM EXPECTED DATA.
2991 021060 016300 002244      MOV    DRADRT(R3),R0
2992 021064 005766 000010      TST    R3SLOT(SP)     ;CHECK THE ACCESS TYPE INDICATOR.
2993 021070 001002      BNE    8$             ;BYTE ACCESS? YES, GO PERFORM BYTE READ.
2994 021072 011001      MOV    (R0),R1        ;NO, PERFORM WORD READ OF DEVICE REGISTER.
2995 021074 000416      BR     12$
2996 021076 100410      8$:  BMI    10$           ;LOW BYTE ACCESS? YES, GO DO LOW BYTE READ.
2997 021100 005200      INC    R0             ;HIGH BYTE ACCESS. FORM HIGH BYTE ADDRESS.
2998 021102 111001      MOVB   (R0),R1        ;READ THE HI BYTE OF THE DUT REGISTER.
2999 021104 000301      SWAB   R1             ;PUT HI BYTE BACK INTO THE HI BYTE.
3000 021106 042701 000377      BIC    @377,R1        ;REMOVE THE UNUSED BYTE IN ACTUAL DATA.
3001 021112 042702 000377      BIC    @377,R2        ;REMOVE THE UNUSED BYTE IN EXPECTED DATA.
3002 021116 000405      BR     12$
3003 021120 111001      10$: MOVB   (R0),R1        ;READ THE LOW BYTE OF THE DUT REGISTER.
3004 021122 042701 177400      BIC    @177400,R1     ;REMOVE THE UNUSED BYTE.
3005 021126 042702 177400      BIC    @177400,R2     ;FORM EXPECTED LOW BYTE FOR COMAPARISON.
3006
3007 021132 046301 002264      12$: BIC    UNBTTB(R3),R1   ;REMOVE UNUSED BITS FROM ACTUAL DATA.
3008 021136 020102      CMP    R1,R2         ;COMPARE ACTUAL AND EXPECTED DATA.
3009 021140 001404      BEQ    14$           ;ACTUAL = EXPECTED? YES, SKIP ERROR.
3010 021142 004737 020222      JSR    PC,CNTERR     ;NO, COUNT THE ERROR, CHECK FOR ERROR SUMMARY.
3011 021146 103401      BCS    14$           ;USE ERROR SUMMARY? YES, SKIP ERROR.
3012
3013      ;No. report "BAD BIT(S) IN DEVICE xxxxx REGISTER FOR LINE nn (D)."  

3013 021150              ERROR
3013 021150 104460
3014 021152 010402      14$: MOV    R4,R2           ;RESTORE THE INNER LOOP DATA PATTERN.
3015 021154 004737 021570      JSR    PC,ROLDAP     ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
3016 021160 062703 000002      ADD    @2,R3         ;SET REGISTER OFFSET TO THE NEXT REGISTER.
3017 021164 020327 000016      CMP    R3,@TXBFCO    ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
3018 021170 003730      BLE    6$           ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
3019
3020      ;+
3021      ; Back into the outer loop. Now set up for next line. Loop if not done.
3021      ;-
3022 021172 013702 021216      16$: MOV    70$,R2        ;SET UP TO ROTATE THE DATA PATTERN.
3023 021176 004737 021570      JSR    PC,ROLDAP     ;ROTATE THE DATA PATTERN.
3024 021202 005205      INC    R5            ;COUNT THIS LINE
3025 021204 020527 000010      CMP    R5,@NUMLNS    ;COMPARE LINE COUNT WITH NUMBER OF LINES.
3026 021210 002704      BLT    4$           ;LOOP IF SOME LINES NOT DONE.
3027
3028 021212      60$: PASS           ;RESTORE GPRS.
3028 021212 004736      JSR    PC,@(SP)+     ;RETURN TO PREG05 SUBRT.
3029 021214 000207      RTS    PC
3030
3031 021216 000000      70$: .WORD 0         ;STORAGE FOR DATA PATTERN OUTSIDE INNER LOOP.

```


GLOBAL SUBROUTINE

- REGTST -

```

3033 .SBTTL GLOBAL SUBROUTINE - REGTST -
3034 ;* *****
3035 ;* - Registers Test Subroutine -
3036 ;* Subroutine to test the Device Under Test (DUT) registers. The used
3037 ;* bits of the registers are either all cleared or all set and then the
3038 ;* data pattern is written and verified using either word or byte
3039 ;* accesses in read/write or read/modify/write mode.
3040 ;*
3041 ;* INPUTS: R3 - Byte indicator (- => low, + => high, 0 => both bytes).
3042 ;* R4 - Access mode (-1 => set then BIC, 1 => clear then BIS,
3043 ;* (-2 => set then MOV, +2 clear then MOV).
3044 ;* ERRNBR - Set up with initial error number.
3045 ;*
3046 ;* OUTPUTS: GPRS0 - GPR save area 0 is destroyed.
3047 ;* Device Under Test registers are written.
3048 ;* Error messages may be printed at the operators console.
3049 ;*
3050 ;* CALLING SEQUENCE: JSR PC,REGTST
3051 ;*
3052 ;* COMMENTS: This routine loop 16 times writing the same data pattern
3053 ;* rotated left once each iteration.
3054 ;* This routine can report errors INITIAL ERRNBR thru INITIAL+2.
3055 ;*
3056 ;* SUBORDINATE ROUTINES CALLED: RDPDR,ROLDAP,SWAPO,WDPDR
3057 ;* -- *****
3058
3059 021220 REGTST:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
021220 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3060 ;*
3061 ; Set up the GPRs for the writting of the data pattern.
3062 ;-
3063 021224 012705 000020 MOV #16.,R5 ;SET UP LOOP COUNTER TO COUNT 16 ITERATIONS.
3064 021230 012702 167410 MOV #167410,R2 ;INITIALIZE THE DATA PATTERN.
3065 021234 032704 000001 BIT #BIT0,R4 ;TEST FOR R/W ACCESS.
3066 021240 001001 BNE 2$ ;R/M/W ACCESS? YES, R4 IS ALL SET UP.
3067 021242 005004 CLR R4 ;NO, INDICATE R/W ACCESS.
3068 021244 2$:
3069 ;+
3070 ; Set up the GPRs for the clearing or setting of all the used bits.
3071 ;-
3072 021244 010400 MOV R4,R0 ;PASS OPERATION TYPE INDICATOR AROUND SWAPO.
3073 021246 004737 022406 JSR PC,SWAPO ;GET ALTERNATE GPR SET IN R1 THRU R5.
3074 021252 013701 003770 MOV ERRNBR,R1 ;SAVE THE INITIAL ERROR NUMBER.
3075 021256 010004 MOV R0,R4
3076 021260 005404 NEG R4 ;SET UP OP TYPE FOR CLEARING OR SETTING.
3077 021262 005002 CLR R2 ;SET UP CLEAR WRITE PATTERN.
3078 021264 026627 000012 000002 CMP R4,SLOT(SP),#2 ;TEST FOR CLEAR THEN MOV TEST SEQUENCE.
3079 021272 001401 BEQ 4$ ;CLEAR THEN MOV? YES, LEAVE WRITE PAT CLEAR.
3080 021274 005102 COM R2 ;NO, SET ALL BITS OF WRITE PATTERN.
3081 021276 005003 4$: CLR R3 ;INDICATE THAT WORD ACCESSES SHOULD BE USED.
3082 021300 005000 CLR R0 ;SET ALTERNATE BYTE EXPECTED DATA PAT TO CLEAR.
3083 021302 026627 000012 177776 CMP R4,SLOT(SP),#-2 ;TEST FOR SET THEN MOV TEST SEQUENCE.
3084 021310 001001 BNE 6$ ;SET THEN MOV? YES, LEAVE ALT BYTE PAT CLEAR.
3085 021312 005100 COM R0 ;NO, SET ALT BYTE EXPECTED DATA PAT TO ALL 1'S.
3086 021314 004737 022406 6$: JSR PC,SWAPO ;RESTORE SWAPPED GPR VALUES TO R1 THRU R5.
3087 ;+
3088 ; Start of data pattern loop.

```

GLOBAL SUBROUTINE

- REGTST -

```

3089
3090 021320      8$:
3091
3092           ;+
3093           ; Set or clear all the used bits of the device registers for all lines.
3094           ; Verify that all the bits were set or cleared correctly.
3095 021320 004737 022406      ;-
3096 021324 004737 023322      JSR    PC,SWAPO      ;GET ALTERNATE GPRS FOR SETTING INTIAL STATES.
3097 021330 010137 003770      JSR    PC,WDPDR      ;GO CLEAR ALL USED REGISTER BITS, ALL LINES.
3098 021334 004737 021000      MOV    R1,ERRNBR     ;SET UP ERROR NUMBER TO INITIAL ERRNBR.
3099 021340 004737 022406      JSR    PC,RDPDR      ;VERIFY ALL USED REGISTER BITS, ALL LINES.
3100           JSR    PC,SWAPO      ;RESTORE MAIN GPRS CONTENTS.
3101           ;+
3102           ; Write data patterns, all lower byte used bits, all registers, all lines.
3103           ; Verify that the data pattern was written correctly.
3104 021344 004737 023322      ;-
3105 021350 005237 003770      JSR    PC,WDPDR      ;WRITE DATA PATTERN TO DEVICE REGISTERS.
3106 021354 004737 021000      INC    ERRNBR        ;SET ERROR NUMBER TO INITIAL+1.
3107 021360 005703              JSR    PC,RDPDR      ;VERIFY DATA PATTERN IN ALTERED BYTE(S).
3108 021362 001411              TST    R3             ;CHECK THE BYTE INDICATOR.
3109           BEQ    10$      ;WORD ACCESS? YES, SKIP SECOND BYTE CHECK.
3110           ;+
3111           ; Check that the alternate (unmodified) byte is clear or set as expected.
3112 021364 010201              ;-
3113 021366 010002              MOV    R2,R1         ;SAVE THE DATA PATTERN.
3114 021370 005403              MOV    R0,R2         ;GET THE ALTERNATE BYTE EXPECTED DATA.
3115 021372 005237 003770      NEG    R3             ;INDICATE THAT OTHER BYTE IS TO BE CHECKED.
3116 021376 004737 021000      INC    ERRNBR        ;SET ERROR NUMBER TO INITIAL+2.
3117 021402 005403              JSR    PC,RDPDR      ;VERIFY DATA PATS IN OTHER BYTES OF REGISTERS.
3118 021404 010102              NEG    R3             ;RESTORE BYTE INDICATOR.
3119           MOV    R1,R2         ;RESTORE DATA PATTERN.
3120           ;+
3121           ; Pepare the next data pattern and loop if not done.
3122 021406 004737 021570      10$: JSR    PC,ROLDAP     ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
3123 021412 005305              DEC    R5             ;COUNT THIS ITERATION OF THE LOOP.
3124 021414 003341              BGT    8$            ;ALL PATTERNS DONE? NO, LOOP.
3125           ;YES, RESTORE ERROR NUMBER AND EXIT.
3126 021416 013737 002450 003770 60$: MOV    GPRS0B,ERRNBR ;GET THE ERROR NUMBR FROM GPR SWAP STORAGE.
3127 021424              PASS                    ;RESTORE GPRS.
3128 021426 000207              RTS    PC            JSR    PC,@(SP)+      ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- REPSMR -

3130
 3131
 3132
 3133
 3134
 3135
 3136
 3137
 3138
 3139
 3140
 3141
 3142
 3143
 3144
 3145
 3146
 3147
 3148
 3149
 3150
 3151
 3152
 3153
 3154
 3155
 3156 021430
 021430 004537 003776
 3157 021434 005737 002462
 3158 021440 001404
 3159
 3160
 3161
 3162 021442 012737 017160 003774
 3163
 3164
 3165
 3166
 3167 021450
 021450 104460
 3168
 3169 021452
 021452 004736
 3170 021454 000207

```
.SBTTL GLOBAL SUBROUTINE - REPSMR -
;+ *****
;* - Report Error Summary Routine -
;* This subroutine reports an error summary for those lines which have
;* exceeded the number of individual errors to report for a single line
;* in a single test. This parameter can be specified by the operator if
;* he/she answers the Software Parameter Questions.
;*
;* INPUTS:   ERCNTB - Label at base of line error counters table.
;*           ERRMSG - Address of primary error message.
;*           ERRNBR - Error number of errors in this routine.
;*           ERSMRF - "Report error summary for line" flags.
;*
;* OUTPUTS:  ERRBLK - Address of error reporting routine (Destroyed).
;*           Summary messages may be printed at the operator console.
;*
;* CALLING SEQUENCE:  JSR   PC,REPSMR
;*
;* COMMENTS:  If no lines have exceeded the maximum number of individual
;* errors to report, no messages are printed by this routine.
;* Error summaries in this routine are reported as errors.
;* The contents of ERRBLK are destroyed.
;*
;* SUBORDINATE ROUTINES CALLED:
;-- *****
REPSMR:: SAVE
;SAVE CONTENTS OF GPRS R0 THRU R5.
;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
TST ERSMRF JSR ;CHECK THE "PRINT LINE ERROR SUMMARY" FLAGS.
BEQ 60$ ;EXIT WITHOUT ACTION IF NO SUMMARY FLAGS SET.
;+
; We have some error summaries to report.
;--
MOV #ER9004,ERRBLK ;SELECT ERROR REPORTING ROUTINE.
;+
; Report
; "ERROR SUMMARY REPORT FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:"
;--
ERROR
TRAP C$ERROR
60$: PASS ;RESTORE GPRS.
;PC,8(SP)+ ;RETURN TO PREG05 SUBRT.
RTS PC JSR
```

GLOBAL SUBROUTINE

- RESETT -

```

3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197 021456
      021456 004537 003776
3198 021462 012702 000040
3199
3200
3201
3202
3203
3204 021466 013704 002244
3205 021472 030214
3206 021474 001406
3207 021476 005003
3208 021500 012701 004704
3209 021504 004737 020340
3210 021510 103012
3211
3212
3213
3214
3215
3216
3217 021512 010277 160526
3218 021516 004737 022330
3219
3220
3221
3222
3223
3224 021522 005003
3225 021524 012701 004704
3226 021530 004737 020340
3227 021534 103410
    
```

```

.SBTTL GLOBAL SUBROUTINE          - RESETT -
;*****
;*          - Reset Device Under Test -
;* This subroutine is used to reset the DUT to a known state.
;* If reset does not successfully complete, ie. time-out occurs, then
;* an abort test error message is reported.
;*
;* INPUTS:   CSRA - Contains the address of the CSR
;*           TXBFCA - Contains address of DUT DMA Buffer Count register.
;*           ERRTBL- ERRTYP,ERNBR,and ERRMSG set up correctly.
;*
;* OUTPUTS:  The DUT performs its reset function into a known state.
;*           CARRY - Clear indicates the test is to be aborted.
;*           ERRBLK - value may be destroyed.
;*           IESTAT - TX and RX interrupt flags are cleared.
;*           TX and RX interrupt enable bits in the DUT's CSR are cleared.
;*
;* CALLING SEQUENCE:  JSR   PC,RESETT
;*
;* COMMENTS:  This subroutine can report errors with numbers initial ERRNBR
;*            This routine does not destroy the value of ERRNBR.
;*
;* SUBORDINATE ROUTINES CALLED: DELAY,MSLGET.
;*****
RESETT:: SAVE                ;SAVE CONTENTS OF GPRS R0 THRU R5.
                JSR          R5,PREG05 ;CALL REGISTER SAVE SUBRT.
                MOV         #BIT05,R2 ;SET BIT MASK OF MASTER RESET BIT.
;+
; Test the state of the master reset bit in the CSR.
; If MR is set then wait for self-test to complete.
; If time-out occurs, report the error and pass-out abort test indicator.
;-
                MOV         CSRA,R4 ;GET THE ADDRESS OF THE DUT'S CSR.
                BIT         R2,(R4) ;CHECK STATE OF MASTER RESET BIT.
                BEQ         2$ ;DON'T DELAY IF MR IS ALREADY CLEAR.
                CLR         R3 ;SET UP DESIRED STATE OF MASTER RESET BIT.
                MOV         #2500.,R1 ;PASS TIME-OUT VALUE OF 2.5 SECONDS.
                JSR         PC,MSLGET ;WAIT FOR SELF-TEST TO COMPLETE, MR CLEAR.
                BCC         4$ ;GO REPORT ERROR IF TIMEOUT OCCURRED.
;+
; Set Master Reset bit in CSR. Clear TX and RX enable bits, etc.
; Skip the selftest.
; Time-out of 2.5 secs, just in case the self-test executes.
;-
2$:             MOV         R2,@CSRA ;SET MASTER RESET BIT, DISABLE TX AND RX INTS.
                JSR         PC,SKPSTS ;TRY TO SKIP THE SELFTEST.
;+
; Set Self-test time-out of 2.5 seconds, and wait for M.R to clear.
; If Time-out occurs, then report the fatal error and pass-out the abort
; test indicator.
;-
                CLR         R3 ;SET UP DESIRED STATE OF MASTER RESET BIT.
                MOV         #2500.,R1 ;PASS TIME-OUT VALUE OF 2.5 SECONDS.
                JSR         PC,MSLGET ;WAIT FOR SELF-TEST TO COMPLETE, MR CLEAR.
                BCS         6$ ;SKIP ERROR REPORT IF MR CLEARED IN TIME.
    
```

GLOBAL SUBROUTINE

- RESETT -

```

3228
3229      ;+
3230      ; Set up error message to report "fatal error found during reset,test aborted".
3231      ; Indicate test is to be aborted by clearing the carry bit.
3232      ;-
3232 021536 012701 012362      4$:   MOV    #EM1601,R1      ;PASS ERROR MESSAGE TO REPORT.
3233 021542 012737 017010 003774  MOV    #ER1603,ERRBLK ;PASS ADDRESS OF ERROR HANDLING ROUTINE.
3234      ;REPORT ERROR "TIME-OUT OCCURRED WAITING FOR MASTER RESET TO CLEAR"
3235      ; "TEST ABORTED"
3236 021550      ERROR      ;          >>>>> ERROR <<<<<<
3236 021550 104460      TRAP    C$ERROR
3237 021552 000241      CLC          ;INDICATE TEST IS TO BE ABORTED.
3238 021554 000403      BR        60$      ;EXIT THIS SUBROUTINE, ABORT TEST INDICATOR.
3239      ;+
3240      ; Clear TX and RX Interrupt enable status flags in IESTAT.
3241      ; Exit with continue test indicator set (ie,carry set).
3242      ;-
3243 021556 005037 002330      6$:   CLR    IESTAT      ;CLEAR TX AND RX INTERRUPT STATUS FLAGS.
3244 021562 000261      SEC          ;INDICATE SUCCESS, CONTINUE TEST.
3245
3246 021564      60$:   PASS          ;RESTORE GPRS, PASS THE FOLLOWING INTACT:
3246 021564 004736      JSR          PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
3247      ;CARRY BIT:IF CLEAR,INDICATES ABORT TEST.
3248 021566 000207      RTS    PC
3249

```

GLOBAL SUBROUTINE

- ROLDAP -

```

3251 .SBTTL GLOBAL SUBROUTINE - ROLDAP -
3252 ;*****
3253 ;* - ROTate Left DATA Pattern
3254 ;* This routine rotates the passed input data pattern left,without going
3255 ;* through the carry.The carry is initially set or cleared depending
3256 ;* upon the state of the MSB of the data pattern,before a ROL instruction
3257 ;* is executed.
3258 ;*
3259 ;* INPUTS: R2 - Contains the data pattern to be rotated
3260 ;*
3261 ;* OUTPUTS: R2 - Contains the rotated data pattern
3262 ;*
3263 ;* CALLING SEQUENCE: JSR PC,ROLDAP
3264 ;*
3265 ;* COMMENTS:
3266 ;*
3267 ;* SUBORDINATE ROUTINES CALLED: NONE
3268 ;*****
3269
3270 021570 ROLDAP::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
021570 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3271 021574 010202 MOV R2,R2 ;SET PROCESSOR STATUS CODES
3272 021576 005702 TST R2 ;CHECK MSB
3273 021600 100402 BMI 2$ ;BRANCH IF SET
3274 021602 000241 CLC ;CLEAR CARRY BIT IF MSB CLEAR
3275 021604 000401 BR 4$ ;
3276 021606 000261 2$: SEC ;SET CARRY IF MSB SET
3277 021610 006102 4$: ROL R2 ;ROTATE DATA PATTERN LEFT
3278 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.
3279 ;R2 - CONTAINS THE ROTATED DATA PATTERN
3280 021620 000207 RTS PC
    
```

GLOBAL SUBROUTINE

- RSTRPT -

```

3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311 021622
      021622 004537 003776
3312
3313
3314
3315
3316 021626 005003
3317 021630 013705 003770
3318 021634 017702 160406
3319 021640 100412
3320
3321
3322
3323 021642 010537 003770
3324 021646 012701 015522
3325 021652 012737 017262 003774
3326
3327
3328
3329
3330 021660
      021660 104460
3331
3332
3333
3334 021662 000261
3335 021664 000545
3336
    
```

```

.SBTTL GLOBAL SUBROUTINE - RSTRPT -
;+ *****
;* - Report any Reset Errors Routine -
;* This routine determines if any error codes are among the diagnostic
;* codes reported placed in the DUT received character FIFO by the
;* Self-test. If any non BMP error codes are found, or if other errors
;* are encountered, appropriate errors are reported. Any BMP codes that
;* are found, are placed on the BMP code queue to be reported later.
;* This routine also purges the DUT FIFO looking for any characters
;* or modem status codes. If any are found, errors are reported.
;*
;* INPUTS: ERRMSG - Address of the primary error message.
;*          ERRNBR - Error number of first error reported by this routine.
;*          NUMLNS - Equated to the number of line on the DUT.
;*          RBUFA - Contains address of the DUT receiver FIFO.
;*
;* OUTPUTS: CARRY - Success flag (set if FIFO cleared successfully).
;*          ERRBLK - Address of the error report routine (Destroyed).
;*          Error messages can be printed at the operators console.
;*
;* CALLING SEQUENCE: JSR PC,RSTRPT
;*
;* COMMENTS: This subroutine can report errors with numbers Initial ERRNBR
;*            thru Initial ERRNBR+4.
;*            This routine does not destroy the value of ERRNBR.
;*
;* SUBORDINATE ROUTINES CALLED: ER0503,ER9007,ER9008,SAVBMP.
;-- *****
RSTRPT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
            JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
;+
; Read correct number (number of line on DUT) of chars from the FIFO.
; Verify that each char is a selftest success code.
;-
        CLR R3 ;CLEAR THE CODE COUNTER.
        MOV ERRNBR,R5 ;SAVE ERRNBR FOR RESTORATION LATER.
2$:     MOV @RBUFA,R2 ;READ A CHAR FROM THE DUT FIFO.
        BMI 4$ ;SKIP ERROR IF DATA.VALID SET FOR CHAR.
;+
; We expect a selftest code, but this FIFO slot is empty.
;-
        MOV R5,ERRNBR ;RESTORE ERROR NUMBER TO INITIAL VALUE.
        MOV @EM9018,R1 ;PASS ERROR MESSAGE INFO TO ER9007 ROUTINE.
        MOV @ER9007,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
;+
; Report error with number Initial ERRNRB.
; "NO SELFTEST CODE IN SELFTEST CODE FIFO SLOT FOR LINE nn AFTER RESET."
;-
        ERROR ; >>>> ERROR <<<<<.
                                     TRAP C$ERROR
;+
; Indicate "success" (because FIFO is purged), and exit this routine.
;-
        SEC ;SET SUCCESS FLAG.
        BR 60$ ;EXIT ROUTINE.
;+
    
```

GLOBAL SUBROUTINE

- RSTRPT -

```

3337      ; Determine if this is not a selftest code.
3338      ;--
3339 021666 012700 070001      4$:  MOV    #70001,R0      ;GENERATE BIT MAP OF ANY CLEAR ERROR BITS OR
3340 021672 040200              BIC    R2,R0          ; BIT 0 WHICH ARE CLEAR.
3341 021674 001033              BNE    8$            ;GO TO REPORT ERROR IF THIS IS NOT A TEST CODE.
3342      ;+
3343      ; We have a test code (either BMP or selftest code).
3344      ; Determine what type of code we have.
3345      ;--
3346 021676 032702 000200      BIT    #8IT7,R2      ;TEST ROM VERSION CODE INDICATOR BIT.
3347 021702 001443              BEQ    10$            ;SKIP ERRORS IF SELFTEST ROM VERSION CODE.
3348 021704 120227 000203      CMPB   R2,#203       ;CHECK IF SKIP SELF TEST CODE.
3349 021710 001440              BEQ    10$            ;SKIP ERROR REPORT IF SKIP SELF TEST CODE FOUND
3350 021712 120227 000201      CMPB   R2,#201       ;CHECK IF NULL CODE PRESENT.
3351 021716 001435              BEQ    10$            ;SKIP ERROR REPORT IF SELF TEST NULL CODE.
3352 021720 012700 000300      MOV    #300,R0      ;TEST CODE TYPE BITS FOR BOTH CODE
3353 021724 040200              BIC    R2,R0          ; TYPE BITS SET (INDICATING BMP CODE).
3354 021726 001003              BNE    6$            ;IF IT IS NOT A BMP CODE GO REPORT ERROR.
3355 021730 004737 022262      JSR    PC,SAVBMP     ;SAVE THE BMP CODE ON THE QUEUE.
3356 021734 000426              BR     10$           ;GO GET THE NEXT CHARACTER FROM THE FIFO.
3357      ;+
3358      ; We have a selftest error code.
3359      ;--
3360 021736 010537 003770      6$:  MOV    R5,ERRNBR     ;RESTORE ERROR NUMBER TO INITIAL VALUE.
3361 021742 005237 003770      INC    ERRNBR        ;CALCULATE INITIAL ERROR NUMBER PLUS 1.
3362 021746 012701 015547      MOV    #EM9020,R1    ;PASS ERROR MESSAGE INFO TO ER9008 ROUTINE.
3363 021752 012737 017340 003774  MOV    #ER9008,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3364      ;+
3365      ; Report error with number Initial ERRNRB + 1.
3366      ; "UNEXPECTED SELFTEST ERROR CODE FOR LINE nn IN FIFO AFTER RESET:"
3367      ;--
3368 021760              ERROR          ;          >>>>> ERROR <<<<<.
3369 021762 000413              BR     10$           ;GO TO END OF LOOP.          TRAP    C$ERROR
3370      ;+
3371      ; We have a non-selftest code (either BMP code or data char).
3372      ;--
3373 021764 010537 003770      8$:  MOV    R5,ERRNBR     ;RESTORE ERROR NUMBER TO INITIAL VALUE.
3374 021770 062737 000002 003770  ADD    #2,ERRNBR     ;CALCULATE INITIAL ERROR NUMBER PLUS 2.
3375 021776 012701 015532      MOV    #EM9019,R1    ;PASS ERROR MESSAGE INFO TO ER9007 ROUTINE.
3376 022002 012737 017262 003774  MOV    #ER9007,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3377      ;+
3378      ; Report error with number Initial ERRNRB + 2.
3379      ; "NON-SELFTEST CODE IN SELFTEST CODE FIFO SLOT FOR LINE nn AFTER RESET."
3380      ;--
3381 022010              ERROR          ;          >>>>> ERROR <<<<<.
3382 022010 104460              BR     10$           ;GO TO END OF LOOP.          TRAP    C$ERROR
3383      ;+
3384      ; End of loop, loop if not all chars have been read from the FIFO.
3385 022012 005203      10$: INC    R3          ;SET CODE COUNTER FOR NEXT ITERATION OF LOOP.
3386 022014 020327 000010      CMP    R3,#NUMLNS   ;TEST FOR ALL CODES READ.
3387 022020 002705      BLT    2$          ;LOOP IF NOT CHARS READ FROM FIFO.
3388      ;+
3389      ; Purge the FIFO until DATA.VALID is clear or until too many chars are read.
3390      ;--
3391 022022 012704 000022      MOV    #18.,R4      ;INITIALIZE THE CHARACTER COUNTER.

```


GLOBAL SUBROUTINE

- RSTRPT -

```

3392 022026 010537 003770      MOV    R5,ERRNBR      ;GET INITIAL VALUE OF THE ERROR NUMBER.
3393 022032 062737 000003 003770  ADD    #3,ERRNBR     ;CALCULATE ERROR NUMBER OF NEXT ERROR.
3394 022040 012737 017340 003774  MOV    #ER9008,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3395 022046 017702 160174      12$:  MOV    @RBUFA,R2     ;READ A CHARACTER FROM THE DUT FIFO.
3396 022052 000261      SEC    ;INDICATE SUCCESS IN CASE DATA.VALID IS CLEAR.
3397 022054 100051      BPL    60$           ;EXIT ROUTINE WITH SUCCESS IF DATA.VALID CLEAR.
3398
3399      ;+
3400      ; We have a character.
3401      ; Determine if character is a data character.
3402 022056 012700 070000      MOV    #70000,R0     ;TEST BITS 12 THRU 14 OF THE
3403 022062 040200      BIC    R2,R0         ; CODE READ FROM THE DUT FIFO.
3404 022064 001403      BEQ    14$           ;SKIP THIS ERROR IF CODE IS NOT A DATA CHAR.
3405
3406      ;+
3407      ; We have an unexpected data character: set up and go to report error.
3408 022066 012701 015573      MOV    #EM9022,R1    ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
3409 022072 000423      BR     22$           ;GO TO REPORT THIS ERROR.
3410
3411      ;+
3412      ; We have an unexpected code.
3413      ; Determine if the code is a modem status code.
3414 022074 032702 000001      14$:  BIT    #BIT0,R2     ;TEST MODEM STATUS INDICATOR BIT OF CODE.
3415 022100 001003      BNE    16$           ;SKIP THIS ERROR IF NOT MODEM STATUS CODE.
3416
3417      ;+
3418      ; We have a modem status code: set up and go to report error.
3419 022102 012701 015612      MOV    #EM9023,R1    ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
3420 022106 000415      BR     22$           ;GO TO REPORT THIS ERROR.
3421
3422      ;+
3423      ; We have an onboard test code.
3424      ; Determine if this code is a BMP code.
3425 022110 032702 000200      16$:  BIT    #BIT7,R2     ;TEST THE ROM VERSION BIT OF THE CODE.
3426 022114 001404      BEQ    18$           ;GOTO SET UP FOR SELFTEST CODE IF ROM VERSION.
3427 022116 012700 000300      MOV    #300,R0      ;TEST THE ERROR TYPE BITS OF THE CODE.
3428 022122 040200      BIC    R2,R0         ;SKIP THIS ERROR IF BMP CODE.
3429 022124 001403      BEQ    20$
3430
3431      ;+
3432      ; We have a selftest code: set up and go to report error.
3433 022126 012701 015634      18$:  MOV    #EM9024,R1    ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
3434 022132 000403      BR     22$           ;GO TO REPORT THIS ERROR.
3435
3436      ;+
3437      ; We have a BMP code: save it on the queue.
3438 022134 004737 022262      20$:  JSR    PC,SAVBMP    ;SAVE THE BMP CODE ON THE QUEUE.
3439 022140 000401      BR     24$
3440
3441      ;+
3442      ; Report the error with error number of Initial ERRNBR + 3.
3443      ; "UNEXPECTED xxx xxxx FOR LINE nn IN FIFO AFTER RESET:"
3444 022142 104460      22$:  ERROR    ; >>>> ERROR <<<<.
3445      ;                                     TRAP    C$ERROR
3446      ;+
3447      ; End of loop.
      ; Count the character we just received, and check for too many received.

```

GLOEAL SUBROUTINE

- RSTRPT -

```

3448
3449 022144 005304      24$:  ;:-      DEC   R4           ;COUNT THIS CHARACTER.
3450 022146 001337      BNE   12$           ;LOOP IF NOT TOO MANY CHARACTERS PURGED.
3451
3452      ;+
3453      ; We read too many valid characters while trying to purge the FIFO.
3454      ; Report error and exit without success.
3455      ; "FIFO WILL NOT PURGE (DATA.VALID STUCK SET), REMAINDER OF TEST SKIPPED."
3456 022150 012701 015411      MOV   #EM9017,R1      ;SELECT PROPER ERROR MESSAGE.
3457 022154 010537 003770      MOV   R5,ERRNBR      ;GET INITIAL ERROR NUMBER.
3458 022160 062737 000004 003770      ADD   #4,ERRNBR      ;CALCULATE INITIAL ERRNBR + 4.
3459 022166 012737 016470 003774      MOV   #ER0503,ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.
3460      ;PRINT ERROR REPORT.
3461 022174      ERROR      ;
3462 022174 104460      CLC           ;CLEAR THE SUCCESS FLAG.          TRAP   C$ERROR
3463
3464 022200      60$:  PASS           ;RESTORE GPRS,
3465 022202 004736      RTS    PC      JSR   PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
          000207      ; CARRY - SUCCESS FLAG (SET IF FIFO IS PURGED).

```

GLOBAL SUBROUTINE

- RXIE0 -

```

3467 .SBTTL GLOBAL SUBROUTINE - RXIE0 -
3468 ;** *****
3469 ;* - RECEIVER INTERRUPT DISABLE -
3470 ;* This routine is used to disable receiver interrupts in the DHV11.
3471 ;*
3472 ;* INPUTS: NONE.
3473 ;*
3474 ;* OUTPUTS: The RX.INT.ENBL bit is cleared in the DUT CSR.
3475 ;* IESTST -contains the updated status of the TX and RX interrupt
3476 ;* enable bits.
3477 ;*
3478 ;* CALLING SEQUENCE: JSR PC,RXIE0
3479 ;*
3480 ;* COMMENTS: The contents of the indirect address register field in
3481 ;* the DUT CSR are destroyed.
3482 ;*
3483 ;* SUBORDINATE ROUTINES CALLED: NONE.
3484 ;-- *****
3485 022204 010046 RXIE0:: MOV RO,-(SP) ;SAVE CONTENTS OF RO ON THE STACK.
3486 022206 106746 MFPS -(SP) ;SAVE PROCESSOR PRIORITY ON STACK.
3487 022210 106427 000340 MTPS @PRI07 ;IGNORE ANY INTERRUPT THAT MAY BE GENERATED.
3488 022214 042737 137777 002330 BIC @137777,IESTAT ;CLEAR RX.INT.ENBL BIT IN IESTAT.
3489 022222 013777 002330 160014 MOV IESTAT,@CSRA ;DISABLE RX INTERRUPTS.
3490 022230 106426 MTPS (SP)+ ;ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
3491 022232 012600 MOV (SP)+,RO ;RESTORE RO.
3492 022234 000207 RTS PC
    
```

GLOBAL SUBROUTINE

- RXIE1 -

```

3494 .SBTTL GLOBAL SUBROUTINE - RXIE1 -
3495 ;** *****
3496 ;* - RECEIVER INTERRUPT ENABLE -
3497 ;* This routine is used to enable receiver interrupts in the DHV11.
3498 ;*
3499 ;* INPUTS: NONE.
3500 ;*
3501 ;* OUTPUTS: The RX.INT.ENBL bit is set in the DUT CSR.
3502 ;* IESTST -contains the updated status of the TX and RX interrupt
3503 ;* enable bits.
3504 ;*
3505 ;* CALLING SEQUENCE: JSR PC,RXIE1
3506 ;*
3507 ;* COMMENTS: The contents of the indirect address register field in
3508 ;* the DUT CSR are destroyed.
3509 ;*
3510 ;* SUBORDINATE ROUTINES CALLED: NONE.
3511 ;-- *****
3512
3513 022236 052737 000100 002330 RXIE1:: BIS #BIT06,IESTAT ;SET RX.INT.ENBL BIT IN IESTAT.
3514 022244 042737 137677 002330 BIC #137677,IESTAT ;CLEAR ALL OTHER BITS, EXCEPT TX AND RX I.E.
3515 022252 013777 002330 157764 MOV IESTAT,@CSRA ;ENABLE RX INTERRUPTS.
3516 022260 000207 RTS PC
    
```

GLOBAL SUBROUTINE

- SAVBMP -

SEQ 0092

```

3518 .SBTTL GLOBAL SUBROUTINE - SAVBMP -
3519 ;** *****
3520 ;* - Save BMP codes Routine -
3521 ;* This routine saves the parameter passed in, onto the BMP code queue
3522 ;* together with the number of the currently executing test.
3523 ;*
3524 ;* INPUTS: R2 - Contains the BMP code that is to be placed on the queue.
3525 ;* BMPCQP - Contains address of next location in the bmp queue.
3526 ;* BMPCQB - Label at base of the BMP code queue.
3527 ;* BMPCQE - Label of next location after the end of the BMP queue.
3528 ;* TSTNUM - Contains the number of the current test.
3529 ;*
3530 ;* OUTPUTS: BMPCQP - Incremented by 4.
3531 ;* The contents of the BMP code queue are updated.
3532 ;*
3533 ;* CALLING SEQUENCE: JSR PC,SAVBMP
3534 ;*
3535 ;* COMMENTS: If the overflow occurs then the last location will be
3536 ;* overwritten by any subsequent attempts to update the queue.
3537 ;*
3538 ;* SUBORDINATE ROUTINES CALLED: None.
3539 ;-- *****
3540
3541 022262 SAVBMP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
022262 004537 003776 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3542 022266 013704 002524 MOV BMPCQP,R4 ;GET THE POINTER TO THE NEXT LOCATION IN QUEUE.
3543 022272 113724 002326 MOVB TSTNUM,(R4)+ ;SAVE THE CURRENT TEST NUMBER ON THE QUEUE.
3544 022276 005204 INC R4 ;INCREMENT THE POINTER TO GIVE AN EVEN ADDRESS.
3545 022300 042702 177400 BIC #177400,R2 ;CLEAR THE UNWANTED BITS FROM THE BMP CODE.
3546 022304 010224 MOV R2,(R4)+ ;SAVE THE BMP CODE ON THE QUEUE.
3547 022306 020427 002726 CMP R4,#BMPCQE ;CHECK IF OVERFLOW WILL OCCUR THE NEXT TIME.
3548 022312 103402 BLO 2$ ;GO SAVE THE POINTER IF WE WILL NOT OVERFLOW.
3549 022314 162704 000004 SUB #4,R4 ;RESET THE POINTER TO THE LAST LOCATION IN QUE.
3550 022320 010437 002524 2$: MOV R4,BMPCQP ;SAVE THE POINTER.
3551
3552 022324 60$: PASS ;RESTORE GPRS.
022324 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3553 022326 000207 RTS PC

```

GLOBAL SUBROUTINE

- SKPSTS -

```

3555 .SBTTL GLOBAL SUBROUTINE - SKPSTS -
3556 ;* *****
3557 ;* - Skip Selftest Routine -
3558 ;* This subroutine is used to skip the selftest after a DUT reset has been
3559 ;* initiated. It must be entered immediately after setting the DUT Master
3560 ;* Reset routine or after the execution of a bus reset (because of timing
3561 ;* considerations).
3562 ;*
3563 ;* INPUTS: CSRA - Contains address of the DUT CSR.
3564 ;* TXBFCA - Contains address of DUT DMA Buffer Count register.
3565 ;*
3566 ;* OUTPUTS: Skip selftest codes are written to the DUT registers.
3567 ;*
3568 ;* CALLING SEQUENCE: JSR PC,SKPSTS
3569 ;*
3570 ;* COMMENTS:
3571 ;*
3572 ;* SUBORDINATE ROUTINES CALLED: DELAY.
3573 ;*
3574 ;* - *****
3575 SKPSTS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
022330 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3576 022334 012704 000012 MOV #10.,R4 ;PASS DELAY VALUE OF 10 MILLI-SECONDS.
3577 022340 004737 020300 JSR PC,DELAY ;DELAY FOR 10 MILLI-SECONDS.
3578 ;*
3579 ; Write skip self-test code (52525) to all the indexed DUT Registers.
3580 ;*
3581 022344 012701 000050 MOV #NUMLNS!BIT05,R1 ;FORM IND.ADR.REG FIELD (PLUS M.R. BIT) WORD.
3582 ;THE ABOVE INCLUSION OF THE M.R. BIT IS NECESSARY BECAUSE OF THE
3583 ; LACK OF A M.R. BIT WRITE LOCK-OUT ON THE DHV11-M.
3584 022350 012703 052525 MOV #52525,R3 ;INITIALISE THE SKIP SELF-TEST CODE.
3585 022354 005301 4$: DEC R1 ;SELECT THE NEXT SET OF DEVICE REGISTERS.
3586 022356 013704 002244 MOV CSRA,R4 ;GET THE ADDRESS OF THE CSR OF THE DUT.
3587 022362 010124 MOV R1,(R4)+ ;SELECT A BANK OF DUT REGISTERS.
3588 022364 010324 6$: MOV R3,(R4)+ ;WRITE THE CODE TO A DUT REGISTER.
3589 022366 020437 002262 CMP R4,TXBFCA ;COMPARE POINTER WITH LAST REGISTER ADDRESS.
3590 022372 103774 BLO 6$ ;LOOP IF NOT ALL REGS DONE IN THIS BANK.
3591 022374 032701 000017 BIT #17,R1 ;TEST FOR IND.ADR.REG FIELD DECREMENTED TO 0.
3592 022400 001365 BNE 4$ ;LOOP UNTIL ALL REGISTERS CONTAIN THE CODE.
3593
3594 022402 60$: PASS ;RESTORE GPRS.
022402 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3595 022404 000207 RTS PC

```

GLOBAL SUBROUTINE

- SWAPO -

```

3597 .SBTTL GLOBAL SUBROUTINE - SWAPO -
3598 ;** *****
3599 ;* - Swap GPRs With GPR Set 0 Routine -
3600 ;* This subroutine swaps the present contents of GPRs R1 thru R5 with
3601 ;* the contents of the number zero GPR save area. The contents of R0
3602 ;* are not altered by this subroutine.
3603 ;*
3604 ;* INPUTS: GPR contents R1 thru R5.
3605 ;* GPRS0B - Label at base of GPR save area number zero.
3606 ;*
3607 ;* OUTPUTS: R1 thru R5 contain the previous contents of GPR save area
3608 ;* zero words 1 thru 5 respectively.
3609 ;* GPRS0 - GPR save area 0 words 1 thru 5, contain previous
3610 ;* contents of GPRs R1 thru R5 respectively.
3611 ;*
3612 ;* CALLING SEQUENCE: JSR PC,SWAPO
3613 ;*
3614 ;* COMMENTS: The state of the CARRY flag is not altered by this routine.
3615 ;*
3616 ;* SUBORDINATE ROUTINES CALLED: None.
3617 ;-- *****
3618
3619 022406 010046 SWAPO:: MOV R0,-(SP) ;SAVE THE CONTENTS OF R0.
3620 ;+
3621 ; Load the stack from the GPRs.
3622 ;-
3623 022410 010146 MOV R1,-(SP) ;SAVE THE CONTENTS OF R1.
3624 022412 010246 MOV R2,-(SP) ;SAVE THE CONTENTS OF R2.
3625 022414 010346 MOV R3,-(SP) ;SAVE THE CONTENTS OF R3.
3626 022416 010446 MOV R4,-(SP) ;SAVE THE CONTENTS OF R4.
3627 022420 010546 MOV R5,-(SP) ;SAVE THE CONTENTS OF R5.
3628 ;+
3629 ; Load the GPRs from the GPR save area 0.
3630 ;-
3631 022422 012700 002450 MOV #GPRS0B,R0 ;GET THE BASE ADDRESS OF GPR SAVE AREA 0.
3632 022426 012001 MOV (R0)+,R1 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 1.
3633 022430 012002 MOV (R0)+,R2 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 2.
3634 022432 012003 MOV (R0)+,R3 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 3.
3635 022434 012004 MOV (R0)+,R4 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 4.
3636 022436 012005 MOV (R0)+,R5 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 5.
3637 ;+
3638 ; Load the GPR save area 0 from the stack.
3639 ;-
3640 022440 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 5 WITH SAVED R5.
3641 022442 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 4 WITH SAVED R4.
3642 022444 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 3 WITH SAVED R3.
3643 022446 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 2 WITH SAVED R2.
3644 022450 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 1 WITH SAVED R1.
3645
3646 022452 012600 MOV (SP)+,R0 ;RESTORE THE INITIAL VALUE OF R0.
3647
3648 022454 000207 RTS PC
    
```


GLOBAL SUBROUTINE

- TXDSBL -

```

3681 .SBTTL GLOBAL SUBROUTINE - TXDSBL -
3682 ;+ *****
3683 ;* - Transmitter Disable -
3684 ;* This subroutine is used to disable transmission on selected lines by,
3685 ;* clearing the associated TX.ENABLE bit on the DUT.
3686 ;*
3687 ;* INPUTS: R5 - Bit's set correspond to lines on which to clear TX.ENABLE.
3688 ;* CSRA - Contains the address of the DUT CSR.
3689 ;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
3690 ;* NUMLNS - Equated to be the maximum number of lines available.
3691 ;* TXAD2A - Contains the address of the TBUFFAD2 register.
3692 ;*
3693 ;* OUTPUTS: R5 - Bit's set indicate the initial states of all TX.ENABLE bits.
3694 ;* TBUFFAD2 - The state of the TX.ENABLE bit may be altered.
3695 ;* The contents of the IND.ADD.REG field in the CSR are destroyed.
3696 ;*
3697 ;* CALLING SEQUENCE: JSR PC,TXDSBL
3698 ;*
3699 ;* COMMENTS:
3700 ;*
3701 ;* SUBORDINATE ROUTINES CALLED: NONE.
3702 ;-- *****
3703
3704 TXDSBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
022570 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3705 022574 010500 MOV R5,R0 ;COPY BIT MAP OF LINES TO DISABLE TRANSMISSION.
3706 022576 012701 000001 MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
3707 022602 013702 002260 MOV TXAD2A,R2 ;GET THE ADDRESS OF THE TBUFFAD2 REGISTER.
3708 022606 005202 INC R2 ;GET THE ADDRESS OF THE MSBYTE OF TBUFFAD2 REG.
3709 022610 012703 000010 MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER PLUS ONE.
3710 022614 013704 002330 MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
3711 022620 005005 CLR R5 ;LOG POSSIBLE TX DISABLED ON ALL LINES.
3712 ;+
3713 ; Select every line in turn, and log the state of each TX.ENABLE bit.
3714 ;--
3715 022622 010477 157416 2$: MOV R4,@CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
3716 022626 105712 TSTB (R2) ;CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.
3717 022630 100001 BPL 4$ ;SKIP NEXT INSTRUCTION IF TX.ENABLE CLEAR.
3718 022632 050105 BIS R1,R5 ;LOG TX ENABLE BIT SET FOR SELECTED LINE.
3719 ;+
3720 ; Clear TX.ENABLE on lines that have a corresponding bit set in the tx disable
3721 ; line bit map.
3722 ;--
3723 022634 030100 4$: BIT R1,R0 ;CHECK STATE OF DISABLE LINE BIT MAP.
3724 022636 001402 BEQ 6$ ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
3725 022640 142712 000200 BICB #BIT7,(R2) ;CLEAR TX.ENABLE BIT ON SELECTED LINE.
3726 022644 005204 6$: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
3727 022646 006301 ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
3728 022650 005303 DEC R3 ;DECREMENT LINE NUMBER.
3729 022652 001363 BNE 2$ ;LOOP TO CHECK NEXT LINE.
3730
3731 022654 000014 60$: PASS R5 ;RESTORE GPRS,EXCEPT
022654 010566 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
022660 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3732 ;R5 - PREVIOUS STATES OF ALL TX.ENABLE BITS.
3733 022662 000207 RTS PC
    
```

GLOBAL SUBROUTINE

- TXENBL -

3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788

022664
022664 004537 003776
022670 010500
022672 012701 000001
022676 013702 002260
022702 005202
022704 012703 000010
022710 013704 002330
022714 005005
022716 010477 157322
022722 105712
022724 100401
022726 050105
022730 030100
022732 001402
022734 152712 000200
022740 005204
022742 006301
022744 005303
022746 001363
022750
022750 010566 000014
022754 004736
022756 000207

```
.SBTTL GLOBAL SUBROUTINE - TXENBL -
;+ *****
;* - Transmitter Enable -
;* This subroutine is used to enable transmission on selected lines by
;* setting the associated TX.ENABLE bit on the DUT.
;*
;* INPUTS: R5 - Bit's set correspond to lines on which to set TX.ENABLE.
;* CSRA - Contains the address of the DUT CSR.
;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
;* NUMLNS - Equated to be the maximum number of lines available.
;* TXAD2A - Contains the address of the TBUFFAD2 register.
;*
;* OUTPUTS: R5 - Bit's set indicate previously disabled lines.
;* TBUFFAD2 - The state of the TX.ENABLE bit may be altered.
;* The contents of the IND.ADD.REG field in the CSR are destroyed.
;*
;* CALLING SEQUENCE: JSR PC,TXENBL
;*
;* COMMENTS:
;*
;* SUBORDINATE ROUTINES CALLED: NONE.
;-- *****

TXENBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
                JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
                MOV R5,R0 ;COPY BIT MAP OF LINES TO ENABLE.
                MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
                MOV TXAD2A,R2 ;GET THE ADDRESS OF THE TBUFFAD2 REGISTER.
                INC R2 ;GET THE ADDRESS OF THE MSBYTE OF TBUFFAD2 REG.
                MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER.
                MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
                CLR R5 ;CLEAR TX.ENABLE BIT LOG OF DISABLED LINES.

;+
; Select every line in turn, and log any TX.ENABLE bit that is clear.
;-
2$: MOV R4,@CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
    TSTB (R2) ;CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.
    BMI 4$ ;SKIP NEXT INSTRUCTION IF TX.ENABLE SET.
    BIS R1,R5 ;LOG TX ENABLE BIT CLEAR FOR SELECTED LINE.

;+
; Set TX.ENABLE on lines that have a corresponding bit set in the tx enable
; line bit map.
;-
4$: BIT R1,R0 ;CHECK STATE OF TX.ENABLE LINE BIT MAP.
    BEQ 6$ ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
    BISB #BIT7,(R2) ;ENABLE TRANSMISSION ON SELECTED LINE.
6$: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
    ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
    DEC R3 ;DECREMENT LINE NUMBER.
    BNE 2$ ;LOOP TO CHECK NEXT LINE.

60$: PASS R5 ;RESTORE GPRS, EXCEPT
                MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
                JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
; R5 - LINE BIT MAP CORRESPONDING TO THE
; PREVIOUS LINES THAT WERE DISABLED.

RTS PC
```

GLOBAL SUBROUTINE

- TXIE0 -

```

3790 .SBTTL GLOBAL SUBROUTINE - TXIE0 -
3791 ;++ *****
3792 ;* - TRANSMITTER INTERRUPT DISABLE -
3793 ;* This routine is used to disable transmitter interrupts in the DHV11.
3794 ;*
3795 ;* INPUTS: NONE.
3796 ;*
3797 ;* OUTPUTS: The TX.INT.ENBL bit is cleared in the DUT CSR.
3798 ;* IESTST -contains the updated status of the TX and RX interrupt
3799 ;* enable bits.
3800 ;*
3801 ;* CALLING SEQUENCE: JSR PC,TXIE0
3802 ;*
3803 ;* COMMENTS: The contents of the indirect address register field in
3804 ;* the DUT CSR are destroyed.
3805 ;*
3806 ;* SUBORDINATE ROUTINES CALLED: NONE.
3807 ;-- *****
3808 022760 010046 TXIE0:: MOV R0,-(SP) ;SAVE CONTENTS OF R0 ON THE STACK.
3809 022762 106746 MFPS -(SP) ;SAVE CURRENT PROCESSOR PRIORITY ON THE STACK.
3810 022764 106427 000340 MTPS #PRI07 ;IGNORE ANY INTERRUPTS THAT MAY BE GENERATED.
3811 022770 042737 177677 002330 BIC #177677,IESTAT ;CLEAR TX.INT.ENBL BIT IN IESTAT.
3812 022776 013777 002330 157240 MOV IESTAT,@CSRA ;DISABLE TX INTERRUPTS.
3813 023004 106426 MTPS (SP)+ ;ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
3814 023006 012600 MOV (SP)+,R0 ;RESTORE R0.
3815 023010 000207 RTS PC
    
```

GLOBAL SUBROUTINE

- TXIE1 -

```

3817 .SBTTL GLOBAL SUBROUTINE - TXIE1 -
3818 ;** *****
3819 ;* - TRANSMITTER INTERRUPT ENABLE -
3820 ;* This routine is used to enable transmitter interrupts in the DHV11.
3821 ;*
3822 ;* INPUTS: NONE.
3823 ;*
3824 ;* OUTPUTS: The TX.INT.ENBL bit is set in the DUT CSR.
3825 ;* IESTST -contains the updated status of the TX and RX interrupt
3826 ;* enable bits.
3827 ;*
3828 ;* CALLING SEQUENCE: JSR PC,TXIE1
3829 ;*
3830 ;* COMMENTS: The contents of the indirect address register field in
3831 ;* the DUT CSR are destroyed.
3832 ;*
3833 ;* SUBORDINATE ROUTINES CALLED: NONE.
3834 ;-- *****
3835
3836 023012 052737 040000 002330 TXIE1:: BIS #BIT14,IESTAT ;SET TX.INT.ENBL BIT IN IESTAT.
3837 023020 042737 137677 002330 BIC #137677,IESTAT ;CLEAR ALL BITS EXCEPT TX RX I.E BITS.
3838 023026 013777 002330 157210 MOV IESTAT,@CSRA ;DISABLE TX INTERRUPTS.
3839 023034 000207 RTS PC
    
```

GLOBAL SUBROUTINE

- UNSDIV -

```

3841 .SBTTL GLOBAL SUBROUTINE - UNSDIV -
3842 ;+ *****
3843 ;* - Unsigned Divide Routine -
3844 ;* This subroutine is used to divide a 32 bit unsigned dividend by a
3845 ;* 16 bit unsigned divisor giving a 16 bit quotient. All numbers are
3846 ;* considered to be unsigned. A success flag is not set on return if
3847 ;* the quotient was too big to be contained in 16 bits.
3848 ;*
3849 ;* INPUTS: R1 - The divisor, unsigned, 16 bits.
3850 ;* R2 - Most significant word of the dividend, unsigned, 16 bits.
3851 ;* R3 - Least significant word of the dividend, unsigned, 16 bits.
3852 ;*
3853 ;* OUTPUTS: R1 - Quotient, unsigned, 16 bits (177777 if overflow).
3854 ;* CARRY - Success flag, set if complete quotient fits in 16 bits.
3855 ;*
3856 ;* CALLING SEQUENCE: JSR PC,UNSDIV
3857 ;*
3858 ;* COMMENTS: If the divisor is 0 the quotient is returned as all ones
3859 ;* (177777) and the carry is clear regardless of the dividend.
3860 ;*
3861 ;* SUBORDINATE ROUTINES CALLED: None.
3862 ;-- *****
3863
3864 023036 UNSDIV:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023036 004537 003776 JSR R5,PREGOS ;CALL REGISTER SAVE SUBRT.
3865
3866 ;+
3867 ; Check for quotient greater than 16 bits condition.
3868 ;-
3868 023042 010204 MOV R2,R4 ;GET MSW OF DIVIDEND FOR SUBTRACT.
3869 023044 160104 SUB R1,R4 ;SUBTRACT DIVISOR FROM MSW OF DIVIDEND.
3870 023046 103403 BCS 2$ ;IF IT DIDN'T GO, WE HAVE QUOTIENT < 16 BITS.
3871 023050 012701 177777 MOV #-1,R1 ;SET QUOTIENT TO ALL ONES (177777),
3872 023054 000442 BR 60$ ;EXIT WITH CARRY CLEAR.
3873
3874 ;+
3875 ; Set up counters and various working GPRs.
3876 ;-
3876 023056 005004 2$: CLR R4 ;CLEAR THE LSW OF THE DIVISOR.
3877 023060 000241 CLC ;CLEAR CARRY FOR THE SHIFT OF THE DIVISOR.
3878 023062 006001 ROR R1 ; DIVISOR BY
3879 023064 006004 ROR R4 ; 2(UNSIGNED)
3880 023066 012700 000020 MOV #16.,R0 ;SET UP INITIAL SHIFT COUNT TO 16.
3881
3882 ;+
3883 ; The subtract and shift loop.
3884 ;-
3884 023072 010246 4$: MOV R2,-(SP) ;SAVE MSWORD OF DIVIDEND.
3885 023074 010346 MOV R3,-(SP) ;SAVE LSWORD OF DIVIDEND.
3886 023076 160403 SUB R4,R3 ;LSWORD DIVIDEND - LSWORD OF DIVISOR.
3887 023100 005602 SBC R2 ;MSWORD DIVIDEND - BORROW .
3888 023102 103402 BCS 6$ ;IF BORROW FROM BORROW SUBTRACT, IT DIDN'T GO.
3889 023104 160102 SUB R1,R2 ;MSWORD DIVIDEND - MSWORD OF DIVISOR.
3890 023106 103003 BCC 8$ ;IF NO BORROW, IT WENT, CARRY IS CLEAR.
3891
3892 ;+
3893 ; It didn't go, so we shift a 1 into the quotient (complemented later).
3894 ; Carry is set.
3895 ;-
3895 023110 012603 6$: MOV (SP)+,R3 ;RESTORE LSWORD OF DIVIDEND.
3896 023112 012602 MOV (SP)+,R2 ;RESTORE MSWORD OF DIVIDEND.

```

GLOBAL SUBROUTINE

- UNSDIV -

```

3897 023114 000401          BR      10$          ;GOTO SHIFT 1 INTO THE QUOTIENT.
3898
3899          ;+
3900          ; It went, so we restore the stack and shift a 0 into quotient (will be
3901          ; complemented later).  Carry is clear.
3902 023116 012626      8$:  MOV      (SP)+,(SP)+      ;POP THE SAVED DIVIDEND OFF OF THE STACK.
3903          ;+
3904          ; Shift the result of the subtract attempt into the quotient shift reg.
3905          ;-
3906 023120 006105      10$:  ROL      R5          ;SHIFT NEXT BIT INTO THE INVERTED QUOTIENT.
3907 023122 000241          CLC          ;DIVIDE THE
3908 023124 006001          ROR      R1          ; DEVISOR BY
3909 023126 006004          ROR      R4          ; 2 (UNSIGNED).
3910 023130 005300          DEC      R0          ;COUNT THIS SHIFT AND SUBTRACT.
3911 023132 001357          BNE     4$          ;LOOP FOR ANOTHER SHIFT & SUB IF NOT DONE.
3912 023134 005105          COM     R5          ;GET QUOTIENT FROM INVERTED QUOTIENT.
3913          ;+
3914          ; Now we either round up or leave quotient alone.
3915          ;-
3916 023136 000241          CLC          ;CLEAR THE CARRY FOR THE SHIFT OF THE DIVIDEND.
3917 023140 006103          ROL      R3          ;MULTIPLY LSWORD OF DIVIDEND BY 2. MSWORD IS 0.
3918 023142 103402          BCS     12$         ;IF CARRY FROM SHIFT, ROUND UP.
3919 023144 160403          SUB     R4,R3         ;SUBTRACT DIVISOR FROM DIVIDEND.
3920 023146 103403          BCS     14$         ;IF BORROW, DON'T ROUND UP.
3921          ;+
3922          ; Round up, extra subtract went.
3923          ;-
3924 023150 005205      12$:  INC     R5          ;INCREMENT THE QUOTIENT BY ONE.
3925 023152 001001          BNE     14$         ;IF NO OVERFLOW, WE LEAVE THE ROUND UP.
3926 023154 005305          DEC     R5          ;DON'T LET ROUNDING CAUSE OVERFLOW.
3927          ;+
3928          ; All done, pass quotient and exit.
3929          ;-
3930 023156 010501      14$:  MOV     R5,R1         ;PASS QUOTIENT BACK IN R1.
3931 023160 000261          SEC          ;INDICATE NO OVERFLOW.
3932
3933 023162          60$:  PASS     R1          ;RESTORE GPRS, LEAVE THE FOLLOWING INTACT:
          023162 010166 000004          MOV     R1,R1SLOT(SP) ;PUT R1 IN STACK SLOT.
          023166 004736          JSR     PC,@(SP)+    ;RETURN TO PREG05 SUBRT.
3934          ;R1 - 16 BIT, UNSIGNED QUOTIENT,
3935 023170 000207          RTS     PC          ;CARRY - SET INDICATES NO OVERFLOW (SUCCESS).

```

GLOBAL SUBROUTINE

- WAIBIC -

```

3937 .SBTTL GLOBAL SUBROUTINE - WAIBIC -
3938 ;** *****
3939 ;* - Wait For Bit Clear Routine -
3940 ;* This subroutine waits for the specified bit to become clear. If the
3941 ;* specified bit goes to a clear state within the specified time-out
3942 ;* period a success indication is returned by this routine.
3943 ;* The last value which is read looking for the condition is returned to
3944 ;* allow the use of this routine to look for destructive read conditions.
3945 ;*
3946 ;* INPUTS: R1 - Time-out value and bit number indication:
3947 ;* Bits 15 thru 12 - Number of bit to test (range 0 thru 15).
3948 ;* Bits 11 thru 0 - Time-out value in milli-seconds (4095 max).
3949 ;* R2 - Address of word containing the bit to test.
3950 ;* MSLCNT.
3951 ;*
3952 ;* OUTPUTS: R2 - The last word which was read to check for the condition.
3953 ;* CARRY - Success flag (CARRY set if bit clr before time-out).
3954 ;*
3955 ;* CALLING SEQUENCE: MOV #130040,R1 ;PASS BIT 11 (13 OCTAL) AND
3956 ;* ; 32 (40 OCTAL) MS DELAY.
3957 ;* MOV #LABEL,R2 ;TEST BIT IN WORD AT "LABEL".
3958 ;* JSR PC,WAIBIC ;WAIT 32 MS FOR BIT 11 TO CLR.
3959 ;*
3960 ;* COMMENTS:
3961 ;*
3962 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
3963 ;-- *****
3964
3965 023172 WAIBIC:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023172 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3966 023176 010204 MOV R2,R4 ;SET UP THE ADDRESS PARAMETER FOR MSLGET.
3967 023200 010102 MOV R1,R2
3968 023202 042701 170000 BIC #170000,R1 ;SEPERATE DELAY COUNT OUT OF PASSED PARAMETER.
3969 023206 042702 007777 BIC #7777,R2 ;SEPERATE LINE NUMBER FIELD OF PASSED PARAM.
3970 023212 000302 SWAB R2 ;PUT LINE NUMBER FIELD IN LSBYTE.
3971 023214 006202 ASR R2 ;SHIFT THE LINE NUMBER FIELD INTO THE PROPER
3972 023216 006202 ASR R2 ; POSITION TO USE IT AS A WORD TABLE OFFSET
3973 023220 006202 ASR R2 ; FOR THE TABLE LOOKUP OF THE LINE BIT MAP.
3974 023222 016202 002410 MOV BITTBL(R2),R2 ;GET BIT MAP OF LINE TO TEST FROM TABLE.
3975 023226 005003 CLR R3 ;INDICATE THAT THE BIT SHOULD BE CLR.
3976 023230 004737 020340 JSR PC,MSLGET ;WAIT FOR THE BIT TO BE CLR WITHIN TIME-OUT.
3977 ; CARRY IS CORRECT UPON MSLGET RETURN.
3978 023234 010002 MOV R0,R2 ;PASS LAST VALUE READ AS OUTPUT PARAMETER.
3979 023236 010266 000006 60: PASS R2 ;RESTORE GPRS, EXCEPT THE FOLLOWING:
023236 004736 MOV R2,R2SLOT(SP) ;PUT R2 IN STACK SLOT.
023242 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3980 ; R2 - LAST VALUE READ LOOKING FOR CONDITION.
3981 023244 000207 RTS PC ; CARRY - SUCCESS FLAG (SET IF BIT FOUND CLR).

```

GLOBAL SUBROUTINE

- WAIBIS -

```

3983 .SBTTL GLOBAL SUBROUTINE - WAIBIS -
3984 ;* *****
3985 ;* - Wait For Bit Set Routine -
3986 ;* This subroutine waits for the specified bit to become set. If the
3987 ;* specified bit goes to a set state within the specified time-out
3988 ;* period a success indication is returned by this routine.
3989 ;* The last value which is read looking for the condition is returned to
3990 ;* allow the use of this routine to look for destructive read conditions.
3991 ;*
3992 ;* INPUTS: R1 - Time-out value and bit number indication:
3993 ;* Bits 15 thru 12 - Number of bit to test (range 0 thru 15).
3994 ;* Bits 11 thru 0 - Time-out value in milli-seconds (4095 max).
3995 ;* R2 - Address of word containing the bit to test.
3996 ;* MSLCNT.
3997 ;*
3998 ;* OUTPUTS: R2 - The last word which was read to check for the condition.
3999 ;* CARRY - Success flag (CARRY set if bit set before time-out).
4000 ;*
4001 ;* CALLING SEQUENCE: MOV #130040,R1 ;PASS BIT 11 (13 OCTAL) AND
4002 ;* ; 32 (40 OCTAL) MS DELAY.
4003 ;* MOV #LABEL,R2 ;TEST BIT IN WORD AT "LABEL".
4004 ;* JSR PC,WAIBIS ;WAIT 32 MS FOR BIT 11 TO SET.
4005 ;*
4006 ;* COMMENTS:
4007 ;*
4008 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
4009 ;* -- *****
4010
4011 023246 WAIBIS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4012 023246 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4013 023252 010204 MOV R2,R4 ;SET UP THE ADDRESS PARAMETER FOR MSLGET.
4014 023254 010102 MOV R1,R2
4015 023256 042701 170000 BIC #170000,R1 ;SEPERATE DELAY COUNT OUT OF PASSED PARAMETER.
4016 023262 042702 007777 BIC #7777,R2 ;SEPERATE LINE NUMBER FIELD OF PASSED PARAM.
4017 023270 006202 SWAB R2 ;PUT LINE NUMBER FIELD IN LSBYTE.
4018 023272 006202 ASR R2 ;SHIFT THE LINE NUMBER FIELD INTO THE PROPER
4019 023274 006202 ASR R2 ; POSITION TO USE IT AS A WORD TABLE OFFSET
4020 023276 016202 002410 ASR R2 ; FOR THE TABLE LOOKUP OF THE LINE BIT MAP.
4021 023302 010203 MOV BITTBL(R2),R2 ;GET BIT MAP OF LINE TO TEST FROM TABLE.
4022 023304 004737 020340 MOV R2,R3 ;INDICATE THAT THE BIT SHOULD BE SET.
4023 JSR PC,MSLGET ;WAIT FOR THE BIT TO BE SET WITHIN TIME-OUT.
4024 023310 010002 ; CARRY IS CORRECT UPON MSLGET RETURN.
4025 023312 010266 000006 60: MOV R0,R2 ;PASS LAST VALUE READ AS OUTPUT PARAMETER.
4026 023316 004736 JSR PC,@(SP)+ ;RESTORE GPRS, EXCEPT THE FOLLOWING:
4027 023320 000207 RTS PC ; R2 - LAST VALUE READ LOOKING FOR CONDITION.
; CARRY - SUCCESS FLAG (SET IF BIT FOUND SET).

```


GLOBAL SUBROUTINE

- WDPDR -

```

4029 .SBTTL GLOBAL SUBROUTINE - WDPDR -
4030 ;*+ *****
4031 ;* - Write Data Pattern to Device Registers -
4032 ;* This routine writes a rotated data pattern to each of the 6 device
4033 ;* registers of each active line of the device under test.
4034 ;* The data pattern is rotated once after each write to a device register
4035 ;* on a particular line. The starting data pattern for each line
4036 ;* is rotated once after writing all the registers on a particular
4037 ;* line. This leads to the following data pattern:
4038 ;* Line 0, register 0 - shifted 0 bit positions
4039 ;* Line 0, register 1 - shifted 1 bit position
4040 ;*
4041 ;* Line 1, register 0 - shifted 1 bit position
4042 ;* Line 2, register 1 - shifted 2 bit positions
4043 ;*
4044 ;*
4045 ;* Any bits fields in the device registers that cannot be altered
4046 ;* are masked out of the data pattern before it is written.
4047 ;* This routine will use either MOV, MOVB, BIS, BISB, BIC, or BICB
4048 ;* instructions. The upper or lower byte can be specified for writing.
4049 ;*
4049 ;* INPUTS: R2 - Used to pass in the data pattern to be rotated & written.
4050 ;* R3 - Byte indicator (- => lo byte, + => hi byte, 0 => both).
4051 ;* R4 - Operation type indicator (- => BIC, + => BIS, 0 => MOV).
4052 ;* ACTLNS - Bit map of the active lines on the device under test.
4053 ;* CSRA - Contains the CSR address of the Device under test.
4054 ;* DRADRT - Base address of device register address table.
4055 ;* LPRO - Equated to LPR reg offset from device CSR address.
4056 ;* NUMLNS - Number of lines on the device under test.
4057 ;* TXBFCD - Equated to TBUFFCT reg offset from device CSR address.
4058 ;* UNBTB - Base address of the unused bit table.
4059 ;*
4060 ;* OUTPUTS: Device registers on all active device lines are modified.
4061 ;*
4062 ;* CALLING SEQUENCE: JSR PC,WDPDR
4063 ;*
4064 ;* COMMENTS: This routine does not write any data to the TX.CHAR registers.
4065 ;* The CSR is cleared except for the IND.ADR.REG field.
4066 ;*
4067 ;* SUBORDINATE ROUTINES CALLED: ROLDAP.
4068 ;*-- *****
4069
4070 023322 WDPDR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023322 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4071 ;*
4072 ;* Set up outer loop which writes the data pattern to each line's registers
4073 ;*
4074 023326 005005 CLR R5 ;CLEAR LINE COUNTER TO SELECT LINE 0.
4075 ;*
4076 ;* The outer loop follows. Each pass through this loop writes data to all of
4077 ;* the device registers for a particular line if it is active.
4078 ;*
4079 023330 010204 20: MOV R2,R4 ;SAVE THE OUTER LOOP DATA PATTERN.
4080 023332 010577 156706 MOV R5,BCSRA ;SET CSR IND.ADR.REG FIELD TO THIS LINE.
4081 023336 006305 ASL R5 ;TURN LINE NUMBER INTO A WORD OFFSET.
4082 023340 036537 002410 002236 BIT BITTBL(R5),ACTLNS
4083 023346 001451 BEQ 20: ;LINE ACTIVE? NO, SKIP THIS LINE.
4084 023350 012701 000004 MOV 0LPRO,R1 ;YES, INITIALIZE THE REGISTER OFFSET.

```

GLOBAL SUBROUTINE

- WDPDR -

```

4085      ;+
4086      ;   The inner loop follows.  Each pass through this loop writes data to a
4087      ;   device register.
4088      ;-
4089 023354 010200      4$:   MOV    R2,R0
4090 023356 046100 002264   BIC    UNBTTB(R1),R0 ;CLEAR BIT FIELDS FOR UNUSED REGISTER BITS.
4091 023362 016103 002244   MOV    DRADRT(R1),R3 ;GET THE ADDRESS OF THE DEVICE REGISTER.
4092 023366 005766 000010   TST    R3SLOT(SP) ;CHECK THE OPERAND TYPE INDICATOR.
4093 023372 003402       BLE    6$ ;HIGH BYTE? NO, SKIP HIGH BYTE ADDRESS SET UP.
4094 023374 005203       INC    R3 ;YES, SET THE REG ADDRESS TO THE HIGH BYTE.
4095 023376 000300       SWAB   R0 ;MOVE HIGH BYTE DATA INTO THE LOW BYTE.
4096 023400 005766 000010   6$:   TST    R3SLOT(SP) ;CHECK THE OPERAND TYPE INDICATOR.
4097 023404 001412       BEQ    12$ ;WORD ACCESS? YES, GO PERFORM WORD ACCESS.
4098      ;+
4099      ;Perform byte access to the specified byte of the specified register.
4100      ;-
4101 023406 005766 000012   TST    R4SLOT(SP) ;NO, CHECK THE ACCESS TYPE INDICATOR.
4102 023412 100403       BMI    8$ ;USE BIC? YES, GO PERFORM BICB INSTRUCTION.
4103 023414 001404       BEQ    10$ ;USE MOV? YES, GO PERFORM MOV B INSTRUCTION.
4104 023416 150013       BISB   R0,(R3) ;NEITHER. PERFORM BISB ACCESS TO REGISTER.
4105 023420 000415       BR     18$
4106 023422 140013   8$:   BICB   R0,(R3) ;PERFORM BICB ACCESS TO REGISTER.
4107 023424 000413       BR     18$
4108 023426 110013   10$:  MOVB   R0,(R3) ;PERFORM MOV B ACCESS TO REGISTER.
4109 023430 000411       BR     18$
4110      ;+
4111      ;Perform word access to the specified register.
4112      ;-
4113 023432 005766 000012   12$:  TST    R4SLOT(SP) ;CHECK THE ACCESS TYPE INDICATOR.
4114 023436 100403       BMI    14$ ;USE BIC? YES, GO PERFORM BIC INSTRUCTION.
4115 023440 001404       BEQ    16$ ;USE MOV? YES, GO PERFORM MOV INSTRUCTION.
4116 023442 050013       BIS    R0,(R3) ;NEITHER. PERFORM BIS ACCESS TO REGISTER.
4117 023444 000403       BR     18$
4118 023446 040013   14$:  BIC    R0,(R3) ;PERFORM BIC ACCESS TO REGISTER.
4119 023450 000401       BR     18$
4120 023452 010013   16$:  MOV    R0,(R3) ;PERFORM MOV ACCESS TO REGISTER.
4121      ;+
4122      ; Prepare the data pattern and offset for the next register on this line.
4123      ;-
4124 023454 004737 021570   18$:  JSR    PC,ROLDAP ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
4125 023460 062701 000002   ADD    #2,R1 ;INCREMENT OFFSET FOR NEXT REGISTER.
4126 023464 020127 000016   CMP    R1,#TXBFCO ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
4127 023470 003731       BLE    4$ ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
4128      ;+
4129      ; Back into the outer loop. Now set up for next line. Loop if not done.
4130      ;-
4131 023472 010402   20$:  MOV    R4,R2 ;SET UP TO ROTATE THE DATA PATTERN.
4132 023474 004737 021570   JSR    PC,ROLDAP ;ROTATE THE DATA PATTERN.
4133 023500 006205       ASR    R5 ;CONVERT BACK TO LINE NUMBER FROM WORD OFFSET.
4134 023502 005205       INC    R5 ;COUNT THIS LINE.
4135 023504 020527 000010   CMP    R5,#NUMLNS ;COMPARE LINE COUNT WITH NUMBER OF LINES.
4136 023510 002707       BLT    2$ ;LOOP IF SOME LINES NOT DONE.
4137      ;+
4138 023512   60$:  PASS ;RESTORE GPRS.
4139 023512 004736       RTS    PC JSR    PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4139 023514 000207

```

GLOBAL SUBROUTINE

- WTWLNC -

```

4141 .SBTTL GLOBAL SUBROUTINE - WTWLNC -
4142 ;* *****
4143 ;* - Line Control Register Setup Routine -
4144 ;* This subroutine is used to set the Device Under Test (DUT) Line
4145 ;* Control Registers (LNCTRL) to the specified state. Only the LNCTRLS
4146 ;* for the specified lines are altered.
4147 ;*
4148 ;* INPUTS: R0 - New line parameters.
4149 ;* R5 - Bit map of lines to be altered.
4150 ;* CSRA - Contains address of the DUT CSR.
4151 ;* IESTAT - Contains the current state of the TX and RX interrupt
4152 ;* enable bits in the CSR.
4153 ;* LNCTRA - Contains address of the DUT LNCTRL registers.
4154 ;*
4155 ;* OUTPUTS: LNCTRL - Specified DUT Line Control Registers are altered.
4156 ;*
4157 ;* CALLING SEQUENCE: JSR PC,WTWLNC
4158 ;*
4159 ;* COMMENTS:
4160 ;*
4161 ;* SUBORDINATE ROUTINES CALLED: ALTFLD.
4162 ;* - *****
4163
4164 023516 WTWLNC:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023516 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4165
4166 ;*
4167 ; Set up the parameters for the call to ALTFLD.
4168 023522 013701 002254 ;-
4169 023526 010002 MOV LNCTRA,R1 ;SET UP THE REGISTER ADDRESS PARAMETER.
4170 023530 010503 MOV R0,R2 ;SET UP THE DESIRED REGISTER CONTENTS.
4171 023532 012704 177777 MOV R5,R3 ;SET UP THE BIT MAP OF LINES TO ALTER.
4172 MOV #-1,R4 ;SELECT ALL REGISTER BITS TO BE ALTERED.
4173 ;*
4174 ; Call the subroutine which alters the register contents.
4175 023536 004737 017630 ;-
4176 JSR PC,ALTFLD ;ALTER THE REGISTER CONTENTS.
4177 023542 60$: PASS ;RESTORE GPRS.
023542 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4178 023544 000207 RTS PC

```

GLOBAL SUBROUTINE

- WTWLNS -

```

4180 .SBTTL GLOBAL SUBROUTINE - WTWLNS -
4181 ;*****
4182 ;* - Write Word to all Lines routine -
4183 ;* This subroutine writes a specified word to the specified DHV device
4184 ;* register for each of the DHV lines. It could be used to clear all
4185 ;* of the LNCTRL registers or to initialize all of the LPR registers to
4186 ;* the same parameters.
4187 ;*
4188 ;* INPUTS: R1 - Address of the specified registers.
4189 ;* R2 - Word to write into the specified registers.
4190 ;* IESTAT - Saved states of the TX.IE and RX.IE bits.
4191 ;* MAPLNS - Equated to bit map of lines on device (8 for DHV11).
4192 ;* CSRA.
4193 ;*
4194 ;* OUTPUTS: DEVICE REGISTERS - Specified registers given new value.
4195 ;* CSR IND.ADR.REG field - Destroyed.
4196 ;* CSR Interrupt Enable bits - Set to states in IESTAT.
4197 ;*
4198 ;* CALLING SEQUENCE: JSR PC,WTWLNS
4199 ;*
4200 ;* COMMENTS: Note that the specified registers for all lines are altered
4201 ;* by this routine. This routine should NOT be used to alter
4202 ;* the states of partial register fields or to alter a register
4203 ;* for fewer than all of the lines.
4204 ;* The specified registers are read before being written.
4205 ;*
4206 ;* SUBROUTINES CALLED: ALTFLD.
4207 ;*****
4208
4209 023546 004537 003776 WTWLNS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
; R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4210 ;+
4211 ; Set up the bit map of lines to change and mask of bits to alter parameters.
4212 ;-
4213 023552 012703 000377 MOV #MAPLNS,R3 ;GET THE BIT MAP OF LINES TO CHANGE.
4214 023556 012704 177777 MOV #-1,R4 ;INDICATE ALL 16 BITS TO BE CHANGED.
4215 ;+
4216 ; Call the subroutine to write the specified registers.
4217 ;-
4218 023562 004737 017630 JSR PC,ALTFLD ;CHANGE THE REGISTERS.
4219
4220 023566 004736 60$: PASS ;RESTORE GPRS.
; PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4221 023570 000207 RTS PC

```

GLOBAL SUBROUTINE

- WTWLPR -

```

4223 .SBTTL GLOBAL SUBROUTINE - WTWLPR -
4224 ;* *****
4225 ;* - Line Parameter Register Setup Routine -
4226 ;* This subroutine is used to set the Device Under Test (DUT) Line
4227 ;* Parameter Registers (LPR) to the specified state. Only the LPRs for
4228 ;* the specified lines are altered.
4229 ;*
4230 ;* INPUTS: R0 - New line parameters.
4231 ;* R5 - Bit map of lines to be altered.
4232 ;* CSRA - Contains address of the DUT CSR.
4233 ;* IESTAT - Contains the current state of the TX and RX interrupt
4234 ;* enable bits in the CSR.
4235 ;* LPRA - Contains address of the DUT LPR.
4236 ;*
4237 ;* OUTPUTS: LPR - Specified DUT Line Parameter Registers are altered.
4238 ;*
4239 ;* CALLING SEQUENCE: JSR PC,WTWLPR
4240 ;*
4241 ;* COMMENTS:
4242 ;*
4243 ;* SUBORDINATE ROUTINES CALLED: ALTFLD.
4244 ;* -- *****
4245
4246 023572 WTWLPR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023572 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4247
4248 ;*
4249 ; Set up the parameters for the call to ALTFLD.
4250 ; -
4250 023576 013701 002250 MOV LPRA,R1 ;SET UP THE REGISTER ADDRESS PARAMETER.
4251 023602 010002 MOV R0,R2 ;SET UP THE DESIRED REGISTER CONTENTS.
4252 023604 010503 MOV R5,R3 ;SET UP THE BIT MAP OF LINES TO ALTER.
4253 023606 012704 177777 MOV #-1,R4 ;SELECT ALL REGISTER BITS TO BE ALTERED.
4254 ;*
4255 ; Call the subroutine which alters the register contents.
4256 ; -
4257 023612 004737 017630 JSR PC,ALTFLD ;ALTER THE REGISTER CONTENTS.
4258
4259 023616 004736 60$: PASS ;RESTORE GPRS.
023616 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4260 023620 000207 RTS PC

```

INTERRUPT SERVICE ROUTINE - CACHRX -

```

4262 .SBTTL INTERRUPT SERVICE ROUTINE - CACHRX -
4263 ;** *****
4264 ;* - Catch Receiver interrupt.
4265 ;* This routine is used in several tests, to log a count of the
4266 ;* number of receiver interrupts that occur.
4267 ;*
4268 ;* INPUTS: CSRA - Contains the address of the CSR.
4269 ;* RXINTC - Holds the count of the number of RX interrupts
4270 ;* that occurred.
4271 ;*
4272 ;* OUTPUTS: RXINTC - Contains the updated interrupt count.
4273 ;*
4274 ;*
4275 ;* CALLING SEQUENCE: Put the address of the label CACHRX in the vector
4276 ;* location.
4277 ;*
4278 ;* COMMENTS:
4279 ;*
4280 ;* SUBORDINATE ROUTINES CALLED: none
4281 ;-- *****
4282
4283 023622 004537 003776 CACHRX::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023622 013701 002334 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4284 023626 013701 002334 MOV RXINTC,R1 ;GET THE RECEIVER INTERRUPT COUNT
4285 023632 005201 INC R1 ;INCREMENT THE COUNT
4286 023634 102001 BVC 2$ ;BRANCH IF NO OVERFLOW OCCURRED
4287 023636 005301 DEC R1 ;RESET THE COUNT TO 177777
4288 023640 010137 002334 2$: MOV R1,RXINTC ;SAVE NEW COUNT VALUE
4289 023644 010137 002334 60$: PASS ;RESTORE GPRS.
023644 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4290 023646 000002 RTI
    
```

INTERRUPT SERVICE ROUTINE - CACHTX -

```

4292 .SBTTL INTERRUPT SERVICE ROUTINE - CACHTX -
4293 ;** *****
4294 ;* - Catch Transmitter interrupt.
4295 ;* This routine is used in several tests, to log a count of the
4296 ;* number of transmission interrupts that occur.
4297 ;*
4298 ;* INPUTS: CSRA - Contains the address of the CSR.
4299 ;* TXINTC - Holds the count of the number of TX interrupts
4300 ;* that occurred.
4301 ;*
4302 ;* OUTPUTS: TXINTC - Contains the updated interrupt count.
4303 ;*
4304 ;*
4305 ;* CALLING SEQUENCE: Put the address of the label CACHTX in the vector
4306 ;* location.
4307 ;*
4308 ;* COMMENTS:
4309 ;*
4310 ;* SUBORDINATE ROUTINES CALLED: none
4311 ;-- *****
4312
4313 023650 CACHTX::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
023650 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4314 023654 013701 002340 MOV TXINTC,R1 ;GET THE TRANSMISSION INTERRUPT COUNT
4315 023660 005201 INC R1 ;INCREMENT THE COUNT
4316 023662 102001 BVC 2$ ;BRANCH IF NO OVERFLOW OCCURRED
4317 023664 005301 DEC R1 ;RESET THE COUNT TO 17777
4318 023666 010137 002340 2$: MOV R1, TXINTC ;SAVE NEW COUNT VALUE
4319 023672 004736 60$: PASS ;RESTORE GPRS.
023672 000002 RTI JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4320 023674 000002

```

INTERRUPT SERVICE ROUTINE - CLKINT -

```

4322 .SBTTL INTERRUPT SERVICE ROUTINE - CLKINT -
4323 ;* *****
4324 ;* This routine is executed CLKHRZ times per second. It decrements the
4325 ;* two timer counters down to zero.
4326 ;*
4327 ;* INPUTS: TIMER1 - Timer counter #1.
4328 ;* TIMER2 - Timer counter #2.
4329 ;* TIMER3 - Timer counter for call of BREAK macro.
4330 ;*
4331 ;* OUTPUTS: The 2 timer counters are decremented if they are not zero.
4332 ;*
4333 ;* CALLING SEQUENCE: Put #CLKINT in the clock interrupt vector slot.
4334 ;* Put the desired time period (seconds times CLKHRZ) in
4335 ;* either TIMER1 or TIMER2 and poll the respective timer
4336 ;* counter to detect its going to 0 on time-out.
4337 ;*
4338 ;* COMMENTS: The 2 counters will not wraparound but will stop at 0. This
4339 ;* allows the detection of a time-out any time after the time-out
4340 ;* has occurred until the timer counter is set to another value.
4341 ;*
4342 ;* SUBORDINATE ROUTINES CALLED: None.
4343 ;-- *****
4344
4345 023676 005737 002364 CLKINT:: TST TIMER1 ;CHECK FOR TIMER1 AT ZERO.
4346 023702 001402 BEQ 2$ ;BRANCH TO LEAVE IT AT ZERO IF IT IS ZERO.
4347 023704 005337 002364 DEC TIMER1 ;DECREMENT TIME COUNT.
4348 023710 005737 002366 2$: TST TIMER2 ;CHECK FOR TIMER2 AT ZERO.
4349 023714 001402 BEQ 4$ ;BRANCH TO LEAVE IT ALONE IF IT'S ALREADY ZERO.
4350 023716 005337 002366 DEC TIMER2 ;DECREMENT TIME COUNT.
4351 023722 005337 002370 4$: DEC TIMER3 ;DECREMENT THE BREAK COUNT.
4352 023726 001006 BNE 60$ ;EXIT IF NOT TIME TO CALL BREAK.
4353 023730 013737 002372 002370 MOV BCOUNT,TIMER3 ;SET UP TIME TILL NEXT BREAK.
4354 023736 010046 MOV RO,-(SP) ;SAVE CONTENTS OF RO FROM BREAK MACRO.
4355 023740 BREAK ;CHECK FOR OPERATOR CONTROL/C. TRAP C$BRK
4356 023742 012600 MOV (SP)+,RO ;RESTORE CONTENTS OF RO.
4357 023744 000002 60$: RTI
    
```


INTERUPT SERVICE ROUTINE

- RXBRRT -

```

4359 .SBTTL INTERUPT SERVICE ROUTINE - RXBRRT -
4360 ;** *****
4361 ;* - BR Level Test Receive Interrupt Service Routine -
4362 ;* This service routine handles Receive Interrupts during the Interrupt
4363 ;* BR Level Test. This routine counts the interrupt and sets a flag
4364 ;* to indicate that the interrupt has occurred. It also checks the
4365 ;* flag which indicates that a TX interrupt has occurred. If the TX
4366 ;* interrupt flag is set, this routine sets an interrupt order error
4367 ;* flag indicating that a transmit interrupt was serviced before a
4368 ;* simultaneous receive interrupt.
4369 ;*
4370 ;* INPUTS: RXINTC - Holds the count of the number of RX interupts.
4371 ;* RXINTF - RX Interrupt flags.
4372 ;*
4373 ;* OUTPUTS: RXINTC - Contains the updated interupt count.
4374 ;* RXINTF - RX Int flags:
4375 ;* (Bit 0 set, bit 14 set if TXINTF bit 0 is set.)
4376 ;*
4377 ;* CALLING SEQUENCE: Put the address of the label RXBRRT in the vector
4378 ;* location.
4379 ;*
4380 ;* COMMENTS: NOTE: The FIFO is purged by this routine.
4381 ;*
4382 ;* SUBORDINATE ROUTINES CALLED: None.
4383 ;-- *****
4384
4385 023746 RXBRRT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4386 023746 004537 003776 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4387 023752 017700 156270 MOV @RBUFA,R0 ;READ THE CHAR OUT OF THE FIFO.
4388 023756 013701 002334 MOV RXINTC,R1 ;GET THE INTERUPT COUNT.
4389 023762 005201 INC R1 ;INCREMENT THE COUNT.
4390 023764 001402 BEQ 2$ ;BYPASS UPDATING COUNT IF OVERFLOW OCCURRED.
4391 023766 010137 002334 MOV R1,RXINTC ;SAVE NEW COUNT VALUE.
4392 023772 013701 002336 2$: MOV RXINTF,R1 ;GET THE RX INTERRUPT FLAGS.
4393 023776 052701 000001 BIS #BIT0,R1 ;SET THE RX INTERRUPT HAS OCCURRED FLAG.
4394 024002 032737 000001 002342 BIT #BIT0,TXINTF ;TEST THE "TX INT HAS OCCURRED" FLAG.
4395 024010 001402 BEQ 4$ ;SKIP SETTING ERROR FLAG IF NO TX INT.
4396 024012 052701 040000 BIS #BIT14,R1 ;SET THE INTERRUPT ORDER ERROR FLAG.
4397 ;+
4398 ; 8 FIFO codes will cause 8 interrupts, after these 8 codes we don't want
4399 ; to check the interrupt order, because perhaps a BMP code has come in
4400 ; between the servicing of the 8 FIFO code interrupts and the servicing
4401 ; of one of the TX interrupts.
4402 024016 023737 002334 000010 4$: CMP RXINTC,NUMLNS ;TEST FOR ALL SELFTEST CODE INTS DONE.
4403 024024 003002 BGT 60$ ;SKIP UPDATING RX INT FLAGS IF EXTRA RX INTS.
4404 024026 010137 002336 MOV R1,RXINTF ;UPDATE THE RX INTERRUPT FLAGS.
4405 024032 004736 60$: PASS ;RESTORE GPRS.
4406 024034 000002 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
RTI

```

INTERUPT SERVICE ROUTINE - RXINPT -

```

4408 .SBTTL INTERUPT SERVICE ROUTINE - RXINPT -
4409 ;* *****
4410 ;* - Receive Character Input Interrupt Service Routine -
4411 ;* This service routine inputs a character from the DUT and loads the
4412 ;* char (complete with status flags) into a receive char buffer in
4413 ;* memory. The interrupt is also counted. The receive char buffer is
4414 ;* monitored to ensure that it does not overflow.
4415 ;*
4416 ;* INPUTS: BUFEND - Labels the end of the host memory buffer.
4417 ;*          BUFPTR - Contains address of next free buffer location.
4418 ;*          CSRA - Contains the address of the DUT CSR.
4419 ;*          RBUFA - Contains the address of the RBUF DUT register.
4420 ;*          RXINTC - Holds the count of the number of RX interupts.
4421 ;*          RXINTF - RX Interrupt flags.
4422 ;*
4423 ;* OUTPUTS: BUFPTR - Contains updated address of next free buffer location.
4424 ;*          RXINTC - Contains the updated interupt count.
4425 ;*          RXINTF - RX Int flags (bit 15 set if RX.DATA.AVAIL is clear).
4426 ;*
4427 ;* CALLING SEQUENCE: Put the address of the label RXINPT in the vector
4428 ;*                  location.
4429 ;*
4430 ;* COMMENTS: In case of overflow of the memory buffer, BUFPTR will be
4431 ;*           maintained equal to BUFEND and the word at BURFPTR will be
4432 ;*           the last word read from the DUT FIFO.
4433 ;*           NOTE: This routine can destroy TX.ACTIONS by reading the CSR.
4434 ;*
4435 ;* SUBORDINATE ROUTINES CALLED: None.
4436 ;*
4437 ;*
4438 RXINPT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4439 024036 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4440 024042 032777 000200 156174 BIT #BIT7,@CSRA ;TEST RX.DATA.AVAIL. BIT OF THE CSR (READS CSR).
4441 024050 001003 BNE 2$ ;BRANCH AROUND SETTING FLAG IF BIT IS SET.
4442 024052 052737 100000 002336 BIS #BIT15,RXINTF ;SET THE RX.DATA.AVAIL CLEAR FLAG.
4443 024064 005201 2$: MOV RXINTC,R1 ;GET THE INTERRUPT COUNT.
4444 024066 001402 INC R1 ;INCREMENT THE COUNT.
4445 024070 010137 002334 BEQ 4$ ;BYPASS UPDATING COUNT IF OVERFLOW OCCURRED.
4446 024074 013702 002324 4$: MOV R1,RXINTC ;SAVE NEW COUNT VALUE.
4447 024100 017722 156142 MOV BUFPTR,R2 ;GET THE POINTER TO NEXT FREE BUFFER WORD.
4448 024104 020237 003726 MOV @RBUFA,(R2)+ ;READ A CHAR FROM THE FIFO INTO BUFFER.
4449 024110 103002 CMP R2,BUFEND ;TEST FOR POINTER BEYOND END OF BUFFER.
4450 024112 010237 002324 BHIS 60$ ;SKIP THE PTR UPDATE IF PTR OUT OF BOUNDS.
4451 024116 004736 60$: MOV R2,BUFPTR ;UPDATE THE BUFFER POINTER.
4452 024120 000002 PASS ;RESTORE GPRS.
RTI JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL TRAP SERVICE ROUTINE - TP4RTN -

```

4454 .SBTTL GLOBAL TRAP SERVICE ROUTINE - TP4RTN -
4455 ;*****
4456 ;* Bus Time-out Trap (004 trap) Service Routine -
4457 ;* This routine is used during the Device Register Address Access Test.
4458 ;* It determines if the 004 trap was caused by an "expected" error or
4459 ;* not by examining the return PC value on the stack. If the trap is
4460 ;* unexpected, this routine jumps to the normal Diagnostic Supervisor
4461 ;* 004 trap handling routine.
4462 ;*
4463 ;* INPUTS: SP - Points to the PC where the trap occurred.
4464 ;* ADRPTR - Label at the address where "expected" traps occur.
4465 ;* TP4FLG - 004 trap flags.
4466 ;*
4467 ;* OUTPUTS: TP4FLG - Bit 15 is set if "expected" trap occurred.
4468 ;*
4469 ;* CALLING SEQUENCE: Put address pointed to by TP4RTN in 004 vector.
4470 ;* Occurrence of 004 trap vectors to this routine.
4471 ;*
4472 ;* COMMENTS: Any 004 trap which occurs at an address other than that labeled
4473 ;* ADRPTR will be handled by the normal 004 trap service routine.
4474 ;*
4475 ;* SUBORDINATE ROUTINES CALLED: None.
4476 ;*****
4477
4478 024122 021627 020140 TP4RTN:: CMP (SP),ADRPTR ;COMPARE EXPECTED ADR AGAINST TRAP RET PC.
4479 024126 001402 BEQ 2$ ;IF THEY MATCH, CONTINUE THIS ROUTINE.
4480 024130 000177 156210 JMP @TP4VEC ;IF NOT,JUMP TO NORMAL 004 TRAP SERVICE RTN.
4481 024134 052737 100000 002346 2$: BIS @BIT15,TP4FLG ;SET THE 004 TRAP OCCURED FLAG.
4482 024142 000002 RTI ;ALL DONE, GO BACK TO THE TEST.
    
```

INTERUPT SERVICE ROUTINE

- TXINTR -

```

4484 .SBTTL INTERUPT SERVICE ROUTINE - TXINTR -
4485 ;** *****
4486 ;* - Transmit Interrupt Service Routine -
4487 ;* This routine handles a transmit interrupt from the Device Under Test
4488 ;* (DUT) by counting the interrupt and reading the DUT CSR to clear the
4489 ;* interrupt request. This routine also sets a flag to indicate that
4490 ;* a TX interrupt has occurred and sets a flag if the TX.ACTION bit is
4491 ;* not set in the read contents of the DUT CSR.
4492 ;*
4493 ;* INPUTS: CSRA - Contains the address of the CSR.
4494 ;* TXINTC - Holds the count of the number of TX interrupts.
4495 ;* TXINTF - TX Interrupt flags.
4496 ;*
4497 ;* OUTPUTS: TXINTC - Contains the updated TX interrupt count.
4498 ;* TXINTF - TX Int flags (bit 0 set, bit 15 set if TX.ACTION clr).
4499 ;*
4500 ;* CALLING SEQUENCE: Put the address of the label TXINTR in the vector
4501 ;* location.
4502 ;*
4503 ;* COMMENTS:
4504 ;*
4505 ;* SUBORDINATE ROUTINES CALLED: none
4506 ;-- *****
4507
4508 024144 TXINTR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
024144 004537 003776 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4509 024150 013701 002340 MOV TXINTC,R1 ;GET THE TX INTERRUPT COUNT.
4510 024154 005201 INC R1 ;INCREMENT THE COUNT.
4511 024156 102001 BVC 2# ;BRANCH IF NO OVERFLOW OCCURRED.
4512 024160 005301 DEC R1 ;RESET THE COUNT TO 177777.
4513 024162 010137 002340 2# MOV R1,TXINTC ;SAVE NEW COUNT VALUE.
4514 024166 013703 002342 MOV TXINTF,R3 ;GET THE TX INTERRUPT FLAGS.
4515 024172 017702 156046 MOV @CSRA,R2 ;READ THE CSR.
4516 024176 100402 BMI 4# ;SKIP SETTING OF FLAG IF TX.ACTION IS SET.
4517 024200 052703 100000 BIS @BIT15,R3 ;SET THE TX.ACTION CLEAR FLAG.
4518 024204 052703 000001 4# BIS @BIT0,R3 ;SET THE TX INT HAS OCCURRED FLAG.
4519 024210 010337 002342 MOV R3,TXINTF ;UPDATE THE TX INTERRUPT FLAGS.
4520 024214 004736 60# PASS ;RESTORE GPRS.
024214 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4521 024216 000002 RTI

```

INTERUPT SERVICE ROUTINE - TXINTR -

```

4523
4524 ;*****
4525 ;
4526 ;           VDHA.RPT
4527 ;
4528 ;*****
4529
4530
4531
4532 .SBTTL  REPORT CODING SECTION
4533
4534 ;**
4535 ; THE REPORT CODING SECTION CONTAINS THE
4536 ; "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
4537 ;--
4538
4539 024220          BGNRPT
4540 024220
4541 024220          EXIT  RPT
4542 024220 000167
4543 024222 000000
4544
4545 024224          ENDRPT
4546 024224
4547 024224 104425

```

```

L1RPT::
        .WORD  J$JMP
        .WORD  L10017-2-.

```

```

L10017: TRAP  C1RPT

```

PROTECTION TABLE

4547
4548
4549
*
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562 024226
024226
4563
4564 024226 177777
4565 024230 177777
4566 024232 177777
4567
4568 024234
4569

```
.SBTTL PROTECTION TABLE
;*****
;
;           FVTSKL4.P11
;*****
;
; **
; THIS TABLE IS USED BY THE RUNTIME SERVICES
; TO PROTECT THE LOAD MEDIA.
; --
          BGNPROT                                LPROT::
          -1                                     ;OFFSET INTO P-TABLE FOR CSR ADDRESS
          -1                                     ;OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
          -1                                     ;OFFSET INTO P-TABLE FOR DRIVE NUMBER
          ENDPROT
```

PROTECTION TABLE

```

4584
4585
4586 ;*****
4587 ;
4588 ;           VDHA.INI
4589 ;
4590 ;*****
4591
4592
4593 .SBTTL INITIALIZE SECTION
4594 ;**
4595 ;*****
4596 ;*   This section contains the code which is performed at the beginning of
4597 ;*   each pass or after a continue command.
4598 ;*   This code performs the following actions:
4599 ;*
4600 ;*   Moves the information held in the hardware P-table into the global
4601 ;*   data area.
4602 ;*
4603 ;*****
4604 ;--
4605 024234      BGNINIT
4606           024234
4607           024234 012700 000040
4608           024240 104447
4609           024242      BCOMPLETE      NEWSTA
4610           024242 103416
4611           024244      ;SEE IF PROGRAM JUST RESTARTED, BR IF YES
4612           024244 012700 000037
4613           024250 104447
4614           024252      BCOMPLETE      NEWRES
4615           024252 103555
4616           024254      ;SEE IF THIS IS A NEW PASS, BR IF YES
4617           024254 012700 000035
4618           024260 104447
4619           024262      BCOMPLETE      NEWPAS
4620           024262 103554
4621           024264      ;SEE IF PROGRAM WAS JUST CONTINUED
4622           024264 012700 000036
4623           024270 104447
4624           024272      BNCOMPLETE     GETPRM
4625           024272 103160
4626           024274 000137 025050
4627           024300      JMP      ENDIT
4628           024300      NEWSTA:
4629           024300      BRESET
4630           024300      ;RESET THE BUS TO PREVENT ILLEGAL INTERRUPTS.
4631           024300      TRAP      C$RESET
4632
4633 ;*
4634 ; Set up for Line Time Clock interrupts.
4635 ;--
4636           024302      CLOCK      L,R1      ;GET THE CLOCK PARAMETERS.
4637           024302 012700 000114
4638           024306 104462
4639           024306      MOV      #'L,RO
4640           024306      TRAP      C$CLK

```

INITIALIZE SECTION

```

024310 010001
4625 024312 012137 002354      MOV      (R1)+,CLKCSR      ;STORE CLOCK CSR ADDRESS.
4626 024316 012137 002356      MOV      (R1)+,CLKBRL     ;STORE CLOCK BUS REQ INT LEVEL.
4627 024322 012137 002360      MOV      (R1)+,CLKVEC     ;STORE CLOCK INTERRUPT VECTOR.
4628 024326 012137 002362      MOV      (R1)+,CLKHRZ     ;STORE CLOCK FREQUENCY.
4629 024332 023727 002362 000062  CMP      CLKHRZ,#50.      ;TEST FOR 50HZ LINE FREQUENCY.
4630 024340 001004              BNE      2$              ;BRANCH IF CLOCK IS NOT 50HZ.
4631 024342 012737 000024 002374  MOV      #20.,MSTICK      ;INDICATE 20MS PER CLOCK TICK.
4632 024350 000403              BR       4$
4633 024352 012737 000021 002374 2$:  MOV      #17.,MSTICK      ;INDICATE 17 MS PER CLOCK TICK.
4634 024360 012746 000300 4$:  SETVEC  CLKVEC,#CLKINT,#PRI06 ;INITIALIZE CLOCK INTERRUPT VECTOR.
                                MOV      #PRI06,-(SP)
                                MOV      #CLKINT,-(SP)
                                MOV      CLKVEC,-(SP)
                                MOV      #3,-(SP)
                                TRAP     C$SVEC
                                ADD      #10,SP
024360 012746 000300
024364 012746 023676
024370 013746 002360
024374 012746 000003
024400 104437
024402 062706 000010
4635 024406 013700 002362      MOV      CLKHRZ,R0        ;INITIALIZE THE BREAK COUNT
4636 024412 006200              ASR      R0                ; TO CAUSE A BREAK
4637 024414 010037 002372      MOV      R0,BCOUNT        ; EVERY 1/2 SECOND.
4638 024420 106427 000240      MTPS     #PRI05           ;ALLOW CLOCK INTERRUPTS DISABLE OTHERS.
4639
4640 ;*
4641 ; Enable the Line Time Clock (LTC) checking to make sure that the CSR
4642 ; is accessible.
4643 ; First set up to catch any 004 traps which occur:
4644 ;-
4644 024424 013737 000004 002344      MOV      4,TP4VEC         ;SAVE THE EXISTING 004 TRAP VECTOR.
4645 024432 012737 024122 000004      MOV      #TP4RTN,4        ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4646
4647 ;*
4648 ; Enable LTC checking for 004 trap in case CSR is not there.
4649 ;-
4649 024440 005037 002346      CLR      TP4FLG           ;CLEAR THE 004 TRAP FLAG.
4650 024444 012737 000100 002350      MOV      #BIT6,WORD1      ;SET UP TO SET BIT6 OF THE LTC CSR.
4651 024452 012700 002350      MOV      #WORD1,R0        ;SET UP WORD1 AS THE CKTRAP MOVE SOURCE.
4652 024456 013701 002354      MOV      CLKCSR,R1        ;SET UP LTC CSR AS DESTINATION FOR CKTRAP MOVE.
4653 024462 004737 020126      JSR      PC,CKTRAP        ;MOVE AND CHECK FOR TRAP.
4654 024466 013737 002344 000004      MOV      TP4VEC,4         ;RESTORE THE NORMAL 004 TRAP VECTOR.
4655 024474 103403              BCS      6$              ;IF NO TRAP, LTC IS THERE SO CONTINUE.
4656 024476 005037 002362      CLR      CLKHRZ           ;CLEAR LTC FREQUENCY WORD TO INDICATE NO LTC.
4657 024502 000402              BR       8$              ;BYPASS THE FOLLOWING CALIBRATION PROCEDURES.
4658
4659 ;*
4660 ; Calibrate the DELAY routine milli-second delay count value.
4661 ;-
4661 024504 004737 017702 6$:  JSR      PC,CALMSL
4662
4663 ;*
4664 ; Check for Memmory Management present on this machine.
4665 ; If MEM MGT is present, disable it.
4666 ;-
4666 024510 013737 000004 002344 8$:  MOV      4,TP4VEC         ;SAVE THE EXISTING 004 TRAP VECTOR.
4667 024516 012737 024122 000004      MOV      #TP4RTN,4        ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4668 024524 005037 002346      CLR      TP4FLG           ;CLEAR THE 004 TRAP FLAG.
4669 024530 005037 002350      CLR      WORD1            ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4670 024534 012700 002350      MOV      #WORD1,R0        ;SELECT CLEARED WORD AS CKTRAP RTN SOURCE.
4671 024540 013701 002400      MOV      MMSRO,R1        ;SELECT MEM MGT SRO REGISTER AS DESTINATION.
4672 024544 005037 002402      CLR      MMPRES           ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
4673 024550 005037 002404      CLR      MMENAB          ;INDICATE MEM MGT IS NOT ENABLED.
4674 024554 004737 020126      JSR      PC,CKTRAP        ;CLEAR THE MEM MGT SRO REG AND CHECK FOR TRAP.

```


INITIALIZE SECTION

```

4675 024560 013737 002344 000004      MOV    TP4VEC,4      ;RESTORE THE NORMAL 004 TRAP VECTOR.
4676 024566 103003                    BCC    10$          ;SKIP INDICATING MEM MGT PRESENT IF IT ISN'T.
4677 024570 012737 000001 002402      MOV    #1,MMPRES    ;INDICATE THAT MEM MGT IS PRESENT.
4678 024576 005037 002332 10$:   CLR    PASCNT       ;CLR COUNTER USED IN REPORTING ROM VERSION #.
4679 024602 000137 024614            JMP    NEWPAS       ;SKIP AROUND THE BUS RESET, IT'S BEEN DONE.
4680
4681 024606                    NEWRES: BRESET      ;RESET THE BUS TO PREVENT ILLEGAL INTERRUPTS.
      024606 104433
4682 024610 005037 002332            CLR    PASCNT       ;CLR COUNTER USED IN REPORTING ROM VERSION #.
4683 024614                    NEWPAS:
4684 024614 012737 177777 002240      MOV    #-1,UNITN    ;RESET LOGICAL DEVICE TO -1
4685
4686      ;+
4687      ; Increment the pass counter, correct for any overflow.
4688      ; This counter is used in the Rom version test.
4689      ;-
4689 024622 005237 002332            INC    PASCNT       ;INCREMENT THE PASS COUNTER.
4690 024626 001002                    BNE    GETPRM       ;BRANCH IF WE HAVE NOT YET! OVERFLOWED.
4691 024630 005337 002332            DEC    PASCNT       ;SET PASS COUNT TO 177777 OCTAL.
4692
4693      ; GET THE HARDWARE PARAMETERS FOR THIS UNIT.
4694 024634                    GETPRM:
4695 024634 005237 002240            INC    UNITN        ;INCREMENT LOGICAL DEVICE NUMBER
4696 024640 023737 002240 002012      CMP    UNITN,L$UNIT ;SEE IF MAXIMUM UNIT NO. EXCEEDED
4697 024646 002362                    BGE    NEWPAS       ;BR IF YES
4698
4699 024650                    GPWARD UNITN,R1      ;GET P-TABLE POINTER INTO R1
      024650 013700 002240
      024654 104442
      024656 010001
4700 024660                    BCOMPLETE 30$      ;BR IF DEVICE AVAILABLE
      024660 103401
4701 024662 000764                    BR     GETPRM       ;SKIP THIS DEVICE
4702
4703
4704
4705 024664 012137 002244 30$:   ;***** HARDWARE PARAMETER MOVING CODE *****
      MOV    (R1)+,CSRA ;STORE DHV11-M CSR ADDRESS IN DEV.REG.ADDRESS TABLE
4706 024670 012102                    MOV    (R1)+,R2    ;GET THE RX INTERRUPT VECTOR ADDRESS.
4707 024672 010237 002232            MOV    R2,RXVECA   ;STORE RX INT VECTOR ADDRESS.
4708 024676 062702 000004            ADD    #4,R2       ;CALCULATE TX INTERRUPT VECTOR ADDRESS.
4709 024702 010237 002234            MOV    R2,TXVECA   ;STORE TX INT VECTOR ADDRESS.
4710 024706 012137 002236            MOV    (R1)+,ACTLNS ;STORE DHV11-M ACTIVE LINE BIT MAP
4711 024712 012702 000377            MOV    #MAPLNS,R2  ;GET THE BIT MAP FOR ALL LINES.
4712 024716 005102                    COM    R2          ;GET A BIT MAP OF NON-EXISTANT LINES.
4713 024720 040237 002236            BIC    R2,ACTLNS   ;CLEAR NON-EXISTANT LINES FROM ACTLNS.
4714 024724 112137 002242            MOVB   (R1)+,BRLEVL ;STORE DHV11-M INTERUPT BUS REQUEST LEVEL
4715
4716      ;+
4717      ; CALCULATE DEVICE REGISTER ADDRESSES,AND PUT THEM IN THE
4718      ; DEVICE REGISTER ADDRESS TABLE.
4719      ;-
4719 024730 013701 002244            MOV    CSRA,R1     ;COPY CSR ADDRESS
4720 024734 005201                    INC    R1          ;INCREMENT CSR ADDRESS
4721 024736 005201                    INC    R1          ; COPY BY 2.
4722 024740 012703 000007            MOV    #7,R3       ;SET UP REGISTER COUNT
4723 024744 012702 002246            MOV    #RBUFA,R2   ;GET LOCATION WHERE RBUF ADDRESS GOES IN TABLE
4724 024750 010122 12$:   MOV    R1,(R2)+    ;STORE REGISTER ADDRESS IN TABLE
4725 024752 005201                    INC    R1          ;INCREMENT REGISTER ADDRESS
4726 024754 005201                    INC    R1          ; BY 2, FOR THE NEXT DEVICE REGISTER.

```

INITIALIZE SECTION

```

4727 024756 005303          DEC    R3          ;DECREMENT REGISTER COUNT
4728 024760 001373          BNE    12$        ;LOOP IF NOT DONE
4729
4730
4731          ;+
4731          ; Initialise the BMP code queue.
4732          ;-
4733 024762 012700 002526    MOV    #BMPQCB,R0  ;GET THE START ADDRESS OF THE QUEUE.
4734 024766 012701 002726    MOV    #BMPQCE,R1  ;GET THE END ADDRESS OF THE QUEUE.
4735 024772 010037 002524    MOV    R0,BMPQCP   ;SET THE POINTER TO THE START OF THE QUEUE.
4736 024776 005020          14$:  CLR    (R0)+       ;CLEAR OUT THE CONTENTS OF THE QUEUE.
4737 025000 020001          CMP    R0,R1       ;CHECK IF END OF QUEUE HAS BEEN REACHED.
4738 025002 103775          BLO    14$         ;LOOP IF NOT ALL DONE.
4739
4740          ;+
4740          ; Report the Unit number if the software P-table question was answered YES,
4741          ; and the maximum unit number is greater than 1.
4742          ;-
4743 025004 032737 000020 002226  BIT    #BIT4,OPTION ;CHECK IF THE QUESTION WAS ANSWERED YES.
4744 025012 001416          BEQ    16$         ;SKIP REPORTING UNIT NUMBER IF IT IS DISABLED.
4745 025014 023727 002012 000001  CMP    L$UNIT,#1   ;CHECK MAXIMUM NUMBER OF UNITS SELECTED.
4746 025022 003412          BLE    16$         ;DO NOT REPORT UNIT NUMBER IF MAX NUMBER < 1.
4747 025024          PRINTF #MFUNIT,UNITN ;REPORT UNIT NUMBER.
          MOV    UNITN,-(SP)
          MOV    #MFUNIT,-(SP)
          MOV    #2,-(SP)
          MOV    SP,R0
          TRAP  C$PNTF
          ADD    #6,SP
4748 025050          16$:
4749
4750 025050 005037 002352  ENDIT: CLR    CTRLCF      ;CLR THE CTRL-C TEST ABORT FLAG.
4751          ;+
4751          ; Set the processor priority to allow LTC interrupts but not others.
4752          ;-
4753 025054 106427 000240          MTPS   #PRI05      ;SET PROCESSOR PRIORITY TO 5.
4754
4755          ;+
4755          ; Enable Line Time Clock if one is available.
4756          ;-
4757 025060 005737 002362          TST    CLKHRZ      ;CHECK FOR A LTC BEING PRESENT.
4758 025064 001403          BEQ    18$         ;LTC PRESENT? NO, SKIP LTC ENABLE.
4759 025066 012777 000100 155260  MOV    #BIT6,@CLKCSR ;YES, ENABLE THE LTC.
4760
4761 025074          18$:
4762
4763 025074          ENDINIT
          L10021:
          TRAP  C$INIT
4764
4765          000000          TNUM == 0          ;INITIALIZE THE ASSEMBLER TEST NUMBER VARIABLE.

```

INITIALIZE SECTION

```

4768 ;*****
4769 ;
4770 ;           VDHA.ATD
4771 ;
4772 ;*****
4774
4775
4776
4777
4778
4779

```

.SBTTL AUTODROP SECTION

```

4780 ;++
4781 ; THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
4782 ; THE "ADR" FLAG WAS SET. THE UNIT(S) UNDER TEST ARE CHECKED TO
4783 ; SEE IF THEY WILL RESPOND. THOSE THAT DON'T ARE IMMEDIATELY
4784 ; DROPPED FROM TESTING.
4785 ;--

```

```

4786 025076          BGNAUTO
4787 025076
4794
4795 025076          ENDAUTO

```

L\$AUTO::

```

4795 025076          ENDAUTO
025076 104461
L10022: TRAP C$AUTO

```

AUTODROP SECTION

```

4797
4798 ;*****
4799 ;
4800 ;           VDHA.CUC
4801 ;
4802 ;*****
4803
4804
4805
4806 .SBTTL  CLEANUP CODING SECTION
4807
4808 ;++
4809 ; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
4810 ; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
4811 ;--
4812
4813 025100          BGNCLN
      025100
4814
4823 025100 005737 002352
4824 025104 001403
4825 025106
      025106 104433
4826 025110 106427 000240
4827 025114
      2$: TST  CTRLCF
      BEQ  2$ ;DID WE GET HERE BY CTRL-C FROM TEST?
4828 025114 005737 002362
      BRESET ;CTRL-C FROM TEST? NO, SKIP BUS RESET.
4829 025120 001402
      MTPS  #PRI05 ;YES, CLR ANY DMAS OR OUTSTANDING INTERRUPTS.
4830 025122 005077 155226
      2$:   TRAP  C$RESET
      CLR  @CLKCSR ;DISABLE DEVICE INTERRUPTS, ALLOW LTC INTS.
4831 025126
      3$:   EXIT  CLN
4832 025126
      025126 104432
      025130 000002
      TRAP  C$EXIT
      .WORD L10023-.
4833
4845
4846 .EVEN
4847
4848 025132
      025132
      025132 104412
      ENDCLN
      L10023: TRAP  C$CLEAN
    
```

CLEANUP CODING SECTION

```

4850
4851 ;*****
4852 ;
4853 ;           VDHA.DRF
4854 ;
4855 ;*****
4856
4857
4858
4859 .SBTTL DROP UNIT SECTION
4860
4861 ;**
4862 ; THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
4863 ; TO NO LONGER BE TESTED.
4864 ;--
4865
4866 025134          BGNDU
4867 025134
4868
4869 ;*****
4870 ;           INSERT DROP CODE HERE. THIS CODE WILL BE EXECUTED AFTER
4871 ;           A "DROP" COMMAND OR A "DODU" MACRO EXECUTION. THE PURPOSE
4872 ;           OF THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A
4873 ;           UNIT HAS BEEN DROPPED. THIS SECTION IS OPTIONAL.
4874 ;*****
4875 025134          PRINTF #DROP,R0          ;REPORT UNIT THAT HAS BEEN DROPPED.
4876 025134 010046          MOV          R0,-(SP)
4877 025136 012746 025160          MOV          #DROP,-(SP)
4878 025142 012746 000002          MOV          #2,-(SP)
4879 025146 010600          MOV          SP,R0
4880 025150 104417          TRAP         C#PNTF
4881 025152 062706 000006          ADD          #6,SP
4882 025156 000427          BR          EDROP          ;BRANCH AROUND THE MESSAGE.
4883
4884 025160          045      101      040 DROP: .ASCIZ/*A UNIT#D6#A DROPPED FROM FURTHER TESTING.#N/
4885 025163          125      116      111
4886 025166          124      045      104
4887 025171          066      045      101
4888 025174          040      104      122
4889 025177          117      120      120
4890 025202          105      104      040
4891 025205          106      122      117
4892 025210          115      040      106
4893 025213          125      122      124
4894 025216          110      105      122
4895 025221          040      124      105
4896 025224          123      124      111
4897 025227          116      107      056
4898 025232          045      116      000
4899
4900 025236          .EVEN
4901 EDROP:
4902 025236          EXIT      DU
4903 025236 000167          .WORD      J$JMP
4904 025240 000000          .WORD      L10024-2-.
4905
4906 4882
4907 4883

```

DROP UNIT SECTION

4884
4885 025242
025242
025242 104453

ENDDU

L10024:
TRAP C\$DU

HARDWARE TEST

- ADRA -

```

4921 .SBTTL HARDWARE TEST - ADRA -
4922 ;++
4923 ;*****
4924 ;*
4925 ;* - REGISTER ADDRESS TEST -
4926 ;*
4927 ;* This test verifies that the Q-bus can read and write to the DHV11
4928 ;* device registers. If the DHV11 does not respond to the access
4929 ;* attempts (If the DHV11 is at the wrong address, for example) the
4930 ;* 004 bus time-out trap is detected by this routine and an error
4931 ;* is reported.
4932 ;*
4933 ;*****
4934 ;--
4935 025252 BGNTST
      025252
4936 000001 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
4937 025252 012737 000001 002326 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (1)
4938 025260 012737 177777 002352 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
4939 ;+
4940 ; Set up to catch any 004 traps which occur:
4941 ;-
4942 025266 013737 000004 002344 MOV 4,TP4VEC ;SAVE THE EXISTING 004 TRAP VECTOR.
4943 025274 012737 024122 000004 MOV #TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4944 025302 005005 CLR R5 ;CLEAR THE ERROR FLAGS.
4945
4946 ;+
4947 ; Set up for the initial iteration of the test loop:
4948 ;-
4949 025304 005004 CLR R4 ;CLEAR THE LINE COUNTER.
4950
4951 ;+
4952 ; Here begins the loop to test the registers for a line.
4953 ; First test the CSR and set the IND.ADR.REG (I.A.R) field.
4954 ;-
4955 025306 005037 002346 2$: CLR TP4FLG ;CLEAR THE 004 TRAP FLAG.
4956 025312 013700 002244 MOV CSRA,R0 ;SET UP CSR AS THE CKTRAP MOVE SOURCE.
4957 025316 012701 025532 MOV #52$,R1 ;SET UP DESTINATION LOCATION FOR CKTRAP MOVE.
4958 025322 004737 020126 JSR PC,CKTRAP ;MOVE AND CHECK FOR TRAP.
4959 025326 103402 BCS 4$ ;IF NO TRAP, BYPASS ERROR.
4960 025330 052705 100001 BIS #100001,R5 ;SET FATAL READ ERROR FLAGS.
4961 025334 042737 000017 025532 4$: BIC #17,52$ ;CLEAR THE I.A.R FIELD OF THE CSR DATA.
4962 025342 050437 025532 BIS R4,52$ ;OR IN THE LINE COUNTER TO THE I.A.R FIELD.
4963 025346 010100 MOV R1,R0 ;USE OLD DESTINATION FOR SOURCE OF CKTRAP MOVE.
4964 025350 013701 002244 MOV CSRA,R1 ;SET UP CSR AS THE CKTRAP MOVE DESTINATION.
4965 025354 004737 020126 JSR PC,CKTRAP ;MOVE AND CHECK FOR TRAP.
4966 025360 103403 BCS 6$ ;IF NO TRAP, BYPASS ERROR.
4967 025362 052705 100002 BIS #100002,R5 ;SET FATAL WRITE ERROR FLAGS.
4968 025366 000440 BR 40$ ;EXIT AND REPORT FATAL ERROR.
4969 ;+
4970 ; Now, we test each register for this line.
4971 ;-
4972 025370 012702 000010 6$: MOV #10,R2 ;INIT REGISTER COUNTER TO 8.
4973 025374 013737 002244 025530 MOV CSRA,50$ ;INITIALIZE THE REGISTER POINTER.
4974 025402 012700 025530 8$: MOV #50$,R0 ;SET UP REGISTER AS THE SOURCE FOR CKTRAP MOVE.
4975 025406 012701 025532 MOV #52$,R1 ;SET UP LOCAL STORAGE AS THE DES FOR CKTRAP.
4976 025412 004737 020126 JSR PC,CKTRAP ;PERFORM THE MOVE, CHECK FOR TRAP.

```


HARDWARE TEST

- ADRA -

```

4977 025416 103402          BCS      10$          ;IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.
4978 025420 052705 100001    BIS      #100001,R5    ;SET FATAL READ ERROR FLAGS.
4979 025424 010100          10$:    MOV      R1,R0      ;USE OLD DEST AS SRC FOR CKTRAP MOVE.
4980 025426 012701 025530    MOV      #50$,R1      ;SET UP REGISTER AS THE DEST FOR CKTRAP MOVE.
4981 025432 004737 020126    JSR      PC,CKTRAP    ;PERFORM THE MOVE, CHECK FOR TRAP.
4982 025436 103402          BCS      12$          ;IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.
4983 025440 052705 100002    BIS      #100002,R5    ;SET FATAL WRITE ERROR FLAGS.
4984 025444 005237 025530    12$:    INC      50$      ;INCREMENT THE REGISTER
4985 025450 005237 025530    INC      50$          ; POINTER BY 2.
4986 025454 005302          DEC      R2           ;COUNT THE REGISTER.
4987 025456 001351          BNE      8$           ;LOOP TO TEST THE NEXT REGISTER ADDRESS.
4988
4989
4990
4991      ;+
      ; Now we set up to test the next line, or to exit if we are done.
4992      ;-
4992 025460 005204          INC      R4           ;INCREMENT THE LINE COUNTER.
4993 025462 020427 000010    CMP      R4,#NUMLNS   ;COMPARE LINE COUNTER AGAINST NUMBER OF LINES.
4994 025466 002707          BLT      2$           ;LOOP TO TEST THE NEXT LINE IF WE'RE NOT DONE.
4995
4996
4997      ;+
4997      ; Done checking device register addresses.
4998      ; Report any errors and exit.
4999      ;-
5000 025470 013737 002344 000004 40$:    MOV      TP4VEC,4      ;RESTORE THE NORMAL 004 TRAP VECTOR.
5001 025476 005705          TST      R5           ;CHECK THE ERROR FLAGS.
5002 025500 100015          BPL      60$          ;EXIT ROUTINE IF NO ERRORS.
5003      ; REPORT "DEVICE REGISTER ACCESS ERRORS"
5004 025502          ERRDF  101,EM0103,ER0101; >>>> ERROR #101 <<<<<.
5004 025502 104455          TRAP    C#ERDF
5004 025504 000145          .WORD  101
5004 025506 006067          .WORD  EM0103
5004 025510 016070          .WORD  ER0101
5005
5006 025512          DODU   UNITN          ;DROP THIS UNIT FROM FUTHER TESTING.
5006 025512 013700 002240    MOV      UNITN,R0     ;
5006 025516 104451          TRAP    C#DODU
5007 025520 005037 002352    CLR      CTRLCF       ;INDICATE NO CTRL-C ABORT FROM TEST.
5008 025524          DOCLN          ;ABORT THIS SUB PASS.
5008 025524 104444          TRAP    C#DCLN
5009 025526 000402          BR      60$          ;
5010
5011      ;+
5011      ; Local storage.
5012      ;-
5013 025530 000000          50$:    .WORD  0           ;STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.
5014 025532 000000          52$:    .WORD  0           ;STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.
5015 025534 005037 002352    60$:    CLR      CTRLCF       ;INDICATE THAT WE ARE NOT WITHIN A TEST.
5016 025540          ENDTST
5016 025540 104401          L10026: TRAP    C#ETST

```

HARDWARE TEST

- MRSTA -

```

5018 .SBTTL HARDWARE TEST - MRSTA -
5019 ;** *****
5020 ;* - Master Reset With Selftest Test -
5021 ;* This test verifies that the Master Reset bit will clear after a device
5022 ;* reset and the performance of the DUT ROM based selftest.
5023 ;*
5024 ;-- *****
5025 025542 BGNTST
    025542
5026 000002 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5027 025542 012737 000002 002326 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (2)
5028 025550 012737 177777 002352 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
5029 025556 012737 000001 003766 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5030 025264 012737 006125 003772 MOV #EM0201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5031 025572 012737 016406 003774 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5032
5033 ;+
5034 ; Wait up to 5 seconds for the DUT Master Reset bit to clear.
5035 025600 012701 011610 ;- MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5036 025604 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5037 025610 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5038 025612 013704 002244 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5039 025616 004737 020340 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5040 025622 103410 BCS 2# ;SKIP TO RESET DUT IF MR CLEAR.
5041
5042 ;+
5043 ; DUT Master Reset bit did not go clear. Device may be stuck in some
5044 ; odd state. Try to reset device with a bus reset.
5045 025624 BRESET ;NO, TRY TO JOG DEVICE WITH BUS RESET.
    025624 104433 TRAP C#RESET
5046 025626 004737 022330 JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST.
5047 025632 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5048 025636 004737 020340 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5049 025642 103016 BCC 4# ;GO REPORT ERROR IF MR BIT DID NOT CLEAR.
5050
5051 ;+
5052 ; Set the Master Reset bit and verify that it clears within the proper time.
5053 025644 012701 011610 2#: MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5054 025650 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5055 025652 004737 020340 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5056 025656 103010 BCC 4# ;GO REPORT ERROR IF MR BIT DID NOT CLEAR.
5057 025660 012702 011610 MOV #5000.,R2
5058 025664 160102 SUB R1,R2 ;CALCULATE # OF MS FOR MR TO CLEAR.
5059 025666 001413 BEQ 6# ;GO REPORT ERROR IF MR CLEAR IMMEDIATELY.
5060 025670 020227 000764 CMP R2,#500.
5061 025674 002417 BLT 8# ;GO REPORT ERROR IF MR CLEAR IN < 1/2 SECOND.
5062 025676 000424 BR 60# ;EXIT THE TEST WITHOUT ERROR.
5063
5064 ;+
5065 ; Error reports:
5066 ;-
5067 025700 012737 000311 003770 4#: ;Report MR bit would not clear after a DUT reset.
5068 025706 012701 006173 MOV #201.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5069 025712 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
    025712 104460 ERROR ;REPORT ERROR. >>>> ERROR #201 <<<<
5070 025714 000415 BR 60# TRAP C#ERROR
5071

```

HARDWARE TEST

- MRSTA -

```

5072
5073 025716 012737 000312 003770 6$: ;Report MR bit clear immediately after DUT reset.
5074 025724 012701 006346          MOV #202.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5075 025730          MOV #EM0203,R1 ;SELECT ERROR MESSAGE.
025730 104460          ERROR ;REPORT ERROR. >>>> ERROR #202 <<<<<
5076 025732 000406          BR 60$ ;EXIT THE TEST. TRAP C#ERROR
5077
5078
5079 025734 012737 000313 003770 8$: ;Report MR clear within 1/2 second of DUT reset.
5080 025742 012701 006511          MOV #203.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5081 025746          MOV #EM0204,R1 ;SELECT ERROR MESSAGE.
025746 104460          ERROR ;REPORT ERROR. >>>> ERROR #203 <<<<<
5082 TRAP C#ERROR
5083 025750 005037 002352          60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5084
5085 025754          ENDTST
025754          L10027:
025754 104401          TRAP C#ETST
    
```

HARDWARE TEST - MRSSTA -

```

5087
5088
5089
5090
5091
5092
5093
5094 025756
      025756
5095      000003
5096 025756 012737 000003 002326
5097 025764 012737 177777 002352
5098 025772 012737 000001 003766
5099 026000 012737 006670 003772
5100 026006 012737 016406 007774
5101
5102
5103
5104 026014 012701 011610
5105 026020 012702 000040
5106 026024 005003
5107 026026 013704 002244
5108 026032 004737 020340
5109 026036 103410
5110
5111
5112
5113
5114 026040
      026040 104433
5115 026042 004737 022330
5116 026046 012701 011610
5117 026052 004737 020340
5118 026056 103024
5119
5120
5121
5122
5123 026060 012701 000310
5124 026064 010214
5125 026066 004737 022330
5126 026072 004737 020340
5127 026076 103007
5128 026100 012702 000310
5129 026104 160102
5130 026106 020227 000012
5131 026112 002415
5132 026114 000431
5133
5134
5135
5136 026116 012701 011300
5137 026122 004737 020340
5138 026126 103416
5139
5140
5141
    
```

```

.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 @ER0201,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 -

```

5142
5143 026130 012737 000455 003770 6$: ;Report MR bit would not clear after a DUT reset.
      MOV #0301.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5144 026136 012701 006173      MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5145 026142      ERROR ;REPORT ERROR. >>>> ERROR #0301 <<<<<
      026142 104460      TRAP C$ERROR
5146 026144 000415      BR 60$ ;EXIT THE TEST.
5147
5148 ;Report MR bit clear within 10 ms after DUT reset.
5149 026146 012737 000456 003770 8$: MOV #0302.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5150 026154 012701 006733      MOV #EM0302,R1 ;SELECT ERROR MESSAGE.
5151 026160      ERROR ;REPORT ERROR. >>>> ERROR #0302 <<<<<
      026160 104460      TRAP C$ERROR
5152 026162 000406      BR 60$ ;EXIT THE TEST.
5153
5154 ;Report MR cleared between 1/5 second and 5 seconds of DUT reset.
5155 026164 012737 000457 003770 10$: MOV #0303.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5156 026172 012701 007073      MOV #EM0303,R1 ;SELECT ERROR MESSAGE.
5157 026176      ERROR ;REPORT ERROR. >>>> ERROR #0303 <<<<<
      026176 104460      TRAP C$ERROR
5158
5159 026200 005037 002352 60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5160
5161 026204      ENDTST
      026204
      026204 104401      L10030: TRAP C$ETST
    
```

HARDWARE TEST

- RXCHRA -

```

5163 .SBTTL HARDWARE TEST - RXCHRA -
5164 ;** *****
5165 ;* - RBUF Register RX Character Field Test -
5166 ;* This test verifies that the RX Character Field of the DUT RBUF register
5167 ;* appears to be functioning correctly. This test uses the codes which
5168 ;* should be in the FIFO after a board reset and skip selftest sequence.
5169 ;*
5170 ;-- *****
5171 026206 BGNTST
    026206
5172          000004          TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5173 026206 012737 000004 002326          MOV    #TNUM,TSTNUM          ;SET UP THE TEST NUMBER. (4)
5174 026214 012737 177777 002352          MOV    #-1,CTRLCF          ;INDICATE THAT WE ARE WITHIN A TEST.
5175 026222 012737 000001 003766          MOV    #1,ERRTYP          ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5176 026230 012737 007252 003772          MOV    #EM0401,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5177 026236 012737 016406 003774          MOV    #ER0201,ERRBLK      ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5178
5179 ;+
5180 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
5181 ; and wait up to 5 seconds for the MR bit to clear.
5182 026244 012701 011610          MOV    #5000.,R1          ;TIME-OUT VALUE IS 5.0 SECONDS.
5183 026250 012702 000040          MOV    #BIT05,R2          ;WAITING FOR MASTER RESET BIT.
5184 026254 005003          CLR    R3          ;WAITING FOR BIT TO CLEAR.
5185 026256 013704 002244          MOV    CSRA,R4          ;BIT IS IN THE DUT'S CSR.
5186 026262 010214          MOV    R2,(R4)          ;SET THE DUT MASTER RESET BIT.
5187 026264 004737 022330          JSR    PC,SKPSTS          ;SKIP THE SELFTEST.
5188 026270 004737 020340          JSR    PC,MSLGET          ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5189 026274 103015          BCC    4$          ;GO REPORT ERROR IF MR DID NOT CLEAR.
5190
5191 ;+
5192 ; Read 6 characters from the DUT and verify that they are valid selftest
5193 ; codes.
5194 026276 012400          MOV    (R4)+,R0          ;INCREMENT POINTER TO POINT TO DUT RBUF REGSTR.
5195 026300 012701 000006          MOV    #6,R1          ;INITIALIZE THE LOOP COUNTER.
5196 026304 011402          2$: MOV    (R4),R2          ;READ A CHARACTER FROM THE DUT RBUF REGISTER.
5197 026306 010200          MOV    R2,R0
5198 026310 042700 177476          BIC    #177476,R0          ;REMOVE ALL BUT BITS SPECIFIC TO SELFTEST CODE.
5199 026314 020027 000201          CMP    R0,#201          ;CHECK THAT BITS 0,6, AND 7 ARE CORRECT.
5200 026320 001012          BNE    6$          ;GO REPORT ERROR IF CODE IS NOT SELFTEST CODE.
5201 026322 005301          DEC    R1          ;COUNT THIS LOOP ITERATION.
5202 026324 001367          BNE    2$          ;LOOP IF NOT ALL LINES DONE.
5203 026326 000415          BR     60$          ;EXIT TEST, NO ERROR FOUND.
5204
5205 ;+
5206 ; Error reports:
5207 ;--
5208 ;Report MR bit would not clear after a DUT reset.
5209 026330 012737 000621 003770 4$: MOV    #0401.,ERRNBR      ;SET THE ERROR NUMBER IN ERROR TABLE.
5210 026336 012701 006173          MOV    #EM0202,R1          ;SELECT ERROR MESSAGE.
5211 026342          ERROR          ;REPORT ERROR. >>>> ERROR #0401 <<<<<
    026342 104460          TRAP    C$ERROR
5212 026344 000406          BR     60$          ;EXIT THE TEST.
5213
5214 ;Report improper code found in DUT RBUF after reset (skip selftest).
5215 026346 012737 000622 003770 6$: MOV    #0402.,ERRNBR      ;SET THE ERROR NUMBER IN ERROR TABLE.
5216 026354 012701 007321          MOV    #EM0402,R1          ;SELECT ERROR MESSAGE.
5217 026360          ERROR          ;REPORT ERROR. >>>> ERROR #0402 <<<<<

```

HARDWARE TEST - RXCHRA -

5218 026360 104460

TRAP C#ERROR

5219 026362 005037 002352

60\$: CLR CTRLCF

;INDICATE THAT WE COMPLETED THE TEST.

5220

5221 026366

ENDTST

L10031:

TRAP C#ETST

026366 104401

HARDWARE TEST - RXFFDA -

5277

5278 026540 005037 002352

5279

5280 026544

026544

026544 104401

60\$: CLR CTRLCF

;INDICATE THAT WE COMPLETED THE TEST.

ENDTST

L10032:

TRAP C\$ETST

HARDWARE TEST - RDAA -

```

5282 .SBTTL HARDWARE TEST - RDAA -
5283 ;+ *****
5284 ;* - CSR RX Data Available Bit Test -
5285 ;* This test verifies that the DUT CSR RX Data Available Bit is set by the
5286 ;* inclusion of the selftest codes in the DUT FIFO and that the bit clears
5287 ;* after the FIFO has been emptied.
5288 ;*
5289 ;- *****
5290 026546 BGNTST
    026546
5291 000006 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5292 026546 012737 000006 002326 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (6)
5293 026554 012737 177777 002352 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5294 026562 012737 000001 003766 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5295 026570 012737 010073 003772 MOV #EM0601,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5296 026576 012737 016406 003774 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5297
5298 ;+
5299 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
5300 ; and wait up to 5 seconds for the MR bit to clear.
5301 026604 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5302 026610 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5303 026614 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5304 026616 013704 002244 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5305 026622 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5306 026624 004737 022330 JSR PC,SKPSTS ;SKIP THE SELFTEST.
5307 026630 004737 020340 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5308 026634 103016 BCC 4$ ;GO REPORT ERROR IF MR DID NOT CLEAR.
5309
5310 ;+
5311 ; Check that the RX Data Available bit is set.
5312 026636 032714 000200 BIT #BIT7,(R4) ;TEST THE DUT RX.DATA.AVAIL BIT.
5313 026642 001422 BEQ 6$ ;GO REPORT ERROR IF BIT IS NOT SET.
5314
5315 ;+
5316 ; Read characters from the DUT RX FIFO and wait for RX.DATA.AVAIL to go clear.
5317 026644 012705 001130 MOV #600.,R5 ;ALLOW READING 600 CHARS BEFORE ERROR.
5318 026650 010403 MOV R4,R3
5319 026652 012300 MOV (R3)+,R0 ;CALCULATE THE RBUF ADDRESS.
5320 026654 011300 MOV (R3),R0 ;READ A CHARACTER FROM THE RX FIFO.
5321 026656 032714 000200 BIT #BIT7,(R4) ;TEST THE DUT RX.DATA.AVAIL BIT.
5322 026662 001427 BEQ 60$ ;EXIT TEST WITHOUT ERROR IF RX.DATA.AVAIL CLR.
5323 026664 005305 DEC R5 ;COUNT THE CHARACTER JUST READ.
5324 026666 001372 BNE 2$ ;LOOP IF NOT TOO MANY CHARS READ FROM FIFO.
5325 026670 000416 BR 8$ ;GO REPORT ERROR IF RX.DATA.AVAIL WOULDN'T CLR.
5326
5327 ;+
5328 ; Error reports:
5329 ;-
5330 ;Report MR bit would not clear after a DUT reset.
5331 026672 012737 001131 003770 4$: MOV #0601.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5332 026700 012701 006173 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5333 026704 ERROR ;REPORT ERROR. >>>> ERROR #0601 <<<<<
    026704 104460 TRAP C$ERROR
5334 026706 000415 BR 60$ ;EXIT THE TEST.
5335
5336 ;Report that RX.DATA.AVAIL bit was not set after a reset completion.

```

HARDWARE TEST

- RDAA -

```

5337 026710 012737 001132 003770 6$:   MOV   #0602.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5338 026716 012701 010127                MOV   #EM0602,R1   ;SELECT ERROR MESSAGE.
5339 026722                ERROR                ;REPORT ERROR.          >>>> ERROR #0602 <<<<<
                                104460                                TRAP   C$ERROR
5340 026724 000406                BR    60$          ;EXIT THE TEST.
5341
5342                ;Report that RX.DATA.AVAIL bit could not be cleared by purging FIFO.
5343 026726 012737 001133 003770 8$:   MOV   #0603.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5344 026734 012701 010307                MOV   #EM0603,R1   ;SELECT ERROR MESSAGE.
5345 026740                ERROR                ;REPORT ERROR.          >>>> ERROR #0603 <<<<<
                                104460                                TRAP   C$ERROR
5346
5347 026742 005037 002352 60$:   CLR   CTRLCF      ;INDICATE THAT WE COMPLETED THE TEST.
5348
5349 026746                ENDTST
                                104401                                L10033:
                                TRAP   C$ETST
    
```

HARDWARE TEST

- RDVA -

```

5351 .SBTTL HARDWARE TEST - RDVA -
5352 ;++ *****
5353 ;* - RBUF RX Data Valid Bit Test -
5354 ;* This test verifies that the DUT RBUF RX Data Valid Bit is set by the
5355 ;* inclusion of the selftest codes in the DUT FIFO and that the bit clears
5356 ;* after the FIFO has been emptied.
5357 ;*
5358 ;-- *****
5359 026750 BGNTST
026750
5360 000007 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5361 026750 012737 000007 002326 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (7)
5362 026756 012737 177777 002352 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
5363 026764 012737 000001 003766 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5364 026772 012737 010472 003772 MOV #EM0701,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5365 027000 012737 016406 003774 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5366 ;+
5367 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
5368 ; and wait up to 5 seconds for the MR bit to clear.
5369 ;-
5370 027006 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
5371 027012 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
5372 027016 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
5373 027020 013704 002244 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
5374 027024 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
5375 027026 004737 022330 JSR PC,SKPSTS ;SKIP THE SELFTEST.
5376 027032 004737 020340 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5377 027036 103012 BCC 4$ ;GO REPORT ERROR IF MR DID NOT CLEAR.
5378 ;+
5379 ; Check that the RX Data Valid bit is set.
5380 ;-
5381 027040 012400 MOV (R4)+,R0 ;INCREMENT POINTER TO PNT TO DUT RBUF REG.
5382 027042 005714 TST (R4) ;TEST THE DUT RX.DATA.VALID BIT.
5383 027044 100016 BPL 6$ ;GO REPORT ERROR IF BIT IS NOT SET.
5384 ;+
5385 ; Read characters from the DUT RX FIFO and wait for RX.DATA.VALID to go clear.
5386 ;-
5387 027046 012705 001130 MOV #600.,R5 ;ALLOW READING 600 CHARS BEFORE ERROR.
5388 027052 011400 2$: MOV (R4),R0 ;READ A CHARACTER FROM THE RX FIFO.
5389 027054 100027 BPL 60$ ;EXIT TEST WITHOUT ERROR IF BIT IS CLEAR.
5390 027056 005305 DEC R5 ;COUNT THE CHARACTER JUST READ.
5391 027060 001374 BNE 2$ ;LOOP IF NOT TOO MANY CHARS READ FROM FIFO.
5392 027062 000416 BR 8$ ;GO REPORT ERROR IF RX.DATA.VALID WOULDN'T CLR.
5393 ;+
5394 ; Error reports:
5395 ;-
5396 ;Report MR bit would not clear after a DUT reset.
5397 027064 012737 001275 003770 4$: MOV #0701.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5398 027072 012701 006173 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
5399 027076 104460 ERROR ;REPORT ERROR. >>>> ERROR #0701 <<<<<
5400 027076 000415 BR 60$ ;EXIT THE TEST. TRAP C$ERROR
5401 027100 000415
5402
5403 ;Report that RX.DATA.VALID bit was not set after a reset completion.
5404 027102 012737 001276 003770 6$: MOV #0702.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5405 027110 012701 010527 MOV #EM0702,R1 ;SELECT ERROR MESSAGE.

```

HARDWARE TEST

- RDVA -

```

5406 027114          ERROR          ;REPORT ERROR.          >>>> ERROR #0702 <<<<<
      027114 104460          ;EXIT THE TEST.          TRAP    C#ERROR
5407 027116 000406          ;Report that RX.DATA.VALID bit could not be cleared by purging FIFO.
5408                                     MOV    #0703.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
5409                                     MOV    #EM0703,R1  ;SELECT ERROR MESSAGE.
      027120 012737 001277 003770 8$:  ERROR          ;REPORT ERROR.          >>>> ERROR #0703 <<<<<
      027126 012701 010707          TRAP    C#ERROR
      027132 104460
5413
5414 027134 005037 002352 60$: CLR    CTRLCF          ;INDICATE THAT WE COMPLETED THE TEST.
5415
5416 027140          ENDTST
      027140
      027140 104401          L10034:
                                TRAP    C#ETST
    
```

HARDWARE TEST

- RLNA -

```

5418
5419
5420
5421
5422
5423
5424
5425
5426 027142
      027142
5427      000010
5428 027142 012737 000010 002326
5429 027150 012737 177777 002352
5430 027156 012737 000001 003766
5431 027164 012737 011072 003772
5432
5433
5434
5435
5436 027172 012701 011610
5437 027176 012702 000040
5438 027202 005003
5439 027204 013704 002244
5440 027210 010214
5441 027212 004737 022330
5442 027216 004737 020340
5443 027222 103016
5444
5445
5446
5447
5448
5449 027224 005001
5450 027226 012400
5451 027230 011402
5452 027232 010203
5453 027234 000303
5454 027236 042703 177760
5455 027242 020301
5456 027244 001017
5457 027246 005201
5458 027250 020127 000010
5459 027254 001365
5460 027256 000423
5461
5462
5463
5464
5465
5466 027260 012737 001441 003770
5467 027266 012737 016470 003774
5468 027274 012701 006173
5469 027300
      027300 104460
5470 027302 000411
5471
5472
    
```

```

.SBTTL HARDWARE TEST - RLNA -
;+ *****
;* - RBUF RX Line Number Field Test -
;* This test verifies that the DUT RBUF RX Line Number field is working
;* correctly by utilizing the selftest codes which are put in the RX
;* FIFO after a board reset.
;*
;-- *****
      BGNTST
;+
; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
; and wait up to 5 seconds for the MR 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.
      MOV    R2,(R4)       ;SET THE DUT MASTER RESET BIT.
      JSR    PC,SKPSTS     ;SKIP THE SELFTEST.
      JSR    PC,MSLGET     ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
      BCC    4$           ;GO REPORT ERROR IF MR DID NOT CLEAR.
;+
; Read characters from the DUT RX FIFO and verify that the line numbers are
; correct.
; One character is read from the FIFO for each possible line on the DUT.
;--
      CLR    R1            ;CLEAR THE LINE COUNTER.
      MOV    (R4)+,R0      ;INCREMENT POINTER TO PNT TO THE DUT RBUF REG.
2$:   MOV    (R4),R2       ;READ A CHARACTER FROM THE DUT RX FIFO.
      MOV    R2,R3
      SWAB   R3
      BIC    #177760,R3   ;REMOVE ALL BUT LINE NUMBER BITS.
      CMP    R3,R1        ;COMPARE WITH EXPECTED LINE NUMBER.
      BNE    6$          ;GO REPORT ERROR IF LINE NUMBERS DON'T MATCH.
      INC    R1          ;INCREMENT THE EXPECTED LINE NUMBER.
      CMP    R1,#NUMLNS  ;COMPARE WITH NUMBER OF LINES ON DUT.
      BNE    2$         ;LOOP UNTIL CODES FOR ALL LINES ARE READ.
      BR    60$         ;EXIT TEST WITHOUT ERROR.
;+
; Error reports:
;--
;Report MR bit would not clear after a DUT reset.
4$:   MOV    #0801.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
      MOV    #ER0503,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
      MOV    #EM0202,R1    ;SELECT ERROR MESSAGE.
      ERROR ;REPORT ERROR.
;+
      BR    60$          ;EXIT THE TEST.
;Report that RX Line Number field is wrong for selftest code.
    
```

```

>>>>> ERROR #0801 <<<<<
TRAP C#ERROR
    
```


HARDWARE TEST

- BMPCHK -

```

5482
5483
5484
5485
5486
5487
5488
5489
5490
5491
5492
549?
5494 027334
      027334
5495      000011
5496 027334 012737 000011 002326
5497 027342 012737 177777 002352
5498 027350 012737 000001 003766
5499 027356 012737 001605 003770
5500
5501
5502
5503
5504 027364 012701 005670
5505 027370 012702 000040
5506 027374 005003
5507 027376 013704 002244
5508 027402 004737 020340
5509 027406 103027
5510
5511
5512
5513 027410 010214
5514 027412 004737 022330
5515
5516
5517
5518
5519 027416 012704 000764
5520 027422 004737 020300
5521 027426 004737 020716
5522 027432 103015
5523
5524
5525
5526 027434 013702 002524
5527 027440 012703 002526
5528 027444 020203
5529 027446 001414
5530
5531
5532
5533
5534 027450 012701 011236
5535 027454
      027454 104455
      027456 001605
    
```

```

.SBTTL HARDWARE TEST - BMPCHK -
;+ *****
;* - BMP check Test -
;* This test is used to verify that the DUT does not immediately fail
;* the on-board background-monitor program, and hence invalidate
;* succeeding tests.
;* This test looks for BMP codes in the fifo for a set period immediately
;* after the self-test is skipped.
;* Any BMP codes that are found are saved on the queue and are also
;* reported in this test.
;+ *****
;-- BGNTST
      T9::
5496      MOV     #TNUM,TSTNUM ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5497      MOV     #-1,CTRLCF ;SET UP THE TEST NUMBER. (9)
5498      MOV     #1,ERRTYP ;INDICATE THAT WE ARE WITHIN A TEST.
5499      MOV     #Q901.,ERRNBR ;SET ERROR TYPE AS FATAL 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.
;+
; Wait for Master reset to clear. Delay for 500 milli-secs before purging
; the fifo.
;--
      MOV     #500.,R4 ;TIME-OUT VALUE IS 500 MILLI-SECONDS.
      JSR     PC,DELAY ;WAIT FOR BMP TO BEGIN EXECUTION.
      JSR     PC,PUFIFO ;PURGE THE FIFO, SAVING ANY BMP CODES.
      BCC     50# ;ABORT THE TEST IF THE FIFO DID NOT CLEAR.
;+
; Report the error if any BMP codes were found.
;--
      MOV     BMPCP,R2 ;GET THE CONTENTS OF THE POINTER TO THE BMP Q.
      MOV     #BMPQB,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:nnnnn"
      MOV     #EM0902,R1 ;PASS THE MESSAGE TO BE REPORTED.
      ERDF    0901,EM0901,ER9301 ; >>>> ERROR 0901 <<<<<.
;+
      TRAP    C=ERDF
      .WORD   901
    
```


HARDWARE TEST

- BMPCHK -

```
027460 011205
027462 017450
5536 027464 000405
5537
5538 027466 012737 001606 003770 50#: MOV #902.,ERRNBR ;SET >>>> ERROR #0902 <<<<<.
5539 027474 004737 022456 JSR PC,TSABRT ;REPORT NON-TEST RELATED ERROR.
5540
5541 027500 005037 002352 60#: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5542
5543 027504
027504
027504 104401
```

L10036: TRAP C#ETST

HARDWARE TEST

- BMPCHK -

```

5545
5546
5547
5548
5549
5550
5551
5552
5553
5554
5555 027506
      027506
5556      000012
5557 027506 012737 000012 002326
5558 027514 012737 177777 002352
5559 027522 012737 000001 003766
5560 027530 012737 011272 003772
5561 027536 012737 016470 003774
5562
5563
5564
5565
5566 027544 012701 005670
5567 027550 012702 000040
5568 027554 005003
5569 027556 013704 002244
5570 027562 004737 020340
5571 027566 103037
5572
5573
5574
5575
5576
5577 027570 012701 000062
5578 027574 010214
5579 027576 004737 022330
5580 027602 004737 020340
5581 027606 103011
5582 027610 020127 000050
5583 027614 003015
5584
5585
5586
5587
5588
5589
5590 027616 012737 001753 003770
5591 027624 004737 021622
5592 027630 000423
5593
5594
5595
5596
5597 027632 012737 001751 003770
5598 027640 012701 011336
5599 027644
      027644 104460
    
```

```

.SBTTL HARDWARE TEST          - SKSELF -
;+ *****
;*          - Skip Self-test Test -
;* This test verifies that the DUT skips the self-test within the
;* time allowed, and that the fifo contains the correct codes after its
;* completion.
;+ *****
;-- *****
      BGTST
                                  T10:
5556      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5557      MOV  #TNUM,TSTNUM    ;SET UP THE TEST NUMBER. (10)
5558      MOV  #-1,CTRLCF     ;INDICATE THAT WE ARE WITHIN A TEST.
5559      MOV  #1,ERRTYP      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5560      MOV  #EM1001,ERRMSG  ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5561      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.
;+
; Determine if the DUT takes too short or too long a time to skip the self-test
; Set-up a time-out of 50 milli-second, if MR is clear in less than 10 milli
; -second, or greater than 50 milli-seconds, report the error.
;--
      MOV  #50.,R1           ;TIME-OUT VALUE IS 50 MILLI-SECONDS.
      MOV  R2,(R4)           ;SET THE DUT MASTER RESET BIT.
      JSR  PC,SKPSTS         ;WRITE THE SKIP SELFTEST CODES TO THE DUT.
      JSR  PC,MSLGET         ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
      BCC  2$                ;GO REPORT ERR IF SKIPPING STEST TOOK TOO LONG.
      CMP  R1,#40.           ;GO REP ERR IF SELFTEST COMPLETED IN < 10 MS.
      BGT  4$
;+
; Self-test completed within 10 milli-sec to 50 milli-seconds.
; Verify that the self-test codes in the fifo are "good" codes ,ie the DUT
; successfully completed the self-test.
; This subroutines reports errors with numbers >>>> 1003 thru 1007 <<<<.
;--
      MOV  #1003.,ERRNBR     ;SET ERROR NUMBER TO 1003.
      JSR  PC,RSTRPT        ;CHECK SELF-TEST CODES IN THE FIFO.
      BR   60$              ;EXIT TEST.
;+
; Error reports:
;--
;Report Skip Self-test took too long.
5597      MOV  #1001.,ERRNBR  ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5598      MOV  #EM1002,R1    ;SELECT ERROR MESSAGE.
5599      ERROR              ;REPORT ERROR. >>>> ERROR #1001 <<<<
                                  TRAP C$ERROR
    
```

HARDWARE TEST

- SKSELF -

```

5600 027646 000414          BR      60$          ;EXIT THE TEST.
5601
5602
5603 027650 012737 001752 003770 4$: ;Report Skip Self-test completed too soon.
5604 027656 012701 011423          MOV    #1002.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5605 027662          MOV    #EM1003,R1 ;SELECT ERROR MESSAGE.
          ERROR          ;REPORT ERROR.          >>>>> ERROR #1002 <<<<<
          TRAP          C$ERROR
5606 027664 000405          BR      60$          ;EXIT THE TEST.
5607
5608 027666 012737 001753 003770 50$: MOV    #1003.,ERRNBR ;SET ERROR NUMBER.
5609 027674 004737 022456          JSR    PC,TSABRT    ;REPORT NON-TEST RELATED ERROR.
5610
5611 027700 005037 002352          60$: CLR    CTRLCF    ;INDICATE THAT WE COMPLETED THE TEST.
5612
5613 027704          ENDTST
          027704          L10037:
          027704 104401          TRAP    C$ETST
    
```

HARDWARE TEST

- SKSELF -

```

5615
5616
5617
5618
5619
5620
5621
5622
5623
5624 027706
      027706
5625      000013
5626 027706 012737 000013 002326
5627 027714 012737 177777 002352
5628 027722 012737 000001 003766
5629 027730 012737 011501 003772
5630 027736 012737 016470 003774
5631
5632
5633
5634
5635 027744 012701 005670
5636 027750 012702 000040
5637 027754 005003
5638 027756 013704 002244
5639 027762 004737 020340
5640 027766 103044
5641
5642
5643
5644 027770 010214
5645 027772 004737 022330
5646
5647
5648
5649
5650 027776 012701 000005
5651 030002 012702 020000
5652 030006 010203
5653 030010 013704 002244
5654 030014 004737 020340
5655 030020 103020
5656
5657
5658
5659
5660
5661
5662
5663 030022 012701 000017
5664 030026 005003
5665 030030 004737 020340
5666 030034 103012
5667 030036 010105
5668 030040 012701 000001
5669 030044 052703 020000
5670 030050 004737 020340
    
```

```

.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-boarded selftest is skipped.
;*
;-- *****
      BGNTST
                                     T11::
5625      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
5626      MOV    #TNUM,TSTNUM      ;SET UP THE TEST NUMBER.          (11)
5627      MOV    #-1,CTRLCF        ;INDICATE THAT WE ARE WITHIN A TEST.
5628      MOV    #1,ERRTYP         ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
5629      MOV    #EM1101,ERRMSG    ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
5630      MOV    #ER0503,ERRBLK    ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
5631
5632      ;+
5633      ; Wait up to 3 seconds for the DUT Master Reset bit to clear.
5634      ; 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.
5641
5642      ;+
5643      ; 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.
5646
5647      ;+
5648      ; Set time out of 5 milli seconds, wait for Diag_fail bit to set.
5649      ; 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.
5656
5657      ;+
5658      ; Set time-out of 15 milli-secs, wait for Diag_Fail to clear.
5659      ; If time-out occurs go report the error.
5660      ; Verify the Diag_fail bit is in a stable state before continuing. Loop
5661      ; back if the state was transitory, using the remainder of the 15 ms time-out.
;--
      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 -

```

5671 030054 103016          BCC 60$          ;EXIT IF DIAG_FAIL BIT STILL CLEAR.
5672 030056 010501          MOV  R5,R1       ;PASS THE REMAINING TIME-OUT VALUE.
5673 030060 000762          BR   2$          ;LOOP TO CHECK AGAIN.
5674
5675          ;+
5676          ; Error reports:
5677          ;-
5678 030062 012737 002115 003770 4$: ;Report Diagnostic fail bit bad.
5679 030070 012701 011742          MOV  #1101.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5680 030074          MOV  #EM1205,R1 ;SELECT ERROR MESSAGE.
5681 030076 104460          ERROR ;REPORT ERROR.          >>>> ERROR #1101 <<<<<
5682 030076 000405          BR   60$          ;EXIT THE TEST.          TRAP  C$ERROR
5683 030100 012737 002116 003770 50$: MOV  #1102.,ERRNBR ;SET THE ERROR NUMBER FOR TSABRT RTN.
5684 030106 004737 022456          JSR  PC,TSABRT   ;REPORT NON-TEST RELATED ERROR.
5685
5686 030112 005037 002352          60$: CLR  CTRLCF   ;INDICATE THAT WE COMPLETED A TEST.
5687
5688 030116          ENDTST
          030116
          030116 104401          L10040:          TRAP  C$ETST
    
```

HARDWARE TEST

- SELFTS -

```

5690
5691
5692
5693
5694
5695
5696
5697
5698 030120
      030120
5699      000014
5700 030120 012737 000014 002326
5701 030126 012737 177777 002352
5702 030134 012737 000001 003766
5703 030142 012737 011525 003772
5704 030150 012737 016470 003774
5705
5706
5707
5708
5709 030156 012701 005670
5710 030162 012702 000040
5711 030166 005003
5712 030170 013704 002244
5713 030174 004737 020340
5714 030200 103062
5715
5716
5717
5718
5719
5720 030202 012701 005670
5721 030206 010214
5722 030210 004737 020340
5723 030214 103030
5724 030216 012702 005670
5725 030222 160102
5726 030224 020227 000062
5727 030230 002431
5728 030232 020227 000764
5729 030236 002434
5730
5731
5732
5733
5734 030240 032714 020000
5735 030244 001406
5736
5737 030246 012737 002264 003770
5738 030254 012701 011742
5739 030260
      030260 104460
5740
5741
5742
5743
5744
    
```

```

.SBTTL HARDWARE TEST - SELFTS -
;+ *****
;* - Self-test Test -
;* This test verifies that the DUT's Self-Test executes within the
;* time allowed, and that the fifo contains the correct codes after its
;* completion.
;+
;-- *****
      BGNTST
;+
; T12::
; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      TNUM == TNUM + 1
; SET UP THE TEST NUMBER. (12)
      MOV #TNUM,TSTNUM
; INDICATE THAT WE ARE WITHIN A TEST.
      MOV #-1,CTRLCF
; SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV #1,ERRTYP
; SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
      MOV #EM1201,ERRMSG
; SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
      MOV #ER0503,ERRBLK
;+
; 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.
;+
; Determine if the Self-test takes too short or too long a time to complete.
; Set-up a time-out of 3 second, if MR is clear in less than 1/2 second, or
; greater than 3 seconds, report the error.
;--
      MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
      MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
      JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
      BCC 4$ ;GO REPORT ERROR SELFTEST TOOK TOO LONG.
      MOV #3000.,R2
      SUB R1,R2 ;CALCULATE # OF MS SELFTEST TO COMPLETE.
      CMP R2,#50.
      BLT 6$ ;SELFTEST SKIPPED? YES, GO REPORT ERROR.
      CMP R2,#500.
      BLT 8$ ;GO REP ERR IF SELFTEST COMPLETED IN < 1/2 SEC.
;+
; Self-test completed within 1sec to 3 seconds.
; Check the state of the Diagnostic Fail bit, report error if it is set.
;--
      BIT #BIT13,(R4) ;DETERMINE IF THE DIAG_FAIL BIT IS CLEAR.
      BEQ 2$ ;SKIP ERROR REPORT IF BIT IS CLEAR.
;Report Diagnostic fail bit bad.
      MOV #1204.,ERRNBR ;SET ERROR NUMBER TO IN ERROR TABLE.
      MOV #EM1205,R1 ;SELECT THE ERROR MESSAGE.
      ERROR ; >>>> ERROR #1204 <<<<<
; TRAP C$ERROR
;+
; Verify that the self-test codes in the fifo are "good" codes ,ie the DUT
; successfully completed the self-test.
; This subroutine reports errors with numbers >>>> 1205 thru 1209 <<<<<.
;--
    
```

HARDWARE TEST

- SELFTS -

```

5745 030262 012737 002265 003770 2$:   MOV   #1205.,ERRNBR   ;SET ERROR NUMBER TO 1205.
5746 030270 004737 021622                JSR   PC,RSTRPT     ;CHECK SELF-TEST CODES IN THE FIFO.
5747 030274 000431                BR    60$           ;EXIT TEST.
5748                ;+
5749                ; Error reports:
5750                ; -
5751                ;Report Self-test took too long to complete.
5752 030276 012737 002261 003770 4$:   MOV   #1201.,ERRNBR   ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5753 030304 012701 011544                MOV   #EM1202,R1     ;SELECT ERROR MESSAGE.
5754 030310                ERROR                ;REPORT ERROR.          >>>> ERROR #1201 <<<<<
                    030310 104460                TRAP   C$ERROR
5755 030312 000422                BR    60$           ;EXIT THE TEST.
5756
5757                ;Report Self-test did not execute after DUT reset.
5758 030314 012737 002262 003770 6$:   MOV   #1202.,ERRNBR   ;SET THE ERROR NUMBER IN ERROR TABLE.
5759 030322 012701 011706                MOV   #EM1204,R1     ;SELECT ERROR MESSAGE.
5760 030326                ERROR                ;REPORT ERROR.          >>>> ERROR #1202 <<<<<
                    030326 104460                TRAP   C$ERROR
5761
5762                ;Report Self-test competed too soon.
5763 030330 012737 002263 003770 8$:   MOV   #1203.,ERRNBR   ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5764 030336 012701 011630                MOV   #EM1203,R1     ;SELECT ERROR MESSAGE.
5765 030342                ERROR                ;REPORT ERROR.          >>>> ERROR #1203 <<<<<
                    030342 104460                TRAP   C$ERROR
5766 030344 000405                BR    60$           ;EXIT THE TEST.
5767
5768 030346 012737 002272 003770 50$:  MOV   #1210.,ERRNBR   ;SET THE ERROR NUMBER FOR TSABRT RTN.
5769 030354 004737 022456                JSR   PC,TSABRT     ;REPORT NON-TEST RELATED ERROR.
5770
5771 030360 005037 002352                60$:  CLR   CTRLCF         ;INDICATE THAT WE COMPLETED THE TEST.
5772
5773 030364                ENDTST
                    030364                L10041:
                    030364 104401                TRAP   C$ETST

```

HARDWARE TEST

- SELFTS -

```

5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785 030366
      030366
5786          000015
5787 030366 012737 000015 002326
5788 030374 012737 177777 002352
5789 030402 012737 000001 003766
5790 030410 012737 011766 003772
5791 030416 012737 016470 003774
5792 030424 012737 002425 003770
5793
5794
5795
5796
5797 030432 012701 005670
5798 030436 012702 000040
5799 030442 005003
5800 030444 013704 002244
5801 030450 004737 020340
5802 030454 103120
5803
5804
5805
5806
5807 030456 010214
5808 030460 004737 022330
5809 030464 012701 000764
5810 030470 004737 020340
5811 030474 103110
5812 030476 005237 003770
5813 030502 012700 000010
5814 030506 005003
5815 030510 013704 002246
5816 030514 011402
5817 030516 100077
5818 030520 042702 007402
5819 030524 020227 170001
5820 030530 001002
5821 030532 012703 177777
5822 030536 005300
5823 030540 001365
5824 030542 005703
5825 030544 100466
5826
5827
5828
5829
5830

```

```

.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
;+
; 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$               ;GO REPORT ERROR IF MR DID NOT CLEAR.
;+
; Reset the DUT, check for rom version 0.
; If Version 0 is found then exit this test.
;-
      MOV    R2,(R4)           ;RESET THE DUT.
      JSR    PC,SKPSTS         ;SKIP THE SELFTEST.
      MOV    #500.,R1          ;PASS TIME-OUT VALUE OF 500 MILLISECS.
      JSR    PC,MSLGET         ;WAIT FOR MR BIT TO CLEAR.
      BCC    50$               ;GO REPORT ERROR IF TIME-OUT OCCURRED.
      INC    ERRNBR            ;SET ERROR NUMBER TO 1302.
      MOV    #8.,R0            ;SET MAXIMUM READ COUNT.
      CLR    R3                 ;CLEAR THE ROM VERSION 0 FLAG.
      MOV    RBUFA,R4          ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
2$:   MOV    (R4),R2            ;READ A CODE FROM THE FIFO.
      BPL    50$               ;GO REPORT ERROR IF THE FIFO IS EMPTY.
      BIC    #7402,R2          ;REMOVE THE LINE NUMBER AND PROC INDICATOR.
      CMP    R2,#170001        ;COMPARE WITH ROM VERSION #0 CODE.
      BNE    4$                ;ROM VERSION #0? NO, SKIP SETTING FLAG.
      MOV    #-1,R3            ;YES, SET THE ROM VERSION #0 FLAGS.
4$:   DEC    R0                 ;DECREMENT MAX READ COUNTER.
      BNE    2$                ;LOOP IF 8 CODES HAVE NOT BEEN READ.
      TST   R3                 ;CHECK THE ROM VERSION #1 INDICATOR.
      BMI    60$               ;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 -

```

5831 030546 012777 000040 151470      MOV    #BIT05,@CSRA      ;SET DUT MASTER RESET BIT, SELECT CHANNEL 0.
5832 030554 012704 000031              MOV    #25.,R4          ;PASS DELAY PERIOD OF 25 MILLI SECS.
5833 030560 004737 020300              JSR    PC.DELAY         ;WAIT FOR SELFTEST TO INITIALISE.
5834 030564 012777 146314 151470      MOV    #146314,@TXBFCA ;WRITE THE FAIL SELF-TEST CODE TO TBUFFCT REG.
5835
5836                                     ;+
5837                                     ; Wait up to 2 seconds for the self-test to complete.
5838                                     ; If time-out occurs, then exit this test.
5839 030572 005237 003770              ; -
5840 030576 012701 003720              INC    ERRNBR           ;SET ERROR NUMBER TO 1303.
5841 030602 012702 000040              MOV    #2000.,R1       ;TIME-OUT VALUE IS 2.0 SECONDS.
5842 030606 005003                      MOV    #BIT05,R2       ;PASS THE BIT MAP OF THE BIT TO TEST.
5843 030610 013704 002244              CLR    R3              ;SET UP THE EXPECTED STATE.
5844 030614 004737 020340              MOV    CSRA,R4         ;BIT IS IN THE DUT'S CSR.
5845 030620 103036                      JSR    PC,MSLGET       ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
5846                                     BCC   50$              ;GO REPORT ERROR IF MR DID NOT CLEAR.
5847                                     ;+
5848                                     ; Verify the diagnostic fail bit is set, indicating the error.
5849                                     ; Report error if diagnostic fail bit is clear.
5850 030622 005237 003770              ; -
5851 030626 032714 020000              INC    ERRNBR           ;SET ERROR NUMBER TO 1304.
5852 030632 001425                      BIT    #BIT13,(R4)     ;CHECK THE STATE OF THE DIAG_FAIL BIT.
5853                                     BEQ   10$              ;GO REPORT ERROR IF DIAG_FAIL BIT CLEAR.
5854                                     ;+
5855                                     ; Remove the 8 self-test codes form the fifo, and verify that at least
5856                                     ; one is a Procl to ram error code (231).
5857 030634 005237 003770              ; -
5858 030640 012700 000010              INC    ERRNBR           ;SET ERROR NUMBER TO 1305.
5859 030644 005001                      MOV    #8.,R0          ;SET MAXIMUM READ COUNT.
5860 030646 013704 002246              CLR    R1              ;CLEAR THE CORRECT CODE COUNTER.
5861 030652 011402                      MOV    RBUFA,R4        ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
5862 030654 100020                      6$:  MOV    (R4),R2      ;READ A CODE FROM THE FIFO.
5863 030656 042702 007400              BPL   50$              ;GO REPORT ERROR IF THE FIFO IS EMPTY.
5864 030662 120227 170231              BIC   #7400,R2         ;REMOVE THE LINE NUMBER FROM THE CODE.
5865 030666 001001                      CMPB  R2,#170231       ;IS IT THE CORRECT ERROR CODE?.
5866 030670 005201                      BNE   8$              ;SKIP NEXT INSTRUCTION, IF NOT A 231 CODE.
5867 030672 005300                      INC   R1              ;INCREMENT COUNTER.
5868 030674 001366                      8$:  DEC   R0           ;DECREMENT MAX READ COUNTER.
5869 030676 005701                      BNE   6$              ;LOOP IF 8 CODES HAVE NOT BEEN READ.
5870 030700 001010                      TST  R1              ;WERE ANY 231 CODES FOUND?.
5871 030702 005237 003770              BNE   60$             ;YES, THEN EXIT.
5872                                     INC   ERRNBR           ;NO, SET ERROR NUMBER TO 1306 AND REPORT ERROR.
5873 030706 012701 012012              ;Report Self-test error reporting bad.
5874 030712 010460                      10$: MOV    #EM1302,R1  ;SELECT ERROR MESSAGE.
5875 030714 000402                      ERROR ;REPORT ERROR.          >>>>> ERROR <<<<<
5876                                     BR    60$              ;EXIT THE TEST.          TRAP    C$ERROR
5877 030716 004737 022456              50$: JSR    PC,TSABRT  ;REPORT NON-RELATED TEST ERROR.
5878
5879 030722 005037 002352              60$: CLR    CTRLCF     ;INDICATE THAT WE COMPLETED THE TEST.
5880
5881 030726                                ENDTST
5881 030726                                L10042:
5881 030726 104401                                TRAP    C$ETST

```

HARDWARE TEST

- STFAIL -

```

5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894 030730
      030730
5895      000016
5896 030730 012737 000016 002326
5897 030736 012737 177777 002352
5898 030744 012737 000001 003766
5899 030752 012737 012051 003772
5900 030760 012737 016470 003774
5901
5902
5903
5904
5905 030766 012701 005670
5906 030772 012702 000040
5907 030776 005003
5908 031000 013704 002244
5909 031004 004737 020340
5910 031010 103131
5911
5912
5913
5914 031012 010214
5915 031014 004737 022330
5916 031020 012701 005670
5917 031024 004737 020340
5918 031030 103121
5919
5920
5921
5922
5923
5924
5925
5926 031032 012705 000040
5927 031036 012703 000143
5928 031042 010304
5929 031044 012737 002571 003770
5930 031052 012701 012101
5931
5932 031056 017702 151164
5933 031062 100077
5934
5935
5936
5937 031064 012700 000301
5938 031070 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::
      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV    #TNUM,TSTNUM      ;SET UP THE TEST NUMBER.          (14)
      MOV    #-1,CTRLCF        ;INDICATE THAT WE ARE WITHIN A TEST.
      MOV    #1,ERRTYP         ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV    #EM1401,ERRMSG     ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
      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.
;+
; 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 -

```

5939 031072 001003          BNE      4$          ;BRANCH IF NOT A BMP CODE.
5940 031074 004737 022262   JSR      PC,SAVBMP    ;SAVE THE BMP CODE ON THE QUEUE.
5941 031100 000435          BR       8$          ;
5942                               ;+
5943                               ; Check if the read data is a self-test code.
5944                               ;-
5945 031102 012700 000201   4$:     MOV      @201,R0    ;SET-UP A BIT MASK OF A SELFTEST CODE.
5946 031106 040200          BIC      R2,R0        ;TRY TO CLEAR THE BIT MASK WITH THE READ DATA.
5947 031110 001431          BEQ      8$          ;BRANCH IF IT IS A SELFTEST CODE.
5948
5949                               ;+
5950                               ; The read data is a ROM version number, determine which one it is.
5951                               ;-
5952 031112 032702 000002   BIT      @BIT1,R2     ;CHECK THE PROCESSOR NUMBER BIT IN THE CODE.
5953 031116 001407          BEQ      6$          ;BRANCH IF IT IS PROC_1 ROM VERSION NUMBER.
5954 031120 010204          MOV      R2,R4        ;SAVE PROC_2 ROM VERSION NUMBER.
5955 031122 042704 177603   BIC      @177603,R4   ;CLEAR ANY UNWANTED BITS.
5956 031126 000241          CLC                    ;CLEAR THE CARRY BIT.
5957 031130 006004          ROR      R4          ;SHIFT THE CODES ALONG TO GET THE ROM
5958 031132 006004          ROR      R4          ; VERSION NUMBER IN THE LOW 5 BITS.
5959 031134 000417          BR       8$          ;
5960 031136 010203          6$:     MOV      R2,R3        ;SAVE PROC_1 ROM VERSION NUMBER.
5961 031140 042703 177603   BIC      @177603,R3   ;CLEAR ANY UNWANTED BITS.
5962 031144 000241          CLC                    ;CLEAR THE CARRY BIT.
5963 031146 006003          ROR      R3          ;SHIFT THE CODE ALONG TO GET THE ROM
5964 031150 006003          ROR      R3          ; VERSION NUMBER IN THE LOW 5 BITS.
5965 031152 020427 000143   CMP      R4,@99.     ;CHECK IF WE HAVE RECEIVE PROC_2 ROM CODE.
5966 031156 001016          BNE     10$         ;GO REPORT BOTH ROM VERSION NUMBERS.
5967
5968                               ;+
5969                               ; Received ROM version numbers out of sequence.
5970                               ; ie. Proc_1's ROM version number found in the fifo before Proc_2's.
5971 031160 012701 012167   ;-
5972 031164 012737 002572 003770   MOV      @EM1403,R1   ;SELECT THE ERROR MESSAGE TO BE REPORTED.
5973 031172 000433          MOV      @1402.,ERRNBR ;SET THE ERROR NUMBER.
5974                               BR       12$         ;GO REPORT ERROR.
5975 031174 005305          8$:     DEC      R5          ;DECREMENT THE MAX TRY COUNTER.
5976 031176 001327          BNE     2$          ;LOOP TO GET THE NEXT CHAR FROM THE FIFO.
5977 031200 012701 012242   MOV      @EM1404,R1   ;SELECT THE ERROR MESSAGE TO BE REPORTED.
5978 031204 012737 002573 003770   MOV      @1403.,ERRNBR ;SET THE ERROR NUMBER.
5979 031212 000423          BR       12$         ;GIVE UP, GO REPORT ERROR.
5980
5981                               ;+
5982                               ; If this is the first pass, and software P-table question was answered YES,
5983                               ; then report the rom version numbers to the operator.
5984 031214 032737 000001 002226 10$:    BIT      @BIT0,OPTION ;CHECK ON THE STATE OF THE SOFTWARE SWITCH.
5985 031222 001431          BEQ     60$         ;EXIT IF NO ROM VERSION PRINTOUT WAS REQUESTED.
5986 031224 023727 002332 000001   CMP      PASCNT,@1    ;CHECK IF THIS IS THE FIRST PASS.
5987 031232 003025          BGT     60$         ;EXIT IF ROM VERS HAVE ALREADY BEEN REPORTED.
5988 031234          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 -

```

5989 031260 000412          BR      60$          ;EXIT THIS TEST.
5990
5991                      ; Error reports:
5992                      ;
5993 031262 012737 016562 003774 12$:  MOV    #ER1401,ERRBLK ;SELECT THE ERROR REPORTING ROUTINE.
5994 031270          104460          ERROR          ;REPORT ERROR.          >>>>> ERROR <<<<<<
5995 031272 000405          BR      60$          TRAP    C#ERROR
5996
5997 031274 012737 002575 003770 50$:  MOV    #1405.,ERRNBR ;SET UP ERROR NUMBER FOR TSABRT RTN.
5998 031302 004737 022456          JSR    PC,TSABRT    ;REPORT NON-TEST RELATED ERROR.
5999
6000 031306 005037 002352          60$:  CLR    CTRLCF      ;INDICATE THAT WE COMPLETED THE TEST.
6001
6002 031312          ENDTST
        031312
        031312 104401          L10043:
                                TRAP    C#ETST
    
```

HARDWARE TEST

- REGWRW -

```

6004
6005
6006
6007
6008
6009
6010
6011
6012
6013 031314
      031314
6014      000017
6015 031314 012737 000017 002326
6016 031322 012737 177777 002352
6017 031330 012737 000001 003766
6018 031336 012737 003101 003770
6019 031344 012737 012445 003772
6020 031352 005037 002462
6021 031356 012700 002464
6022 031362 004737 020200
6023
6024
6025
6026
6027
6028 031366 004737 021456
6029 031372 103402
6030 031374 000137 031516
6031
6032
6033
6034 031400 005237 003770
6035 031404 012702 000017
6036 031410 013704 002244
6037 031414 010214
6038 031416 011401
6039 031420 042701 177760
6040 031424 020102
6041 031426 001406
6042
6043 031430 012737 016712 003774
6044 031436 005003
6045 031440 005005
6046 031442
      031442 104460
6047 031444 005302
6048 031446 002362
6049
6050
6051
6052
6053
6054 031450 005237 003770
6055 031454 005003
6056 031456 012704 000002
6057 031462 004737 021220
6058
    
```

```

.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 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 >>>> 1601 <<<<.
;-
      JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCS .+6 ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
      JMP 604 ;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.
24: 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 44 ;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
44: DEC R2 ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
      BGE 24 ;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

- REGWRW -

```

6059
6060 ; Write and verify 16 data patterns in all used bits of all registers on all
6061 ; active lines. Before writing each pattern, set all the bits.
6062 ; REGTST routine reports errors with numbers >>>> ERROR 1606 - 1608 <<<<.
6063 031466 012737 003106 003770 ;-
6064 031474 005003 MOV @1606.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6065 031476 005403 CLR R3 ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
6066 031500 004737 021220 NEG R4 ;INDICATE R/W ACCESS, SET FIRST.
6067 JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6068 ;
6069 ; Print error summary reports if necessary.
6070 ; The following routine reports errors with number >>>> ERROR # 1609 <<<<
6071 031504 012737 003111 003770 ;-
6072 031512 004737 021430 MOV @1609.,ERRNBR ;SET UP ERROR NUMBER FOR NEXT RTN.
6073 031516 005037 002352 JSR PC,REPSMR ;REPORT ERROR SUMMARY IF NECESSARY.
6074 031522 60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
        031522
        031522 104401

```

L10044: TRAP C\$ETST

HARDWARE TEST

- REGWRM -

```

6076
6077
6078
6079
6080
6081
6082
6083
6084
6085 031524
      031524
6086      000020
6087 031524 012737 000020 002326
6088 031532 012737 177777 002352
6089 031540 012737 000001 003766
6090 031546 012737 003245 003770
6091 031554 012737 012513 003772
6092 031562 005037 002462
6093 031566 012700 002464
6094 031572 004737 020200
6095
6096
6097
6098
6099
6100 031576 004737 021456
6101 031602 103402
6102 031604 000137 031732
6103
6104
6105
6106 031610 005237 003770
6107 031614 012702 000017
6108 031620 013704 002244
6109 031624 042714 000017
6110 031630 050214
6111 031632 011401
6112 031634 042701 177760
6113 031640 020102
6114 031642 001406
6115
6116 031644 012737 016712 003774
6117 031652 005003
6118 031654 005005
6119 031656
      031656 104460
6120 031660 005302
6121 031662 002360
6122
6123
6124
6125
6126
6127 031664 005237 003770
6128 031670 005003
6129 031672 012704 000001
6130 031676 004737 021220
    
```

```

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

      BGNTST

      T16::
      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV  #TNUM,TSTNUM    ;SET UP THE TEST NUMBER.      (17)
      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE WITHIN A TEST.
      MOV  #1,ERRTYP       ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
      MOV  #1701,ERRNBR    ;SET UP ERROR NUMBER IN THE ERROR TABLE.
      MOV  #EM1701,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 >>>> 1701 <<<<.
;--
      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/modify/write capability to indirect address field of CSR
;--
      INC  ERRNBR          ;SET THE ERROR REPORT NUMBER TO 1702.
      MOV  #17,R2          ;SET LOOP COUNT.
      MOV  CSRA,R4         ;GET CSR ADDRESS.
2$:   BIC  #17,(R4)        ;CLEAR THE DUT CSR USING READ/MODIFY/WRITE.
      BIS  R2,(R4)        ;WRITE COUNT TO CSR USING READ/MODIFY/WRITE.
      MOV  (R4),R1        ;READ BACK THE CONTENTS OF THE CSR
      BIC  #177760,R1     ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
      CMP  R1,R2          ;CHECK FOR CORRECT DATA WRITTEN/READ.
      BEQ  4$             ;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 # 1702 <<<<
;--
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 using R/M/W. Before writing each pattern, clear all the bits.
; REGTST routine reports errors with numbers >>>> ERROR 1703 - 1705 <<<<.
;--
      INC  ERRNBR          ;SET THE ERROR NUMBER TO 1703.
      CLR  R3              ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
      MOV  #1,R4          ;INDICATE R/M/W ACCESS, CLEAR FIRST.
      JSR  PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
    
```


HARDWARE TEST

- REGBRW -

```

6149 .SBTTL HARDWARE TEST - REGBRW -
6150 ;** *****
6151 ;* - Device Register Byte Access Read and Write Test -
6152 ;*
6153 ;* This test verifies that the device registers can be read and written
6154 ;* correctly using byte accesses.
6155 ;*
6156 ;-- *****
6157
6158 031740 BGNTST
6159 031740
6159 000021 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6160 031740 012737 000021 002326 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (18)
6161 031746 012737 177777 002352 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
6162 031754 012737 000001 003766 MOV #1,ERRTYP ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
6163 031762 012737 003411 003770 MOV #1801.,ERRNBR ;SET UP ERROR NUMBER IN THE ERROR TABLE.
6164 031770 012737 012570 003772 MOV #EM1801,ERRMSG ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
6165 031776 005037 002462 CLR ERSMRF ;CLEAR THE ERROR SUMMARY FLAGS.
6166 032002 012700 002464 MOV #ERCNTB,R0
6167 032006 004737 020200 JSR PC,CLR16W ;CLEAR THE ERROR COUNTER TABLE.
6168
6169 ;+
6170 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
6171 ; Clear TX and RX interrupt enable bits in the CSR.
6172 ; This subroutine reports errors >>>> 1801 <<<<.
6173 032012 004737 021456 ;--
6174 032016 103402 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6175 032020 000137 032176 BCS .+6 ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
6176 032024 012737 003412 003770 JMP 60$ ;YES, EXIT THE TEST.
6177 MOV #1802.,ERRNBR ;SET THE ERROR REPORT NUMBER TO 1802.
6178
6179 ;+
6180 ; Verify read/write capability to indirect address field of CSR.
6181 ; Use byte accesses.
6182 032032 012702 000017 ;--
6183 032036 013704 002244 MOV #17,R2 ;SET LOOP COUNT.
6184 032042 110214 MOV CSRA,R4 ;GET CSR ADDRESS.
6185 032044 111401 2$: MOVB R2,(R4) ;WRITE COUNT TO CSR.
6186 032046 042701 177760 MOVB (R4),R1 ;READ BACK THE CONTENTS OF THE CSR
6187 032052 020102 BIC #177760,R1 ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
6188 032054 001406 CMP R1,R2 ;CHECK FOR CORRECT DATA WRITTEN/READ.
6189 032056 012737 016712 003774 BEQ 4$ ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
6190 032064 005003 ;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
6191 032066 005005 MOV #ER1601,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
6192 032070 104460 CLR R3 ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
6193 032072 005302 CLR R5 ;CAUSE REPORT OF LINE 0.
6194 032074 002362 ERROR ; >>>> ERROR # 1802 <<<<
6195
6196 ;+
6197 ; Write and verify 16 data patterns in all used bits of all lower bytes of all
6198 ; registers on all active lines. Use READ/WRITE accesses. Before writing
6199 ; each pattern, clear all the used bits of all active registers.
6200 ; REGTST routine reports errors with numbers >>>> ERROR 1803 - 1805 <<<<.
6201 032076 005237 003770 ;--
6202 032102 012703 177777 INC ERRNBR ;SET THE ERROR NUMBER TO 1803.
6203 032106 012704 000002 MOV #-1,R3 ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
MOV #2,R4 ;INDICATE R/W ACCESS, CLEAR FIRST.

```

HARDWARE TEST

- REGBRW -

```

6204 032112 004737 021220      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, clear all the used bits of all active registers.
6209                               ; REGTST routine reports errors with numbers >>>> ERROR 1806 - 1808 <<<<.
6210                               ;-
6211 032116 012737 003416 003770      MOV      #1806.,ERRNBR   ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6212 032124 005403                               NEG      R3              ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6213 032126 004737 021220      JSR      PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
6214                               ;+
6215                               ; Write and verify 16 data patterns in all used bits of all lower bytes of all
6216                               ; registers on all active lines. Use READ/WRITE accesses. Before writing
6217                               ; each pattern, set all the used bits of all active registers.
6218                               ; REGTST routine reports errors with numbers >>>> ERROR 1809 - 1811 <<<<.
6219                               ;-
6220 032132 012737 003421 003770      MOV      #1809.,ERRNBR   ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6221 032140 005403                               NEG      R3              ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
6222 032142 005404                               NEG      R4              ;INDICATE R/W ACCESS, SET FIRST.
6223 032144 004737 021220      JSR      PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
6224                               ;+
6225                               ; Write and verify 16 data patterns in all used bits of all high bytes of all
6226                               ; registers on all active lines. Use READ/WRITE accesses. Before writing
6227                               ; each pattern, set all the used bits of all active registers.
6228                               ; REGTST routine reports errors with numbers >>>> ERROR 1812 - 1814 <<<<.
6229                               ;-
6230 032150 012737 003424 003770      MOV      #1812.,ERRNBR   ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6231 032156 005403                               NEG      R3              ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6232 032160 004737 021220      JSR      PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
6233                               ;+
6234                               ; Print error summary reports if necessary.
6235                               ; The following routine reports errors with number >>>> ERROR # 1815 <<<<
6236                               ;-
6237 032164 012737 003427 003770      MOV      #1815.,ERRNBR   ;SET UP ERROR NUMBER FOR NEXT RTN.
6238 032172 004737 021430      JSR      PC,REPSMR      ;REPORT ERROR SUMMARY IF NECESSARY.
6239 032176 005037 002352      60$:    CLR      CTRLCF   ;INDICATE THAT WE COMPLETED THE TEST.
6240 032202                               ENDTST
        032202
        032202 104401

```

```

L10046:
        TRAP      C$ETST

```

HARDWARE TEST

- REGBRM -

```

6242
6243
6244
6245
6246
6247
6248
6249
6250
6251 032204
      032204
6252      000022
6253 032204 012737 000022 002326
6254 032212 012737 177777 002352
6255 032220 012737 000001 003766
6256 032226 012737 003555 003770
6257 032234 012737 012636 003772
6258 032242 005037 002462
6259 032246 012700 002464
6260 032252 004737 020200
6261
6262
6263
6264
6265
6266 032256 004737 021456
6267 032262 103402
6268 032264 000137 032446
6269 032270 012737 003556 003770
6270
6271
6272
6273
6274 032276 012702 000017
6275 032302 013704 002244
6276 032306 142714 000017
6277 032312 150214
6278 032314 111401
6279 032316 042701 177760
6280 032322 020102
6281 032324 001406
6282
6283 032326 012737 016712 003774
6284 032334 005003
6285 032336 005005
6286 032340
      032340 104460
6287 032342 005302
6288 032344 002360
6289
6290
6291
6292
6293
6294
6295 032346 005237 003770
6296 032352 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 ERSRFB ;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 -

```

6297 032356 012704 000001      MOV    #1,R4      ;INDICATE R/M/W ACCESS, CLEAR FIRST.
6298 032362 004737 021220      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, clear all the used bits of all active registers.
6304      ; REGTST routine reports errors with numbers >>>> ERROR 1906 - 1908 <<<<.
6305 032366 012737 003562 003770      MOV    #1906.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6306 032374 005403                NEG    R3         ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6307 032376 004737 021220      JSR    PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6308
6309      ;+
6310      ; Write and verify 16 data patterns in all used bits of all lower bytes of all
6311      ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
6312      ; writing each pattern, set all the used bits of all active registers.
6313      ; REGTST routine reports errors with numbers >>>> ERROR 1909 - 1911 <<<<.
6314 032402 012737 003565 003770      MOV    #1909.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6315 032410 005403                NEG    R3         ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
6316 032412 005404                NEG    R4         ;INDICATE R/M/W ACCESS, SET FIRST.
6317 032414 004737 021220      JSR    PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6318
6319      ;+
6320      ; Write and verify 16 data patterns in all used bits of all high bytes of all
6321      ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
6322      ; writing each pattern, set all the used bits of all active registers.
6323      ; REGTST routine reports errors with numbers >>>> ERROR 1912 - 1914 <<<<.
6324 032420 012737 003570 003770      MOV    #1912.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
6325 032426 005403                NEG    R3         ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6326 032430 004737 021220      JSR    PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6327
6328      ;+
6329      ; Print error summary reports if necessary.
6330      ; The following routine reports errors with number >>>> ERROR # 1915 <<<<
6331 032434 012737 003573 003770      MOV    #1915.,ERRNBR ;SET UP ERROR NUMBER FOR NEXT RTN.
6332 032442 004737 021430      JSR    PC,REPSMR ;REPORT ERROR SUMMARY IF NECESSARY.
6333 032446 005037 002352      CLR    CTRLCF    ;INDICATE THAT WE COMPLETED THE TEST.
6334 032452
        032452
        032452 104401

```

```

L10047: TRAP C$ETST

```

HARDWARE TEST

- IDBIT -

```

6336
6337
6338
6339
6340
6341
6342
6343
6344 032454
      032454
6345      000023
6346 032454 012737 000023 002326
6347 032462 012737 177777 002352
6348 032470 012737 000001 003766
6349 032476 012737 003721 003770
6350 032504 012737 012713 003772
6351
6352
6353
6354
6355
6356 032512 004737 021456
6357 032516 103016
6358
6359
6360
6361 032520 017701 147526
6362 032524 032701 000400
6363 032530 001411
6364 032532 012737 003722 003770
6365 032540 012701 012755
6366 032544 012737 016470 003774
6367 032552
      032552 104460
6368 032554 005037 002352
6369 032560
      032560
      032560 104401
    
```

```

.SBTTL HARDWARE TEST      - IDBIT -
;+ *****
;*      - Device Register ID Bit Test -
;*
;*      This test verifies that the DUT STAT register ID bit reads as clear.
;*
;-- *****

      BGNTST
      T19::
      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV  #TNUM,TSTNUM    ;SET UP THE TEST NUMBER. (20)
      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
      MOV  #1,ERRTYP       ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
      MOV  #2001.,ERRNBR   ;SET UP ERROR NUMBER IN THE ERROR TABLE.
      MOV  #EM2001,ERRMSG  ;SET UP ERROR MESSAGE FOR TEST IN ERROR 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 >>>> 2001 <<<<.
;-
      JSR  PC,RESETT       ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCC  60$             ;FATAL RESET ERROR? YES, EXIT THE TEST.
;+
; Read the STAT register ID bit and verify that it is clear.
;-
      MOV  @STAT,R1        ;READ THE STAT REGISTER CONTENTS.
      BIT  #BIT8,R1        ;CHECK THE ID BIT.
      BEQ  60$             ;ID BIT CLEAR? YES, EXIT THE TEST.
      MOV  #2002.,ERRNBR   ;NO, SET THE ERROR REPORT NUMBER TO 2002.
      MOV  #EM2002,R1      ;GET THE PROPER ERROR MESSAGE.
      MOV  #ER0503,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
      ERROR                ;ERROR NUMBER >>>> 2002 <<<<
      TRAP C$ERROR
60$:  CLR  CTRLCF          ;INDICATE THAT WE COMPLETED THE TEST.
      ENDTST
      L10050:
      TRAP C$ETST
    
```

HARDWARE TEST

- NOTXDV -

```

6371
6372
6373
6374
6375
6376
6377
6378
6379
6380 032562
      032562
6381      000024
6382 032562 012737 000024 002326
6383 032570 012737 177777 002352
6384 032576 012737 000001 003766
6385 032604 012737 004065 003770
6386 032612 012737 013030 003772
6387 032620 012737 017422 003774
6388
6389
6390
6391
6392
6393 032626 004737 020156
6394 032632 103054
6395 032634 005237 003770
6396
6397
6398
6399
6400
6401
6402 032640 013705 002236
6403 032644 012700 000200
6404 032650 004737 023516
6405 032654 012700 177670
6406 032660 004737 023572
6407 032664 012704 000012
6408 032670 004737 020300
6409 032674 004737 022570
6410
6411
6412
6413
6414
6415 032700 013705 002236
6416 032704 005004
6417 032706 000241
6418 032710 006005
6419 032712 103020
6420
6421
6422
6423
6424
6425 032714 010477 147324
6426 032720 012777 000012 147320
    
```

```

.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::
; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
; SET UP THE TEST NUMBER.          (21)
; INDICATE THAT WE ARE IN A TEST.
; SET ERROR TYPE AS FATAL IN ERROR TABLE.
; SET THE FIRST ERROR NUMBER IN ERROR TABLE.
; SET ERROR MESSAGE ADDRESS IN ERRTBL.
; SELECT THE CORRECT ERROR REPORTING ROUTINE.
      TNUM == TNUM + 1
      MOV    #TNUM,TSTNUM
      MOV    #-1,CTRLCF
      MOV    #1,ERRTYP
      MOV    #2101,ERRNBR
      MOV    #EM2101,ERRMSG
      MOV    #ER9101,ERRBLK
;+
; 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
      BCC    60$
      INC    ERRNBR
;+
; 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
      MOV    #200,R0
      JSR    PC,WTWLNLC
      MOV    #177670,R0
      JSR    PC,WTWLPR
      MOV    #10.,R4
      JSR    PC,DELAY
      JSR    PC,TXDSBL
;+
; 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
      CLR    R4
2$:   CLC
      ROR    R5
      BCC    4$
;+
; 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
      MOV    #12,@TXCHA
    
```

HARDWARE TEST

- NOTXDV -

```

6427
6428
6429
6430
6431 032726 012701 170002
6432 032732 013702 002244
6433 032736 004737 023246
6434 032742 103004
6435
6436 032744 010401
6437 032746 012702 013073
6438
6439 032752
032752 104460
6440
6441
6442
6443 032754 005204
6444 032756 005705
6445 032760 001352
6446 032762 000400
6447
6448 032764 005037 002352
6449 032770
032770
032770 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 -

```

6451
6452
6453
6454
6455
6456
6457
6458
6459
6460
6461
6462 032772
      032772
6463
6464 032772 000025
6465 033000 012737 000025 002326
6466 033006 012737 177777 002352
6467 033014 012737 000001 003766
6468 033022 012737 004231 003770
6469 033030 012737 013167 003772
6470
6471
6472
6473
6474
6475 033036 004737 020156
6476 033042 103066
6477
6478
6479
6480
6481
6482
6483 033044 013705 002236
6484 033050 012700 000200
6485 033054 004737 023516
6486 033060 012700 177670
6487 033064 004737 023572
6488 033070 012704 000012
6489 033074 004737 020300
6490 033100 004737 022570
6491
6492
6493
6494
6495
6496 033104 013705 002236
6497 033110 005004
6498 033112 012737 004232 003770 2:
6499 033120 000241
6500 033122 006005
6501 033124 103032
6502
6503
6504
6505
6506
    
```

```

.SBTTL  HARDWARE TEST          - TXDVAL -
;+ *****
;*          - TX_DATA_VALID/TX_ACTION Test -
;* This test verifies that if a data word is written to the TXCHAR register
;* with the TX_DATA_VALID bit set, a corresponding 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

      T21::
      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV  #TNUM,TSTNUM        ;SET UP THE TEST NUMBER. (22)
      MOV  #-1,CTRLCF          ;INDICATE THAT WE ARE IN A TEST.
      MOV  #1,ERRTYP           ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV  #2201.,ERRNBR       ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
      MOV  #EM2201,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 >>>> 2201 <<<<.
;-
      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 active lines.
;-
      MOV  ACTLNS,R5           ;PASS THE ACTIVE LINE BIT MAP.
      MOV  #200,R0             ;PASS THE LNCTRL CONTENTS.
      JSR  PC,WTWLNCR          ;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.
      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 set.
; Verify that a corresponding TX_ACTION is generated.
;-
      MOV  ACTLNS,R5           ;GET THE ACTIVE LINE BIT MAP.
      CLR  R4                  ;CLEAR THE LINE NUMBER COUNTER.
      MOV  #2202.,ERRNBR      ;SET THE ERROR NUMBER TO 2202.
      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 data word (ASCII <LF>) to TXCHR register with the most significant
; bit (TX_DATA_VALID) set.
;-
    
```


HARDWARE TEST

- TXDVAL -

```

6507 033126 010477 147112      MOV    R4,@CSRA      ;SELECT THE LINE CURRENTLY UNDER TEST.
6508 033132 012777 100012 147106  MOV    @100012,@TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
6509
6510
6511
6512
6513 033140 012701 170002      ;+
; Wait for a TX_ACTION to be returned, report error if no TX_ACTION
; found before time-out occurs.
6514 033144 013702 002244      ;-
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.
BCS    4$             ;SKIP ERROR REPORT IF TX-ACTION FOUND.
MOV    @EM2202,R2     ;PASS THE ERROR MESSAGE TO BE REPORTED.
6515 033150 004737 023246      ;"NO TX_ACT FOUND AFTER VALID DATA WORD TX'D".
6516 033154 103403
6517 033156 012702 013224      BR     6$             ;GO REPORT THE ERROR.
6518
6519 033162 000411
6520
6521
6522
6523 033164 005237 003770      ;+
; Verify TX_ACTION returned from correct line.
4$:   INC    ERRNBR      ;INCREMENT ERROR NUMBER TO 2103.
SWAB   R2              ;GET THE LINE NUMBER IN THE LOW BYTE.
BIC    @177760,R2     ;CLEAR THE UNWANTED BITS.
CMP    R2,R4          ;IS IT THE CORRECT LINE NUMBER?.
BEQ    8$             ;YES; SKIP THE ERROR REPORT.
MOV    @EM2203,R2     ;PASS THE ERROR MESSAGE TO BE REPORTED.
6524 033170 000302
6525 033172 042702 177760      ; "INCORRECT LINE # RETURNED WITH TX_ACT"
6526 033176 020204
6527 033200 001404
6528 033202 012702 013316      6$:   MOV    R4,R1
MOV    ERROR          ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
;
; >>>> ERROR <<<<<.
;                                     TRAP    C#ERROR
6529
6530 033206 010401
6531 033210
033210 104460
6532
6533
6534
6535
6536 033212 005204
6537 033214 005705
6538 033216 001335
6539
6540 033220 005037 002352      ;+
; Verify all active lines have been tested.
8$:   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.
6541 033224
033224
033224 104401
60$:  CLR    CTRLCF     ;INDICATE THAT WE ARE NOT WITHIN A TEST.
;
;                                     L10052:
;                                     TRAP    C#ETST

```

HARDWARE TEST

- TXENBI-

```

6543
6544
6545
6546
6547
6548
6549
6550
6551
6552 033226
      033226
6553      000026
6554 033226 012737 000026 002326
6555 033234 012737 177777 002352
6556 033242 012737 000001 003766
6557 033250 012737 004375 003770
6558 033256 012737 013423 003772
6559 033264 012737 017422 003774
6560
6561
6562
6563
6564
6565 033272 004737 020156
6566 033276 103110
6567
6568
6569
6570
6571
6572
6573 033300 013705 002236
6574 033304 012700 000200
6575 033310 004737 023516
6576 033314 012700 177670
6577 033320 004737 023572
6578 033324 012704 000012
6579 033330 004737 020300
6580 033334 012705 000377
6581 033340 004737 022664
6582
6583
6584
6585
6586 033344 012703 000001
6587 033350 005004
6588 033352 012737 004376 003770 20:
6589 033360 030337 002236
6590 033364 001447
6591
6592
6593
6594
6595
6596 033366 010305
6597 033370 004737 022570
6598 033374 010477 146644
    
```

```

.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::
6553      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6554      MOV    #TNUM,TSTNUM        ;SET UP THE TEST NUMBER.          (23)
6555      MOV    #-1,CTRLCF          ;INDICATE THAT WE ARE IN A TEST.
6556      MOV    #1,ERRTYP           ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
6557      MOV    #2301.,ERRNBR       ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
6558      MOV    #EM2301,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERR_TBL.
6559      MOV    #ER9101,ERRBLK      ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
6560
6561      ; Reset the DUT to a known state, remove the status codes from the fifo.
6562      ; Clear TX and RX interrupt enable bits in the CSR.
6563      ; This subroutine reports error >>>> 2301 <<<<<.
6564
6565      ;--      JSR    PC,CLNRST          ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6566      BCC    60#                  ;RESET FAILURE?, ABORT THIS TEST.
6567
6568      ;+
6569      ; Set internal loopback on all active lines.
6570      ; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
6571      ; 2 stop bits.
6572      ; Enable transmitters on all lines.
6573      ;--      MOV    ACTLNS,R5          ;PASS THE ACTIVE LINE BIT MAP.
6574      MOV    #200,R0              ;PASS THE LNCTRL CONTENTS.
6575      JSR    PC,WTWLNLC          ;INITIALISE THE LNCTRL REGISTERS.
6576      MOV    #177670,R0         ;PASS THE LPR CONTENTS.
6577      JSR    PC,WTWLPR          ;INITIALISE THE LPR REGISTERS ON ALL LINES.
6578      MOV    #10.,R4            ;PASS DELAY TIME OF 10 MILLI-SECONDS.
6579      JSR    PC,DELAY           ;WAIT FOR LNCTR AND LPR REGS TO BE UPDATED.
6580      MOV    #MAPLNS,R5         ;PASS THE BIT MAP CORRESPONDING TO ALL LINES.
6581      JSR    PC,TXENBL          ;ENABLE TRANSMITTERS ON ALL LINES.
6582
6583      ;+
6584      ; Test all active lines individually.
6585      ; Disable transmission on each active line.
6586      ;--      MOV    #1,R3              ;SET UP THE LINE BIT MAP FOR CHANNEL 0.
6587      CLR    R4                  ;CLEAR THE LINE NUMBER COUNTER.
6588      MOV    #2302.,ERRNBR       ;SET THE ERROR NUMBER TO 2302.
6589      BIT    R3,ACTLNS           ;CHECK IF THE LINE IS ACTIVE.
6590      BEQ    6#                  ;SKIP TESTING THIS LINE IF IT IS INACTIVE.
6591
6592      ;+
6593      ; Clear the TX_ENABLE bit in TBUFAD2 register.
6594      ; Select the line under test.
6595      ; Verify it is clear, report error if set.
6596      ;--      MOV    R3,R5              ;PASS THE BIT MAP OF THE LINE UNDER TEST.
6597      JSR    PC,TXDSBL           ;DISABLE TRANSMISSION ON THE LINE UNDER TEST.
6598      MOV    R4,BCSRA            ;SELECT THE LINE CURRENTLY UNDER TEST.
    
```

HARDWARE TEST

- TXENBI -

```

6599 033400 005777 146654      TST    @TXAD2A      ;VERIFY THE TX_ENABLE BIT IS SET.
6600 033404 100433      BMI    4$          ;GO REPORT ERROR IF TX_ENABLE BIT SET.
6601
6602
6603      ; Write data word (ASCII <LF>) to TXCHR register.
6604      ; Wait for a TX_ACTION to be returned, report error if a TX_ACTION
6605      ; is found before time-out occurs.
6606 033406 012737 004377 003770      MOV    @2303.,ERRNBR ;SET ERROR NUMBER TO 2303.
6607 033414 012777 100012 146624      MOV    @100012,@TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
6608 033422 012701 170002      MOV    @170002,R1    ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
6609 033426 013702 002244      MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6610 033432 004737 023246      JSR    PC,WAIBIS    ;WAIT FOR TX_ACTION TO COME BACK.
6611 033436 103016      BCC    4$          ;GO REPORT ERROR IF NO TX-ACTION FOUND.
6612
6613      ; Wait for the data to appear in the fifo, report error if data found.
6614
6615 033440 005237 003770      INC    ERRNBR      ;SET ERROR NUMBER TO 2304.
6616 033444 012701 070012      MOV    @70012,R1    ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
6617 033450 013702 002244      MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6618 033454 004737 023246      JSR    PC,WAIBIS    ;WAIT FOR RX_DATA_AVAILABLE TO SET.
6619 033460 103405      BCS    4$          ;REPORT ERROR IF DATA RECEIVED IN THE FIFO.
6620 033462 005237 003770      INC    ERRNBR      ;SET ERROR NUMBER TO 2305.
6621 033466 017702 146554      MOV    @RBUFA,R2    ;READ THE DATA FROM THE FIFO.
6622 033472 100004      BPL    6$          ;SKIP ERROR REPORT IF DATA IS THERE.
6623
6624 033474 010401      4$:    MOV    R4,R1      ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
6625 033476 012702 013461      MOV    @EM2302,R2    ;PASS THE MESSAGE TO BE REPORTED.
6626
6627 033502      ERROR      ; "TX_ENABLE BIT BAD ON LINE: nn".
6628 033502 104460      ; >>>> ERROR <<<<<.
6629
6630      TRAP    C$ERROR
6631      ;+
6632      ; Verify all active lines have been tested.
6633
6634      ;-
6635 6$:    CLC          ;CLEAR THE CARRY BIT PRIOR TO ROTATION.
6636      ROL    R3      ;SHIFT THE BIT MAP FOR THE NEXT LINE.
6637      INC    R4      ;INCREMENT THE LINE NUMBER COUNTER.
6638 033504 000241      CMP    R4,@NUMLNS    ;HAVE ALL THE LINES BEEN TESTED?.
6639 033506 006103      BLT    2$          ;NO; BRANCH TO TEST THE NEXT LINE.
6640 033510 005204
6641 033512 020427 000010
6642 033516 002715
6643
6644 60$:   CLR    CTRLCF    ;INDICATE THAT WE ARE NOT WITHIN A TEST.
6645 033524      ENDTST
6646 033524 104401      L10053:
6647
6648      TRAP    C$ETST
    
```

HARDWARE TEST

- TXENBA-

```

6640
6641
6642
6643
6644
6645
6646
6647
6648
6649 033526
      033526
6650
6651 033526 000027
6652 033534 012737 000027 002326
6653 033542 012737 177777 002352
6654 033550 012737 000001 003766
6655 033556 012737 004541 003770
6656 033564 012737 013517 003772
6657
6658
6659
6660
6661
6662 033572 004737 020156
6663 033576 103127
6664
6665
6666
6667
6668
6669
6670 033600 013705 002236
6671 033604 012700 000200
6672 033610 004737 023516
6673 033614 012700 177670
6674 033620 004737 023572
6675 033624 012704 000012
6676 033630 004737 020300
6677 033634 012705 000377
6678 033640 004737 022570
6679
6680
6681
6682
6683 033644 012703 000001
6684 033650 005004
6685 033652 012737 004542 003770 2$:
6686 033660 030337 002236
6687 033664 001463
6688
6689
6690
6691
6692
6693 033666 010305
6694 033670 004737 022664
6695 033674 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,WTWLNLC         ;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 THIS 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-

```

6696 033700 010477 146340      MOV    R4,@CSRA      ;SELECT THE LINE CURRENTLY UNDER TEST.
6697 033704 005777 146350      TST    @TXAD2A      ;VERIFY THE TX_ENABLE BIT IS SET.
6698 033710 100045                BPL    6$           ;GO REPORT ERROR IF TX_ENABLE BIT CLEAR.
6699
6700
6701
6702
6703
6704 033712 012737 004543 003770 4$:
6705 033720 012777 100012 146320      MOV    @2403.,ERRNBR ;SET ERROR NUMBER TO 2403.
6706 033726 012701 170002                MOV    @100012,@TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
6707 033732 013702 002244                MOV    @170002,R1    ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
6708 033736 004737 023246                MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6709 033742 103030                JSR    PC,WAIBIS    ;WAIT FOR TX_ACTION TO COME BACK.
6710
6711
6712
6713 033744 005237 003770                BCC    6$           ;GO REPORT ERROR IF NO TX-ACTION FOUND.
6714 033750 012701 070012                ;+
6715 033754 013702 002244                ; Wait for the data to appear in the fifo, report error if time-out.
6716 033760 004737 023246                ;-
6717 033764 103017                INC    ERRNBR      ;SET ERROR NUMBER TO 2404.
6718 033766 005237 003770                MOV    @70012,R1   ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
6719 033772 017702 146250                MOV    CSRA,R2    ;PASS THE ADDRESS OF THE REGISTER TO TEST.
6720 033776 100012                JSR    PC,WAIBIS  ;WAIT FOR RX_DATA_AVAILABLE TO SET.
6721 034000 005237 003770                BCC    6$           ;REPORT ERROR IF NO DATA RECEIVED IN THE FIFO.
6722 034004 000302                INC    ERRNBR      ;SET ERROR NUMBER TO 2405.
6723 034006 042702 177760                MOV    @RBUFA,R2  ;READ THE DATA FROM THE FIFO.
6724 034012 020204                BPL    6$           ;GO REPORT ERROR IF THER IS'NT ANY DATA THERE.
6725 034014 001003                INC    ERRNBR      ;SET ERROR NUMBER TO 2406.
6726 034016 005305                SWAB   R2          ;PUT THE LINE NUMBER IN THE LOW BYTE.
6727 034020 001334                BIC   @177760,R2  ;CLEAR THE UNWANTED BITS.
6728 034022 000404                CMP   R2,R4       ;DID THE DATA COME FROM THE CORRECT LINE?.
6729
6730 034024 010401 013461 6$:                BNE   6$           ;NO; GO REPORT THE ERROR.
6731 034026 012702 013461                DEC   R5           ;DECREMENT THE TXCHAR/LOOP COUNTER.
6732
6733 034032 104460                BNE   4$           ;LOOP TO TX THE NEXT CHAR.
6734
6735
6736
6737 034034 010305                BR    8$           ;GO TEST THE NEXT LINE.
6738 034036 004737 022570                MOV    R4,R1      ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
6739 034042 000241                MOV    @EM2302,R2 ;PASS THE MESSAGE TO BE REPORTED.
6740 034044 006103                ERROR
6741 034046 005204                ; "TX_ENABLE BIT BAD ON LINE: nn".
6742 034050 020427 000010                ; >>>> ERROR <<<<<.
6743 034054 002676                TRAP   C$ERROR
6744
6745 034056 005037 002352                ;+
6746 034062 104401                ; Verify all active lines have been tested.
                                ;-
6747 034062 104401                8$:    MOV    R3,R5    ;PASS THE BIT MAP OF THE LINE UNDER TEST.
                                JSR    PC,TXDSBL  ;CLEAR THE TX_ENABLE BIT ON THIS LINE.
                                CLC                    ;CLEAR THE CARRY BIT PRIOR TO ROTATION.
                                ROL    R3              ;SHIFT THE BIT MAP FOR THE NEXT LINE.
                                INC    R4              ;INCREMENT THE LINE NUMBER COUNTER.
                                CMP    R4,@NUMLNS     ;HAVE ALL THE LINES BEEN TESTED?.
                                BLT    2$             ;NO; BRANCH TO TEST THE NEXT LINE.
6748
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6760
6761
6762
6763
6764
6765
6766
6767
6768
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784
6785
6786
6787
6788
6789
6790
6791
6792
6793
6794
6795
6796
6797
6798
6799
8000
8001
8002
8003
8004
8005
8006
8007
8008
8009
8010
8011
8012
8013
8014
8015
8016
8017
8018
8019
8020
8021
8022
8023
8024
8025
8026
8027
8028
8029
8030
8031
8032
8033
8034
8035
8036
8037
8038
8039
8040
8041
8042
8043
8044
8045
8046
8047
8048
8049
8050
8051
8052
8053
8054
8055
8056
8057
8058
8059
8060
8061
8062
8063
8064
8065
8066
8067
8068
8069
8070
8071
8072
8073
8074
8075
8076
8077
8078
8079
8080
8081
8082
8083
8084
8085
8086
8087
8088
8089
8090
8091
8092
8093
8094
8095
8096
8097
8098
8099
9000
9001
9002
9003
9004
9005
9006
9007
9008
9009
9010
9011
9012
9013
9014
9015
9016
9017
9018
9019
9020
9021
9022
9023
9024
9025
9026
9027
9028
9029
9030
9031
9032
9033
9034
9035
9036
9037
9038
9039
9040
9041
9042
9043
9044
9045
9046
9047
9048
9049
9050
9051
9052
9053
9054
9055
9056
9057
9058
9059
9060
9061
9062
9063
9064
9065
9066
9067
9068
9069
9070
9071
9072
9073
9074
9075
9076
9077
9078
9079
9080
9081
9082
9083
9084
9085
9086
9087
9088
9089
9090
9091
9092
9093
9094
9095
9096
9097
9098
9099

```

L10054: TRAP C\$ETST

HARDWARE TEST

- INTA -

```

6748
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6760 034064
      034064
6761      000030
6762 034064 012737 000030 002326
6763 034072 012737 177777 002352
6764 034100 012737 000001 003766
6765 034106 012737 003101 003770
6766 034114 012737 013553 003772
6767
6768
6769
6770
6771
6772 034122 004737 021456
6773 034126 103402
6774 034130 000137 035070
6775 034134 012737 005053 003770
6776
6777
6778
6779 034142 012705 000377
6780 034146 004737 022664
6781
6782
6783
6784
6785
6786
6787 034152 005037 002334
6788 034156 005037 002336
6789 034162 005037 002340
6790 034166 012737 002726 002324
6791 034174 106427 000240
6792 034200
      034200 012746 000240
      034204 012746 024036
      034210 013746 002232
      034214 012746 000003
      034220 104437
      034222 062706 000010
6793 034226
      034226 012746 000240
      034232 012746 023650
      034236 013746 002234
      034242 012746 000003
    
```

.SBTTL HARDWARE TEST - INTA -

```

;+ *****
;*
;* - Interrupt Test -
;* This test verifies that the Device Under Test (DUT) will generate
;* reception and transmission interrupts correctly. This test does
;* not depend on the use of the serial line transmission or reception
;* capabilities of the DUT. The lines are put in internal loopback
;* to minimize any external effects that could be caused on devices
;* attached to the serial lines.
;*
;-- *****
    
```

BGNTST

```

T24::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (26)
      MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
      MOV #1,ERRTYP ;SET ERROR FATAL ERROR TYPE IN ERROR TABLE.
      MOV #1601.,ERRNBR ;SET FIRST ERROR REPORT NUMBER IN ERROR TABLE.
      MOV #EM2601.,ERRMSG ;SET TEST ERROR MESSAGE IN ERROR 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 from >>>> 2601 thru 2602 <<<<.
;-
      JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCS 2$ ;SKIP AROUND ABORTING TEST IF NO ERROR FOUND.
      JMP 60$ ;ABORT TEST IF FATAL ERROR FOUND DURING RESET.
2$: MOV #2603.,ERRNBR ;SET THE ERROR REPORT NUMBER TO 2603.
;+
; Enable transmitters on all lines.
;-
4$: MOV #MAPLNS,R5 ;PASS ACTIVE LINE BIT MAP.
      JSR PC,TXENBL ;ENABLE TRANSMISSION ON ALL LINES.
;+
; Test reception interrupts.
; Set up for RX and TX interrupts:
; RX interrupt service routine inputs a char and counts the interrupt.
; TX interrupt service routine counts TX interrupts.
;-
      CLR RXINTC ;CLEAR THE RX INTERRUPT COUNTER.
      CLR RXINTF ;CLEAR THE RX INTERRUPT FLAGS.
      CLR TXINTC ;CLEAR THE TX INTERRUPT COUNTER.
      MOV #BUFBAS,BUFPTR ;LOAD THE BUFFER PTR WITH THE BUFFER BASE ADR.
      MTPS #PRI05 ;DISABLE DEVICE INTERRUPTS.
      SETVEC RXVECA,#RXINPT,#PRI05 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
      MOV #PRI05,-(SP)
      MOV #RXINPT,-(SP)
      MOV #RXVECA,-(SP)
      MOV #3,-(SP)
      TRAP C$SVEC
      ADD #10,SP
      SETVEC TXVECA,#CACHTX,#PRI05 ;SET UP INTERRUPT VECTOR TO CATCH TX INT.
      MOV #PRI05,-(SP)
      MOV #CACHTX,-(SP)
      MOV #TXVECA,-(SP)
      MOV #3,-(SP)
    
```

HARDWARE TEST

- INTA -

6794	034254	106427	000000						
6795									
6796									
6797									
6798									
6799									
6800	034260	004737	022236						
6801	034264	012704	000004						
6802	034270	004737	020300						
6803	034274	004737	022204						
6804									
6805									
6806									
6807									
6808	034300	005737	002334						
6809	034304	001017							
6810									
6811									
6812									
6813	034306	012701	013762						
6814	034312	032777	000200	145724					
6815	034320	001416							
6816	034322	012701	013674						
6817	034326	032777	100000	145712					
6818	034334	001410							
6819	034336	012701	013603						
6820	034342	000405							
6821									
6822									
6823									
6824	034344	005737	002336						
6825	034350	100006							
6826	034352	012701	014056						
6827									
6828									
6829									
6830	034356								
	034356	104455							
	034360	005053							
	034362	013553							
	034364	016470							
6831									
6832									
6833									
6834	034366	013702	002340						
6835	034372	001406							
6836									
6837	034374	012701	010003						
6838	034400								
	034400	104455							
	034402	005054							
	034404	013553							
	034406	016514							
6839									
6840									

```

MTPS    #PRI00          ;ALLOW INTERRUPTS.
;+
;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 CLEAR.
MOV     #EM2602,R1      ;SET UP MSG, "DATA.VALID IS SET".
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$:     ERRDF   2603,EM2601,ER0503;          >>>> ERROR #2603 <<<<<.
;+
; Verify that no TX interrupts have been generated so far in this test.
;-
10$:    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 <<<<<.
;+
; Clean out the interrupt vectors used in this test.

```

HARDWARE TEST

- INTA -

```

6841
6842 034410 106427 000240      12$:  MTPS  #PRI05      ;DISABLE DEVICE INTERRUPTS.
6843 034414      013700 002232      CLRVEC  RXVECA      ;RETURN RX INT VECTOR TO UNUSED POOL.
      034414      013700 002232      MOV      RXVECA,R0
      034420      104436      TRAP     C$CVEC
6844 034422      CLRVEC  TXVECA      ;RETURN TX INT VECTOR TO UNUSED POOL.
      034422      013700 002234      MOV      TXVECA,R0
      034426      104436      TRAP     C$CVEC
6845
6846      ;+
6847      ; Test transmission interrupts.
6848      ; Set up for RX and TX interrupts:
6849      ; RX interrupt service routine counts RX interrupts.
6850      ; TX interrupt service routine counts the interrupt and sets flags.
6851 034430 005037 002334      CLR      RXINTC      ;CLEAR THE RX INTERRUPT COUNTER.
6852 034434 005037 002340      CLR      TXINTC      ;CLEAR THE TX INTERRUPT COUNTER.
6853 034440 005037 002342      CLR      TXINTF      ;CLEAR THE RX INTERRUPT FLAGS.
6854 034444      SETVEC  RXVECA,#CACHRX,#PRI05 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
      034444      012746 000240      MOV      #PRI05,-(SP)
      034450      012746 023622      MOV      #CACHRX,-(SP)
      034454      013746 002232      MOV      RXVECA,-(SP)
      034460      012746 000003      MOV      #3,-(SP)
      034464      104437      TRAP     C$SVEC
      034466      062706 000010      ADD      #10,SP
6855 034472      SETVEC  TXVECA,#TXINTR,#PRI05 ;SET UP INT VECTOR TO TX INT ROUTINE.
      034472      012746 000240      MOV      #PRI05,-(SP)
      034476      012746 024144      MOV      #TXINTR,-(SP)
      034502      013746 002234      MOV      TXVECA,-(SP)
      034506      012746 000003      MOV      #3,-(SP)
      034512      104437      TRAP     C$SVEC
      034514      062706 000010      ADD      #10,SP
6856 034520 106427 000000      MTPS  #PRI00      ;ALLOW INTERRUPTS.
6857
6858      ;+
6859      ; Verify that the TX_ACTION bit is clear.
6860 034524 012705 000022      ;-
6861 034530 012701 000144      MOV      #18.,R5      ;INITIALIZE THE LOOP COUNTER.
6862 034534 012702 100000      MOV      #100.,R1     ;SET 100 MS TIME-OUT.
6863 034540 013704 002244      MOV      #BIT15,R2    ;SELECT TX_ACTION BIT TO TEST.
6864 034544 012703 100000      MOV      CSRA,R4      ;PASS OUT CSR AS THE WORD TO TEST.
6865 034550 004737 020454      14$:  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.
6866 034554 103020      CLR      R3           ;NOW, WAIT FOR TX_ACTION CLEAR.
6867 034556 005003      JSR      PC,MSLOOP  ;WAIT UP TO 100 MS FOR TX_ACTION CLEAR.
6868 034560 004737 020454      BCC     16$         ;IF TIME-OUT, REPORT TX_ACTION WON'T CLEAR.
6869 034564 103005      DEC     R5           ;DECREMENT THE TX_ACTION SET COUNTER.
6870 034566 005305      BNE     14$         ;LOOP IF NOT TOO MANY TX_ACTIONS FOUND.
6871 034570 001365      ;REPORT "TX_ACTION SET REPEATEDLY AFTER RESET, NO DATA SENT."
6872      MOV      #EM2607,R1 ;SELECT ERROR MESSAGE.
6873 034572 012701 014200      BR      18$         ;GO TO REPORT THE ERROR.
6874 034576 000402      16$:  MOV      #EM2608,R1 ;SELECT TX_ACTION STUCK SET MSG.
6875 034600 012701 014274      18$:  ERRDF  2605,EM2606,ER0503; >>>> ERROR #2605 <<<<.
6876 034604      TRAP     C$ERDF
      034604      104455      .WORD  2605
      034606      005055      .WORD  EM2606
      034610      014141      .WORD  ER0503
      034612      016470
6877 034614 000424      BR      24$         ;GO TO TEST WITH TX_ACTION SET.

```


HARDWARE TEST

- INTA -

```

6878
6879
6880
6881 034616 004737 023012
6882 034622 012704 000062
6883 034626 004737 020300
6884 034632 005737 002340
6885 034636 001413
6886 034640 012701 014200
6887 034644 005737 002342
6888 034650 100002
6889 034652 012701 014345
6890
6891 034656
034656 104455
034660 005056
034662 014141
034664 016470
6892
6893
6894
6895 034666 005037 002340
6896 034672 005037 002342
6897
6898
6899
6900 034676 012705 000377
6901 034702 012700 000200
6902 034706 004737 023516
6903 034712 012700 156430
6904 034716 004737 023572
6905
6906
6907
6908 034722 013701 002246
6909 034726 012702 100000
6910 034732 004737 023546
6911
6912
6913
6914 034736 012704 000372
6915 034742 004737 020300
6916
6917
6918
6919 034746 005737 002340
6920 034752 001007
6921
6922
6923
6924 034754 012701 014424
6925 034760 005777 145260
6926 034764 100407
6927 034766 012701 014516
6928
6929
6930

```

```

;+
; Verify that no interrupts occur with TX_ACTION clear.
;-
20$: JSR PC,TXIE1 ;ENABLE TX_INTERRUPTS.
MOV #50.,R4 ;PASS 50 MS TIME TO THE DELAY ROUTINE.
JSR PC,DELAY ;DELAY 50 MILLI-SECONDS TO ALLOW INTS TO OCCUR.
TST TXINTC ;TEST THE TX INTERRUPT COUNT.
BEQ 24$ ;SKIP THE ERROR IF NO TX INTERRUPTS.
MOV #EM2607,R1 ;SELECT MESSAGE IN CASE TX INT FLAG CLEAR.
TST TXINTF ;TEST THE TX INTERRUPT FLAGS.
BPL 22$ ;GO REPORT ERROR IF TX FLAG IS CLEAR.
MOV #EM2609,R1 ;TX FLAG IS SET, SELECT PROPER ERROR MESSAGE.
;REPORT "TRANSMIT INTERRUPT TEST ERROR:..."
22$: ERRDF 2606,EM2606,ER0503; >>>> ERROR #2606 <<<<.
TRAP C$ERDF
.WORD 2606
.WORD EM2606
.WORD ER0503
;+
; Prepare TX interrupt counter and flags.
;-
24$: CLR TXINTC ;CLEAR THE TX INTERRUPT COUNT.
CLR TXINTF ;CLEAR THE TX INTERRUPT FLAGS.
;+
; Set up line parameters for transmission.
;-
MOV #MAPLNS,R5 ;PASS ACTIVE LINES BIT MAP.
MOV #200,R0 ;PASS INERT STATE, INTERNAL LOOPBACK.
JSR PC,WTWLN ;DISABLE RECPTION AND DMA, ETC. ON DUT.
MOV #156430,R0 ;SPECIFY 9600BPS,1STOP,NO PARITY,8BITS/CHAR.
JSR PC,WTWLP ;WRITE TO ALL LPR REGISTERS.
;+
; Send a null char to each line.
;-
MOV TXCHA,R1 ;SET UP TXCHAR REGISTER ADDRESS.
MOV #100000,R2 ;SET CHARACTER TO BE TRANSMITTED = NULL.
JSR PC,WTWLN ;SEND NULL CHAR TO EACH LINE.
;+
; Delay 250 milli-seconds to allow interrupts to occur.
;-
MOV #250.,R4 ;SET UP FOR 250 MS DELAY.
JSR PC,DELAY ;WAIT 250 MS.
;+
; Verify that TX interrupts occurred.
;-
TST TXINTC ;CHECK THE TX INTERRUPT COUNTER.
BNE 26$ ;SKIP THE FOLLOWING ERROR IF WE GOT TX INTS.
;+
; Determine the reason that we received no interrupts.
;-
MOV #EM2610,R1 ;SET UP MSG IN CASE "TX_ACTION IS SET".
TST @CSRA ;CHECK THE DUT CSR.
BMI 28$ ;GO TO REPORT ERROR IF TX_ACTION IS SET.
MOV #EM2611,R1 ;SET UP "TX_ACTION NOT SET" MESSAGE.
;+
; Check to verify that TX_ACTION was set for each interrupt.
;-

```

HARDWARE TEST

- INTA -

```

6931 034772 005737 002342      26$:   TST   TXINTF      ;CHECK THE TX INTERRUPT FLAGS.
6932 034776 100006              BPL   30$             ;SKIP ERROR IF TX_ACTION CLR FLAG IS CLEAR.
6933 035000 012701 014345      MOV   #EM2609,R1     ;SET UP TX INT WITH "TX_ACTION CLR" MSG.
6934
6935      ;+
6936      ; Report "TRANSMIT INTERRUPT TEST ERROR:...."
6937 035004              28$:   ERRDF  2607,EM2606,ER0503;      >>>> ERROR #2607 <<<<<.
        035004 104455              TRAP   C$ERDF
        035006 005057              .WORD  2607
        035010 014141              .WORD  EM2606
        035012 016470              .WORD  ER0503
6938
6939      ;+
6940      ; Verify that no TX interrupts have been generated so far in this test.
6941 035014 013702 002334      30$:   MOV   RXINTC,R2      ;LOAD # OF RX INTERRUPTS FOR ER0504 RTN.
6942 035020 001406              BEQ   32$             ;SKIP ERROR IF NO RX INTERRUPTS.
6943 035022 012701 007713      MOV   #EM0525,R1     ;SET UP MESSAGE ADR FOR INDIRECT PRINT.
6944      ;REPORT "RX INTERRUPTS(S) RECEIVED WITH RX INTERRUPTS DISABLED."
6945 035026              ERRDF  2608,EM2606,ER0504;      >>>> ERROR #2608 <<<<<.
        035026 104455              TRAP   C$ERDF
        035030 005060              .WORD  2608
        035032 014141              .WORD  EM2606
        035034 016514              .WORD  ER0504
6946
6947      ;+
6948      ; Disable interrupts and clean out the interrupt vectors used in this test.
6949 035036 005001              32$:   CLR   R1           ;CLEAR BOTH TRANSMITTER
6950 035040 004737 022760      JSR   PC,TXIE0       ; INTERRUPT ENABLE AND RECEIVER
6951 035044 004737 022204      JSR   PC,RXIE0       ; INTERRUPT ENABLE BITS IN THE DUT CSR.
6952 035050 106427 000240      MTPS #PRI05         ;DISABLE DEVICE INTERRUPTS.
6953 035054              CLRVEC RXVECA        ;RETURN RX INT VECTOR TO UNUSED POOL.
        035054 013700 002232              MOV   RXVECA,RO
        035060 104436              TRAP  C$CVEC
6954 035062              CLRVEC TXVECA        ;RETURN TX INT VECTOR TO UNUSED POOL.
        035062 013700 002234              MOV   TXVECA,RO
        035066 104436              TRAP  C$CVEC
6955
6956 035070 005037 002352      60$:   CLR   CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
6957
6958 035074              ENDTST
        035074
        035074 104401              L10055: TRAP  C$ETST
6959

```

HARDWARE TEST

- BRLEVA -

```

6961 .SBTTL HARDWARE TEST - BRLEVA -
6962 ;+ *****
6963 ;* - BR Level Test B -
6964 ;* This test verifies that the Device Under Test (DUT) will generate
6965 ;* reception and transmission interrupts at the correct BR level.
6966 ;* This test does not depend on the use of the serial line transmission
6967 ;* or reception capabilities of the DUT. The lines are put in internal
6968 ;* loopback to minimize any external effects that could be caused on
6969 ;* devices attached to the serial lines.
6970 ;*
6971 ;-- *****
6972
6973 035076 BGNTST
        035076
6974
6975 035076 000031 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
6976 035104 012737 000031 002326 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (30)
6977 035112 012737 177777 002352 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
6978 035120 012737 000001 003766 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
6979 035126 012737 005671 003770 MOV #3001.,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
6980 035134 005037 002462 MOV #EM3001,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERTTBL.
6981 CLR ERSMRF ;INITIALIZE THE "REPORT ERROR SUMMARY" FLAGS.
6982 ;+
6983 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
6984 ; Clear TX and RX interrupt enable bits in the CSR.
6985 ; This subroutine reports errors from >>>> 3001 thru 3002 <<<<.
6986 ;-
6986 035140 004737 021456 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
6987 035144 103402 BCS 2$ ;SKIP AROUND ABORTING TEST IF NO ERROR FOUND.
6988 035146 000137 035736 JMP 60$ ;ABORT TEST IF FATAL ERROR FOUND DURING RESET.
6989 035152 012737 005673 003770 2$: MOV #3003.,ERRNBR ;SET THE ERROR REPORT NUMBER TO 3003.
6990 ;+
6991 ; Enable transmitters on all lines.
6992 ;-
6993 035160 012705 000377 4$: MOV #MAPLNS,R5 ;PASS ACTIVE LINE BIT MAP.
6994 035164 004737 022664 JSR PC,TXENBL ;ENABLE TRANSMISSION ON ALL LINES.
6995 ;+
6996 ; Generate a transmission interrupt request.
6997 ; Processor priority should be at 7 disabling ints.
6998 ;-
6999 035170 106427 000340 MTPS #PRI07 ;DISABLE ALL INTERRUPTS.
7000 035174 SETVEC TXVECA,#TXINTR,#PRI07 ;SET UP INTERRUPT VECTOR TO CATCH TX INT.
        035174 012746 000340 MOV #PRI07,-(SP)
        035200 012746 024144 MOV #TXINTR,-(SP)
        035204 013746 002234 MOV TXVECA,-(SP)
        035210 012746 000003 MOV #3,-(SP)
        035214 104437 TRAP C$SVEC
        035216 062706 000010 ADD #10,SP
7001 ;+
7002 ; Set up DUT for transmission interrupts:
7003 ; Set up internal loopback.
7004 ; Set up line parameters for transmission.
7005 ;-
7006 035222 012705 000377 MOV #MAPLNS,R5 ;PASS ACTIVE LINES BIT MASK.
7007 035226 012700 000200 MOV #200,R0 ;PASS INERT STATE, INTERNAL LOOPBACK.
7008 035232 004737 023516 JSR PC,WTWLNLC ;DISABLE RECEPTION AND DMA, ETC. ON DUT.
7009 035236 012700 156430 MOV #156430,R0 ;SPECIFY 9600BPS,1STOP,NO PARITY,8BITS/CHAR.
7010 035242 004737 023572 JSR PC,WTWLPRL ;WRITE INTO ALL LPR REGISTERS.
    
```

HARDWARE TEST

- BRLEVA -

```

7011
7012      ;+
7013      ; Send a null char to each line.
7014      ;-
7014 035246 013701 002246      MOV    TXCHA,R1      ;SET UP TXCHAR REGISTER ADDRESS.
7015 035252 012702 100000      MOV    #100000,R2    ;SET CHARACTER TO BE TRANSMITTED = NULL.
7016 035256 004737 023546      JSR    PC,WTWLN5     ;SEND NULL CHAR TO EACH LINE.
7017
7018      ;+
7018      ; Delay 50 ms to allow time for the interrupt to be generated.
7019      ;-
7020 035262 012704 000062      MOV    #50.,R4      ;PASS 50 MS TIME TO THE DELAY ROUTINE.
7021 035266 004737 020300      JSR    PC,DELAY     ;DELAY 50 MILLI-SECONDS.
7022
7023      ;+
7023      ; Generate a reception interrupt request.
7024      ;-
7025 035272      SETVEC  RXVECA,#RXBRRT,#PRI07 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
7025 035272 012746 000340      MOV    #PRI07,-(SP)
7025 035276 012746 023746      MOV    #RXBRRT,-(SP)
7025 035302 013746 002232      MOV    RXVECA,-(SP)
7025 035306 012746 000003      MOV    #3,-(SP)
7025 035312 104437      TRAP   C+SVEC
7025 035314 062706 000010      ADD    #10,SP
7026
7027      ;+
7027      ; Set up for the loop which tests the interrupt BR levels.
7028      ;-
7029 035320 012705 000340      MOV    #340,R5     ;SET UP THE PRIORITY LEVEL TO 7.
7030 035324 005003      CLR    R3          ;CLEAR THE RX PRIORITY STORE AND FLAGS.
7031 035326 005002      CLR    R2          ;CLEAR THE TX PRIORITY STORE AND FLAGS.
7032
7033      ;+
7033      ; Enable TX and RX interrupts.
7034      ; Processor priority should be at 7 disabling the interrupts.
7035      ;-
7036 035330 004737 022236      JSR    PC,RXIE1    ;ENABLE RECEIVER INTERRUPTS.
7037 035334 004737 023012      JSR    PC,TXIE1    ;ENABLE TRANSMITTER INTERRUPTS.
7038
7039      ;+
7039      ; Loop, lowering the processor priority until the DUT interrupts on RX and TX.
7040      ;-
7041 035340 005037 002340      6$:   CLR    TXINTC    ;CLEAR THE TX INTERRUPT COUNTER.
7042 035344 005037 002342      CLR    TXINTF    ;CLEAR THE TX INTERRUPT FLAGS.
7043 035350 005037 002334      CLR    RXINTC    ;CLEAR THE RX INTERRUPT COUNTER.
7044 035354 005037 002336      CLR    RXINTF    ;CLEAR THE RX INTERRUPT FLAGS.
7045 035360 106405      MTPS   R5         ;SET PROCESSOR PRIORITY TO THE SELECTED VALUE.
7046 035362 012704 000001      MOV    #1,R4     ;PASS 1 MS COUNT TO THE DELAY ROUTINE.
7047 035366 004737 020300      JSR    PC,DELAY   ;DELAY 1 MS TO ALLOW INTERRUPTS TO OCCUR.
7048
7049      ;+
7049      ; Determine if any RX DUT interrupts occurred.
7050      ; Log the processor priority for the RX interrupt if first RX int.
7051      ;-
7052 035372 005737 002334      TST    RXINTC     ;CHECK THE RECEIVE INTERRUPT COUNTER.
7053 035376 001412      BEQ    8$         ;SKIP THE PRIORITY LOG IF NO RX INT OCCURRED.
7054
7055      ;+
7055      ; If this is the first RX interrupt, log the priority.
7056      ;-
7057 035400 005703      TST    R3         ;CHECK THE RX PRIORITY STORE AND FLAGS.
7058 035402 001010      BNE    8$         ;GOTO TEST FOR TX INTS IF NOT THE FIRST RX INT.
7059 035404 010503      MOV    R5,R3     ;LOG THE PRESENT PRIORITY IN THE RX PRIO STORE.
7060 035406 052703 100000      BIS    #BIT15,R3 ;SET THE RX INT HAS OCCURRED FLAG.
7061 035412 013700 002336      MOV    RXINTF,R0 ;GET THE RX INTERRUPT ROUTINE FLAGS.

```

HARDWARE TEST

- BRLEVA -

```

7062 035416 042700 137777      BIC    #137777,R0      ;CLEAR ALL BUT THE TX INT ERROR FLAG.
7063 035422 050003      BIS    R0,R3          ;IF TX INT ERROR, SET BIT 14 OF THE PRIO FLAGS.
7064
7065      ;+
7066      ; Determine if any TX DUT interrupts have occurred.
7067      ; Log the present processor priority if this is the first TX interrupt.
7068 035424 005737 002340      8$:   TST    TXINTC      ;CHECK THE TRANSMIT INTERRUPT COUNTER.
7069 035430 001405      BEQ    10$           ;SKIP THE PRIORITY LOG IF NO TX INT OCCURRED.
7070
7071      ;+
7072      ; If this is the first TX interrupt, log the priority.
7073 035432 005702      ; -
7074 035434 100403      TST    R2            ;CHECK THE TX PRIORITY STORE AND FLAGS.
7075 035436 010502      BMI    10$           ;SKIP THE LOGGING IF NOT FIRST TX INTERRUPT.
7076 035440 052702 100000      MOV    R5,R2         ;LOG THE PRESENT PRIORITY IN THE TX PRIO STORE.
7077      BIS    #BIT15,R2      ;SET THE TX INT HAS OCCURRED FLAG.
7078      ;+
7079      ; Select next processor priority.
7080      ; Test for both RX and TX interrupts having occurred, loop if not.
7081 035444 162705 000040      10$:  SUB    #40,R5       ;DECREMENT PRIORITY LEVEL BY ONE.
7082 035450 002402      BLT    12$           ;GOTO CHECK FOR ERRORS IF BELOW PRIORITY ZERO.
7083 035452 030203      BIT    R2,R3         ;AND PRIO FLAGS TOGETHER, ALTER NONE OF THEM.
7084 035454 100331      BPL    6$            ;LOOP IF RX AND TX INTS HAVEN'T BOTH OCCURRED.
7085
7086      ;+
7087      ; Disable interrupts and clear interrupt vectors.
7088 035456 106427 000340      12$:  MTPS   #PRI07      ;DISABLE ALL INTERRUPTS.
7089 035462 035462 002232      CLRVEC RXVECA        ;RETURN RX INT VECTOR TO UNUSED POOL.
7090 035470 035466 104436      MOV    RXVECA,R0     ;
7091 035470 013700 002234      CLRVEC TXVECA        ;RETURN TX INT VECTOR TO UNUSED POOL.
7092 035474 104436      MOV    TXVECA,R0     ;
7093      TRAP   C#CVEC      ;
7094
7095      ;+
7096      ; Verify that RX and TX interrupts occurred,
7097      ; at the proper BR level, and
7098      ; in the proper order.
7099      ; Determine if TX interrupt occurred.
7100      ; -
7101      TST    R2            ;DETERMINE WHETHER TX INT OCCURRED OR NOT.
7102 035476 005702      BMI    16$           ;SKIP THESE ERRORS IF TX INT OCCURRED.
7103 035500 100414
7104
7105      ;+
7106      ; Determine reason that no TX int occurred.
7107      ; -
7108      MOV    #EM2610,R1    ;SELECT "NO TX INT FROM TX.ACTION" MESSAGE.
7109      TST    @CSRA         ;CHECK THE TX.ACTION BIT OF THE DUT CSR.
7110      BMI    14$           ;SKIP TX.ACTION CLR MSG SELECTION IF IT IS SET.
7111      MOV    #EM2611,R1    ;SELECT "TX.ACTION CLEAR AFTER CHARS SENT" MSG.
7112      ;REPORT "INTERRUPT BR LEVEL TEST ERROR:"
7113 14$:  ERRDF  3003,EM3001,ER0503;      >>>> ERROR #3003 <<<<<.
7114      TRAP   C#ERDF      ;
7115      .WORD  3003        ;
7116      .WORD  EM3001      ;
7117      .WORD  ER0503      ;
7118
7119      BR    18$           ;SKIP THE BR LEVEL CHECK, NO TX INT OCCURRED.
7120
7121      ;+
7122      ; Verify that the TX interrupt was at the proper BR level.

```

HARDWARE TEST

- BRLEVA -

```

7111
7112 035532 010204      16:  ;  MOV    R2,R4      ;CALCULATE THE BR LEVEL
7113 035534 042704      177400 ;  BIC    #177400,R4 ;  THAT THE TRANSMIT
7114 035540 006204      ;  ASR    R4         ;  INTERRUPT WAS
7115 035542 006204      ;  ASR    R4         ;  REQUESTED AT, WHICH
7116 035544 006204      ;  ASR    R4         ;  IS ONE GREATER THAN
7117 035546 006204      ;  ASR    R4         ;  THE PROCESSOR PRIORITY
7118 035550 006204      ;  ASR    R4         ;  LEVEL AT WHICH THE
7119 035552 005204      ;  INC    R4         ;  TRANSMIT INTERRUPT OCCURRED.
7120 035554 113705      002242 ;  MOVB   BRLEVL,R5  ;GET THE EXPECTED INTERRUPT BR LEVEL.
7121 035560 120405      ;  CMPB   R4,R5     ;COMARE THE INTERRUPT BR LEVEL WITH EXPECTED.
7122 035562 001406      ;  BEQ    18:       ;SKIP THE ERROR IF BR LEVEL IS CORRECT.
7123 ;REPORT "TX INTERRUPT GENERATED AT WRONG BR LEVEL: ..."
7124 035564 012701      014726 ;  MOV    #EM3003,R1 ;SELECT THE ERROR MESSAGE FOR THE ERROR CALL.
7125 035570      ;  ERRDF  3004,EM3001,ER3001;  >>>> ERROR #3004 <<<<<.
      035570 104455      ;  TRAP   C#ERDF
      035572 005674      ;  .WORD  3004
      035574 014601      ;  .WORD  EM3001
      035576 017070      ;  .WORD  ER3001

7126 ;+
7127 ; Determine if RX interrupt occurred.
7128 ;-
7129 035600 005703      18:  ;  TST    R3         ;CHECK THE RX INT OCCURRED FLAG.
7130 035602 100415      ;  BMI    22:       ;SKIP THESE ERRORS IF RX INT OCCURRED.
7131 ;+
7132 ; Determine reason that no RX int occurred.
7133 ;-
7134 035604 012701      013603 ;  MOV    #EM2602,R1 ;SELECT "NO RX INT FROM TX.ACTION" MSG.
7135 035610 032777      000200 144426 ;  BIT    #BIT7,#CSRA ;CHECK THE RX.DATA.AVAIL BIT OF THE DUT CSR.
7136 035616 001002      ;  BNE    20:       ;SKIP RX.DATA.AVAIL CLR MSG IF BIT IS SET.
7137 035620 012701      014632 ;  MOV    #EM3002,R1 ;SELECT "NO RX.DATA.AVAIL AFTER RESET" MSG.
7138 ;REPORT "INTERRUPT BR LEVEL TEST ERROR:"
7139 035624      20:  ;  ERRDF  3005,EM3001,ER0503;  >>>> ERROR #3005 <<<<<.
      035624 104455      ;  TRAP   C#ERDF
      035626 005675      ;  .WORD  3005
      035630 014601      ;  .WORD  EM3001
      035632 016470      ;  .WORD  ER0503

7140 035634 000423      ;  BR     24:       ;SKIP THE BR CHECK IF NO RX INT OCCURRED.
7141 ;+
7142 ; Verify that the RX interrupt was at the proper BR level.
7143 ;-
7144 035636 010304      22:  ;  MOV    R3,R4      ;CALCULATE THE BR LEVEL
7145 035640 042704      177400 ;  BIC    #177400,R4 ;  THAT THE RECEIVE
7146 035644 006204      ;  ASR    R4         ;  INTERRUPT WAS
7147 035646 006204      ;  ASR    R4         ;  REQUESTED AT, WHICH
7148 035650 006204      ;  ASR    R4         ;  IS ONE GREATER THAN
7149 035652 006204      ;  ASR    R4         ;  THE PROCESSOR PRIORITY
7150 035654 006204      ;  ASR    R4         ;  LEVEL AT WHICH THE
7151 035656 005204      ;  INC    R4         ;  RECEIVE INTERRUPT OCCURRED.
7152 035660 113705      002242 ;  MOVB   BRLEVL,R5  ;GET THE EXPECTED INTERRUPT BR LEVEL.
7153 035664 120405      ;  CMPB   R4,R5     ;COMARE THE INTERRUPT BR LEVEL WITH EXPECTED.
7154 035666 001406      ;  BEQ    24:       ;SKIP THE ERROR IF BR LEVEL IS CORRECT.
7155 ;REPORT "RX INTERRUPT GENERATED AT WRONG BR LEVEL: ..."
7156 035670 012701      015002 ;  MOV    #EM3004,R1 ;SELECT ERROR MESSAGE FOR THE ERROR CALL.
7157 035674      ;  ERRDF  3006,EM3001,ER3001;  >>>> ERROR #3006 <<<<<.
      035674 104455      ;  TRAP   C#ERDF
      035676 005676      ;  .WORD  3006

```

HARDWARE TEST - BRLEVA -

```

035700 014601
035702 017070                                .WORD  EM3001
                                                .WORD  ER3001
7158
7159
7160
7161
7162 035704 032703 040000
7163 035710 001406
7164
7165 035712 012701 015056
7166 035716
035716 104455
035720 005677
035722 014601
035724 016470

;+
; Test for interrupts occurring in the proper order.
;-
24: BIT    #BIT14,R3      ;CHECK THE IMPROPER INT ORDER ERROR FLAG.
    BEQ    26:           ;SKIP ERROR REPORT IF ERROR DID NOT OCCUR.
;REPORT "TX INTERRUPT GIVEN PRECEDENCE OVER SIMULTANEOUS RX INT."
    MOV    #EM3005,R1    ;SELECT THE ERROR MESSAGE FOR INDIRECT PRINT.
    ERDF   3007,EM3001,ER0503; >>>> ERROR #3007 <<<<.
                                TRAP  C#ERDF
                                .WORD 3007
                                .WORD EM3001
                                .WORD ER0503

;+
; Clean up, exit the test.
;-
26: JSR    PC,TXIEO      ;CLEAR TRANSMITTER INTERRUPTS.
    JSR    PC,RXIEO      ;CLEAR RECEIVER INTERRUPTS.
60: CLR    CTRLCF        ;INDICATE THAT WE ARE NOT WITHIN A TEST.
    MTPS  #PRI05         ;DISABLE ALL BUT LTC INTERRUPTS.
    ENDTST

                                L10056:
                                TRAP  C#ETST
7175

```

HARDWARE TEST

- DIABMP -

```

7177 .SBTTL HARDWARE TEST - DIABMP -
7178 ;* *****
7179 ;* - Diagnostic field (BMP) Test -
7180 ;* This test verifies that a request to the DUT to report BMP status
7181 ;* codes is complied with, within the specified time.
7182 ;* All active lines are tested.
7183 ;* *****
7184 ;-- *****
7185 035750 BGNTST
      035750
7186 000032 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
7187 035750 012737 000032 002326 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (31)
7188 035756 012737 177777 002352 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
7189 035764 012737 000001 003766 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
7190 035772 012737 006035 003770 MOV #3101.,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
7191 036000 012737 015150 003772 MOV #EM3101,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERTBL.
7192 036006 012737 017422 003774 MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
7193 ;*
7194 ; Reset the DUT to a known state, remove the status codes from the fifo.
7195 ; Clear TX and RX interrupt enable bits in the CSR.
7196 ; This subroutine reports error >>>> 3101 <<<<.
7197 ;-
7198 036014 004737 020156 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
7199 036020 103077 BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.
7200 ;*
7201 ; Test all active lines individually.
7202 ; Write the request code to the diagnostic field in the LPR register.
7203 ; Verify that a BMP code is returned within the correct time.
7204 ;-
7205 036022 013705 002236 MOV ACTLNS,R5 ;GET THE ACTIVE LINE BIT MAP.
7206 036026 005004 CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
7207 036030 013703 002244 MOV CSRA,R3 ;GET THE ADDRESS OF THE DUT'S CSR.
7208 036034 000241 2$: CLC ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
7209 036036 006005 ROR R5 ;SHIFT THE BIT MAP INTO THE CARRY BIT.
7210 036040 103064 BCC 8$ ;DO NOT TEST THE LINE IF IT IS INACTIVE.
7211 ;*
7212 ; Select the line under test.
7213 ; Write the BMP request code to the DIAG field in the LPR register.
7214 ;-
7215 036042 012737 006036 003770 MOV #3102.,ERRNBR ;SET THE ERROR NUMBER TO 3102.
7216 036050 010413 MOV R4,(R3) ;SELECT THE LINE CURRENTLY UNDER TEST.
7217 036052 052777 000002 144170 BIS #2,@LPRA ;WRITE THE BMP REQUEST CODE TO THE LPR.
7218 ;*
7219 ; Wait for BMP request code to be cleared, report error if time-out
7220 ; occurs.
7221 ;-
7222 036060 012701 010764 MOV #10764,R1 ;TEST BIT 1, TIMEOUT OF 500 MILLI SECS.
7223 036064 013702 002250 MOV LPRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.
7224 036070 004737 023172 JSR PC,WAIBIC ;WAIT FOR REQUEST CODE TO CLEAR.
7225 036074 103042 BCC 6$ ;GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.
7226 ;*
7227 ; Wait for BMP code to appear in the fifo, report error if time-out
7228 ; occurs.
7229 ;-
7230 036076 005237 003770 INC ERRNBR ;SET ERROR NUMBER TO 3103.
7231 036102 012701 070012 MOV #70012,R1 ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
7232 036106 013702 002244 MOV CSRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.

```


HARDWARE TEST

- DIABMP -

```

7233 036112 004737 023246      JSR    PC,WAIBIS      ;WAIT FOR RX_DATA_AVAILABLE TO SET.
7234 036116 103031              BCC    6$             ;GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.
7235                          ;*
7236                          ; Read the BMP code (if it is there) from the RBUF register.
7237                          ; Determine if it is a valid BMP code.
7238                          ; Verify the BMP code was received from the correct channel.
7239                          ; If the BMP code does not indicate DUT running ok, then save it on
7240                          ; the queue to be reported in a later test.
7241                          ;-
7242 036120 005237 003770      INC    ERRNBR         ;SET ERROR NUMBER TO 3104.
7243 036124 017702 144116      MOV    @RBUFA,R2     ;GET THE BMP CODE FROM THE FIFO.
7244 036130 100024              BPL    6$             ;GO REPORT ERROR IF NO BMP CODE FOUND.
7245 036132 005237 003770      INC    ERRNBR         ;SET ERROR NUMBER TO 3105.
7246 036136 012700 170301      MOV    @170301,R0    ;SET-UP A BMP CODE MASK.
7247 036142 040200              BIC    R2,R0         ;TRY TO CLEAR THE BMP MASK.
7248 036144 001016              BNE    6$             ;GO REPORT ERROR IF IT IS NOT A VALID BMP CODE.
7249 036146 005237 003770      INC    ERRNBR         ;SET THE ERROR NUMBER TO 3106.
7250 036152 010200              MOV    R2,R0         ;COPY THE BMP CODE.
7251 036154 000300              SWAB   R0            ;PUT THE LINE NUMBER IN THE LOW BYTE.
7252 036156 042700 177760      BIC    @177760,R0    ;CLEAR THE UNWANTED BITS.
7253 036162 120400              CMPB   R4,R0         ;DID THE BMP CODE COME FROM THE CORRECT LINE?.
7254 036164 001006              BNE    6$             ;NO; GO REPORT ERROR.
7255 036166 120227 000305      CMPB   R2,@305      ;IS THE BMP CODE A "GOOD ONE"?.
7256 036172 001407              BEQ    8$             ;YES; SKIP SAVING THE BMP CODE ON THE QUEUE.
7257 036174 004737 022262      JSR    PC,SAVBMP     ;SAVE THE BMP CODE ON THE QUEUE.
7258 036200 000404              BR     8$             ;GO SEE IF THERE ARE ANY MORE LINE TO TEST.
7259
7260 036202 010401              6$:    MOV    R4,R1     ;PASS THE LINE NUMBER TO BE REPORTED.
7261 036204 012702 015204      MOV    @EM3102,R2    ;PASS THE ERROR MESSAGE TO BE REPORTED.
7262
7263 036210              ERROR              ; "BMP REQUEST BIT BAD ON LINE:"
7264 036210 104460              ;          >>>> ERROR <<<<<.
7265                          ;          TRAP    C$ERROR
7266                          ;*
7267                          ; Verify all active lines have been tested.
7268                          ;-
7269 036212 005204              8$:    INC    R4         ;INCREMENT THE LINE NUMBER COUNTER.
7270 036214 005705              TST    R5             ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
7271 036216 001306              BNE    2$             ;YES; BRANCH TO TEST THE NEXT LINE.
7270 036220 005037 002352      60$:   CLR    CTRLCF     ;INDICATE THAT WE ARE NOT WITHIN A TEST.
7271 036224              ENDTST
7271 036224              L10057:          TRAP    C$ETST
036224 104401

```

HARDWARE TEST

- DIABMP -

```

7273
7274
7275
7276
7277
7278
7279
7280
7281
7282
7283
7284 036226
      036226
7285      000033
7286 036226 012737 000033 002326
7287 036234 012737 177777 002352
7288 036242 013702 002524
7289 036246 012703 002526
7290 036252 020203
7291 036254 001411
7292
7293
7294
7295
7296
7297 036256 012701 016013
7298 036262
      036262 104455
      036264 022125
      036266 015676
      036270 017450
7299
7300 036272 012737 002526 002524
7301
7302 036300 005037 002352
7303 036304
      036304
      036304 104401
    
```

```

.SBTTL HARDWARE TEST          - REPBMP -
;+ *****
;*          - 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::
      TNUM == TNUM + 1          ;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  #BMPQB,R2           ;GET THE CONTENTS OF THE POINTER.
      MOV  #BMPQB,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$ERDF
                                .WORD 9301
                                .WORD EM9301
                                .WORD ER9301
      MOV  #BMPQB,BMPQB        ;SET POINTER BACK TO THE BEGINING OF THE QUE.
60$:  CLR  CTRLCF              ;INDICATE THAT WE ARE NOT WITHIN A TEST.
      ENDTST
                                L100:
                                TRAP  C$ETST
    
```

HARDWARE TEST

- REPBMP -

```

7306 :*****
7307 :
7308 :           VDHA.HWQ
7309 :
7310 :*****
7311 :
7312 :
7313 :
7314 :.SBTTL  HARDWARE PARAMETER CODING SECTION
7315 :
7316 :
7317 :
7318 :
7319 :***
7320 : THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
7321 : THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
7322 : MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7323 : INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
7324 : MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7325 : WITH THE OPERATOR.
7326 :--
7327 036306          BGNHRD
7328 036306          000022
7329 036310
7330 :
7331 :
7332 :
7333 :
7334 :
7335 :
7336 :
7337 :
7338 :
7339 036310          000031
7340 036310          036354
7341 036312          160000
7342 036314          177776
7343 036316
7344 :
7345 :
7346 :
7347 :
7348 :
7349 :
7350 :
7351 :
7352 :
7353 :
7354 :
7355 :
7356 :

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

.EVEN
L10061:

HARDWARE PARAMETER CODING SECTION

7357	036354	103	123	122	HWPTQ1: .ASCIZ /CSR ADDRESS: /	
	036357	040	101	104		
	036362	104	122	105		
	036365	123	123	072		
	036370	040	000			
7358	036372	111	116	124	HWPTQ2: .ASCIZ /INTERRUPT VECTOR ADDRESS: /	
	036375	105	122	122		
	036400	125	120	124		
	036403	040	126	105		
	036406	103	124	117		
	036411	122	040	101		
	036414	104	104	122		
	036417	105	123	123		
	036422	072	040	000		
7359	036425	101	103	124		HWPTQ3: .ASCIZ /ACTIVE LINE BIT MAP: /
	036430	111	126	105		
	036433	040	114	111		
	036436	116	105	040		
	036441	102	111	124		
	036444	040	115	101		
	036447	120	072	040		
	036452	000				
7360	036453	111	116	124	HWPTQ5: .ASCIZ /INTERRUPT BR LEVEL: /	
	036456	105	122	122		
	036461	125	120	124		
	036464	040	102	122		
	036467	040	114	105		
	036472	126	105	114		
	036475	072	040	000		
7361					.EVEN	
7362						

HARDWARE PARAMETER CODING SECTION

```

7365 ;*****
7366 ;
7367 ;           VDHA.SWQ
7368 ;
7369 ;*****
    
```

```

7371
7372
7373 .SBTTL SOFTWARE PARAMETER CODING SECTION
7374
7375
    
```

```

7376 ;++
7377 ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
7378 ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
7379 ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7380 ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
7381 ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7382 ; WITH THE OPERATOR.
7383 ;--
    
```

```

7384 036500          BGNSFT
       036500 000013
       036502
                                .WORD L10062-L$SOFT/2
                                L$SOFT::
    
```

```

7385
7394 ;UNIT NUMBER PRINTOUT QUESTION:
7395 036502          GPRML SWPTQ1,0,20,YES
       036502 000130
       036504 036530          .WORD T$CODE
       036506 000020          .WORD SWPTQ1
                                .WORD 20
    
```

```

7396 ;NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE QUESTION:
7397 036510          GPRMD SWPTQ2,2,D,177777,0,177777,YES
       036510 001052
       036512 036604          .WORD T$CODE
       036514 177777          .WORD SWPTQ2
       036516 000000          .WORD 177777
       036520 177777          .WORD T$LOLIM
                                .WORD T$HILIM
    
```

```

7398 ;ROM VERSION PRINTOUT ON FIRST PASS QUESTION:
7399 036522          GPRML SWPTQ3,0,1,YES
       036522 000130
       036524 036673          .WORD T$CODE
       036526 000001          .WORD SWPTQ3
                                .WORD 1
    
```

```

7400
7401          .EVEN
7402
7403 036530          ENDSFT
                                .EVEN
                                L10062:
    
```

```

7404
7405
7412 036530          122      105      120
       036533          117      122      124
       036536          040      125      116
       036541          111      124      040
       036544          116      125      115
       036547          102      105      122
       036552          040      101      123
       036555          040      105      101
       036560          103      110      040
       036563          125      116      111
       SWPTQ1: .ASCIZ /REPORT UNIT NUMBER AS EACH UNIT IS TESTED: /
    
```

SOFTWARE PARAMETER CODING SECTION

	036566	124	040	111
	036571	123	040	124
	036574	105	123	124
	036577	105	104	072
	036602	040	000	
7413	036604	116	125	115
	036607	102	105	122
	036612	040	117	106
	036615	040	111	116
	036620	104	111	126
	036623	111	104	125
	036626	101	114	040
	036631	104	101	124
	036634	101	040	105
	036637	122	122	117
	036642	122	123	040
	036645	124	117	040
	036650	122	105	120
	036653	117	122	124
	036656	040	117	116
	036661	040	101	040
	036664	114	111	116
	036667	105	072	040
	036672	000		
7414	036673	122	117	115
	036676	040	126	105
	036701	122	123	111
	036704	117	116	040
	036707	120	122	111
	036712	116	124	117
	036715	125	124	040
	036720	117	116	040
	036723	124	110	105
	036726	040	106	111
	036731	122	123	124
	036734	040	120	101
	036737	123	123	072
7415	036742	040	000	

SWPTQ2: .ASCIZ /NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE: /

SWPTQ3: .ASCIZ /ROM VERSION PRINTOUT ON THE FIRST PASS: /

.EVEN

SOFTWARE PARAMETER CODING SECTION

```

7417
7418 ;*****
7419 ;
7420 ;           FVTSKL6.P11
7421 ;
7422 ;*****
7423
7424
7425
7426 036744 $PATCH::
7427 036744   .BLKW  24
7428
7435
7436
7437
7438
7439 037014           LASTAD
                                .EVEN
                                .WORD  0
                                .WORD  0
                                037014 000000
                                037016 000000
                                037020
7440 037020 L$LAST::
7441           ENDMOD
7442
7443
7444
7445
7446
7447
7448           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 G	EM0526 010003 G	EM3002 014632 G
BITTBL 002410 G	C#CLCK= 000062	DFPTBL 002214 G	EM0601 010073 G	EM3003 014726 G
BITO = 000001 G	C#CLEA= 000012	DIAGMC= 000000	EM0602 010127 G	EM3004 015002 G
BIT00 = 000001 G	C#CLOS= 000035	DRADRT 002244 G	EM0603 010307 G	EM3005 015056 G
BIT01 = 000002 G	C#CLP1= 000006	DROP 025160	EM0701 010472 G	EM3101 015150 G
BIT02 = 000004 G	C#CPBF= 000074	DR00MG 006004 G	EM0702 010527 G	EM3102 015204 G
BIT03 = 000010 G	C#CPME= 000075	DR02MG 006010 G	EM0703 010707 G	EM9009 015245 G
BIT04 = 000020 G	C#CVEC= 000036	DR04MG 006015 G	EM0801 011072 G	EM9010 015271 G
BIT05 = 000040 G	C#DCLN= 000044	DR06MG 006021 G	EM0802 011132 G	EM9014 015315 G
BIT06 = 000100 G	C#DODU= 000051	DR10MG 006026 G	EM0901 011205 G	EM9017 015411 G
BIT07 = 000200 G	C#DRPT= 000024	DR12MG 006035 G	EM0902 011236 G	EM9018 015522 G
BIT08 = 000400 G	C#DU = 000053	DR14MG 006046 G	EM1001 011272 G	EM9019 015532 G
BIT09 = 001000 G	C#EDIT= 000000	DR16MG 006057 G	EM1002 011336 G	EM9020 015547 G
BIT1 = 000002 G	C#ERDF= 000055	EDROP 025236	EM1003 011423 G	EM9022 015573 G
BIT10 = 002000 G	C#ERHR= 000056	EF.CON= 000036 G	EM1101 011501 G	EM9023 015612 G
BIT11 = 004000 G	C#ERRO= 000060	EF.NEW= 000035 G	EM1201 011525 G	EM9024 015634 G
BIT12 = 010000 G	C#ERSF= 000054	EF.PWR= 000034 G	EM1202 011544 G	EM9026 015652 G
BIT13 = 020000 G	C#ERSO= 000057	EF.RES= 000037 G	EM1203 011630 G	EM9301 015676 G
BIT14 = 040000 G	C#ESCA= 000010	EF.STA= 000040 G	EM1204 011706 G	EM9302 015716 G
BIT15 = 100000 G	C#ESEG= 000005	EF0503 004134 G	EM1205 011742 G	EM9303 015746 G
BIT2 = 000004 G	C#ESUB= 000003	EF0505 004141 G	EM1301 011766 G	EM9304 016013 G
BIT3 = 000010 G	C#ETST= 000001	EF1401 004214 G	EM1302 012012 G	ENDETB 003726 G
BIT4 = 000020 G	C#EXIT= 000032	EF1402 004316 G	EM1401 012051 G	ENDIT 025050
BIT5 = 000040 G	C#FREQ= 000101	EF1601 004353 G	EM1402 012101 G	ERCNTB 002464 G
BIT6 = 000100 G	C#FRME= 000100	EF1602 004377 G	EM1403 012167 G	ERLTBL 002726 G
BIT7 = 000200 G	C#GETB= 000026	EF1603 004441 G	EM1404 012242 G	ERRBLK 003774 G
BIT8 = 000400 G	C#GETW= 000027	EF1604 004503 G	EM1405 012314 G	ERRMSG 003772 G
BIT9 = 001000 G	C#GMAN= 000043	EF3001 004600 G	EM1406 012327 G	ERRNBR 003770 G
BMPCQB 002526 G	C#GPHR= 000042	EF3002 004647 G	EM1407 012342 G	ERRTYP 003766 G
BMPCQE 002726 G	C#GPRI= 000040	EF9001 004716 G	EM1408 012354 G	ERSMRF 002462 G
BMPCQP 002524 G	C#INIT= 000011	EF9002 005000 G	EM1601 012362 G	ER0101 016070 G
BOE = 000400 G	C#INLP= 000020	EF9003 005057 G	EM1604 012445 G	ER0201 016406 G
BRLEVL 002242 G	C#MANI= 000050	EF9004 005113 G	EM1701 012513 G	ER0503 016470 G
BUFBAS 002726 G	C#MAP = 000102	EF9005 005150 G	EM1801 012570 G	ER0504 016514 G
BUFEND 003726 G	C#MEM = 000031	EF9006 005201 G	EM1901 012636 G	ER1401 016562 G
BUF MID 003326 G	C#MMU = 000103	EF9010 005225 G	EM2001 012713 G	ER1601 016712 G
BUF PTR 002324 G	C#MSG = 000023	EF9016 005324 G	EM2002 012755 G	ER1603 017010 G
BUF3QT 003526 G	C#OPNR= 000034	EF9017 005421 G	EM2101 013030 G	ER3001 017070 G
CACHRX 023622 G	C#OPNW= 000104	EF9018 005475 G	EM2102 013073 G	ER9004 017160 G
CACHTX 023650 G	C#PNTB= 000014	EF9019 005602 G	EM2201 013167 G	ER9007 017262 G
CALMSL 017702 G	C#PNTF= 000017	EF9301 005626 G	EM2202 013224 G	ER9008 017340 G
CKTRAP 020126 G	C#PNTS= 000016	EF9302 005704 G	EM2203 013316 G	ER9101 017422 G
CLKBRL 002356 G	C#PNTX= 000015	EM0101 020534 G	EM2301 013423 G	ER9301 017450 G
CLKCSR 002354 G	C#PUTB= 000072	EM0102 020620 G	EM2302 013461 G	EVL = 000004 G
CLKHRZ 002362 G	C#PUTW= 000073	EM0103 006067 G	EM2401 013517 G	E#END = 002100
CLKINT 023676 G	C#QIO = 000377	EM0201 006125 G	EM2601 013553 G	E#LOAD= 000035
CLKVEC 002360 G	C#RDBU= 000007	EM0202 006173 G	EM2602 013603 G	F#AU = 000015
CLNRST 020156 G	C#REFG= 000047	EM0203 006346 G	EM2603 013674 G	F#AUTO= 000020
CLR16W 020200 G	C#REL = 000077	EM0204 006511 G	EM2604 013762 G	F#BGN = 000040
CNTERR 020222 G	C#RESE= 000033	EM0301 006670 G	EM2605 014056 G	F#CLEA= 000007
CSRA 002244 G	C#REVI= 000004	EM0302 006733 G	EM2606 014141 G	F#DU = 000016
CSRO = 000000 G	C#FLA= 000021	EM0303 007073 G	EM2607 014200 G	F#END = 000041

Symbol table

F\$HARD= 000004	I\$PTAB= 000041	L\$PROT 024226 G	L10054 034062	RMATBB 002304 G
F\$HW = 000013	I\$PWR = 000041	L\$PRT 002112 G	L10055 035074	ROLDAP 021570 G
F\$INIT= 000006	I\$RPT = 000041	L\$REPP 002062 G	L10056 035746	RSTRPT 021622 G
F\$JMP = 000050	I\$SEG = 000041	L\$REV 002010 G	L10057 036224	RXBDBTX= 000030 G
F\$MOD = 000000	I\$SETU= 000041	L\$RPT 024220 G	L10060 036304	RXBETX= 000020 G
F\$MSG = 000011	I\$SFT = 000041	L\$SOFT 036502 G	L10061 036354	RXBFUL= 000100 G
F\$PROT= 000021	I\$SRV = 000041	L\$SPC 002056 G	L10062 036530	RXBRRT 023746 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 022236 G
F\$SEG = 000003	J\$JMP = 000167	L\$STA 002030 G	MMENAB 002404 G	RXINPT 024036 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	ROSLOT= 000002 G
F\$TEST= 000001	LPCSLT= 000036 G	L10000 002224	MSG3 016331 G	R1SLOT= 000004 G
GETPRM 024634	LPRA 002250 G	L10001 002232	MSLCNT 002376 G	R2SLOT= 000006 G
GPRS08 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 025244 G	L10005 016560	NDERPT 002230 G	SAVBMP 022262 G
G\$EXCP= 000400	L\$AUT 002070 G	L10006 016710	NEWPAS 024614	SFPTBL 002226 G
G\$HILI= 000002	L\$AUTO 025076 G	L10007 017006	NEWRES 024606	SKPSTS 022330 G
G\$LOLI= 000001	L\$CCP 002106 G	L10010 017066	NEWSTA 024300	STATO 002252 G
G\$NO = 0C0000	L\$CLEA 025100 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 024224	O\$BGNS= 000001	SWAPO 022406 G
G\$RADB= 000000	L\$DLY 002116 G	L10021 025074	O\$DU = 000001	SWPTQ1 036530
G\$RADD= 000040	L\$DTP 002040 G	L10022 025076	O\$ERRT= 000001	SWPTQ2 036604
G\$RADL= 000120	L\$DTYP 002034 G	L10023 025132	O\$GNSW= 000001	SWPTQ3 036673
G\$RADO= 000020	L\$DU 025134 G	L10024 025242	O\$POIN= 000001	S\$LSYM= 010000
G\$XFER= 000004	L\$DUT 002072 G	L10025 025250	O\$SETU= 000000	TIMER1 002364 G
G\$YES = 000010	L\$DVTY 004036 G	L10026 025540	PAROA 002406 G	TIMER2 002366 G
HELP = 000000	L\$EF 002052 G	L10027 025754	PASCNT 002332 G	TIMER3 002370 G
HOE = 100000 G	L\$ENVI 002044 G	L10030 026204	PCSL0T= 000016 G	TNUM = 000033 G
HWPTQ1 036354	L\$ERRT 003766 G	L10031 026366	PNT = 001000 G	TP4FLG 002346 G
HWPTQ2 036372	L\$ETP 002102 G	L10032 026544	PREGRT 004020 G	TP4RTN 024122 G
HWPTQ3 036425	L\$EXP1 002046 G	L10033 026746	PREG05 003776	TP4VEC 002344 G
HWPTQ5 036453	L\$EXP4 002064 G	L10034 027140	PRI = 002000 G	TSABRT 022456 G
IBE = 010000 G	L\$EXP5 002066 G	L10035 027332	PRI00 = 000000 G	TSTNUM 002326 G
IDU = 000040 G	L\$HARD 036310 G	L10036 027504	PRI01 = 000040 G	TXAD1A 002256 G
IER = 020000 G	L\$HIME 002120 G	L10037 027704	PRI02 = 000100 G	TXAD10= 000012 G
IESTAT 002330 G	L\$HPCP 002016 G	L10040 030116	PRI03 = 000140 G	TXAD2A 002260 G
ISR = 000100 G	L\$HPTP 002022 G	L10041 030364	PRI04 = 000200 G	TXAD20= 000014 G
IXE = 004000 G	L\$HW 002214 G	L10042 030726	PRI05 = 000240 G	TXBFCA 002262 G
I\$AU = 000041	L\$ICP 002104 G	L10043 031312	PRI06 = 000300 G	TXBFC0= 000016 G
I\$AUTO= 000041	L\$INIT 024234 G	L10044 031522	PRI07 = 000340 G	TXCHA 002246 G
I\$CLN = 000041	L\$LADP 002026 G	L10045 031736	PUFIFO 020716 G	TXCHRO= 000002 G
I\$DU = 000041	L\$LAST 037020 G	L10046 032202	RBUFA 002246 G	TXDSBL 022570 G
I\$HRD = 000041	L\$LOAD 002100 G	L10047 032452	RBUFO = 000002 G	TXENBL 022664 G
I\$INIT= 000041	L\$LUN 002074 G	L10050 032560	RDPDR 021000 G	TXIE0 022760 G
I\$MOD = 000041	L\$MREV 002050 G	L10051 032770	REGTST 021220 G	TXIE1 023012 G
I\$MSG = 000041	L\$NAME 002000 G	L10052 033224	REPSMR 021430 G	TXINTC 002340 G
I\$PROT= 000040	L\$PRIO 002042 G	L10053 033524	RESETT 021456 G	TXINTF 002342 G

Symbol table

TXINTR 024144 G	T\$SAVL= 177777	T\$\$PRO= 010020	T2 025542 G	UAM = 000200 G
TXVECA 002234 G	T\$SEGL= 177777	T\$\$RPT= 010017	T20 032562 G	UNBTTB 002264 G
T\$ARGC= 000003	T\$SUBN= 000000	T\$\$SOF= 010062	T21 032772 G	UNITN 002240 G
T\$CODE= 000130	T\$TAGL= 177777	T\$\$SW = 010001	T22 033226 G	UNSDIV 023036 G
T\$ERRN= 022125	T\$TAGN= 010063	T\$\$TES= 010060	T23 033526 G	WAIBIC 023172 G
T\$EXCP= 000000	T\$TEMP= 000000	T1 025252 G	T24 034064 G	WAIBIS 023246 G
T\$FLAG= 000050	T\$TEST= 000033	T10 027506 G	T25 035076 G	WDPDR 023322 G
T\$GMAN= 000000	T\$TSTM= 177777	T11 027706 G	T26 035750 G	WORD1 002350 G
T\$HILI= 177777	T\$TSTS= 000001	T12 030120 G	T27 036226 G	WTMLNC 023516 G
T\$LAST= 000001	T\$\$AU = 010025	T13 030366 G	T3 025756 G	WTMLNS 023546 G
T\$LOLI= 000000	T\$\$AUT= 010022	T14 030730 G	T4 026206 G	WTMLPR 023572 G
T\$LSYM= 010000	T\$\$CLE= 010023	T15 031314 G	T5 026370 G	X\$ALWA= 000000
T\$LTNO= 000033	T\$\$DU = 010024	T16 031524 G	T6 026546 G	X\$FALS= 000040
T\$NEST= 177777	T\$\$HAR= 010061	T17 031740 G	T7 026750 G	X\$OFF= 000400
T\$NSO = 000000	T\$\$HW = 010000	T18 032204 G	T8 027142 G	X\$TRIE= 000020
T\$NS1 = 000005	T\$\$INI= 010021	T19 032454 G	T9 027334 G	\$PATCH 036744 G
T\$PTNU= 000000	T\$\$MSG= 010016			

. ABS. 037020 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 269
 Work file writes: 279
 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: 00:52:56.11
 CVDHADO.BIN,CVDHADO.LST/-SP=SVC40/ML,CVDHADO.P11