

DHV11-M

DHV11-M TST ORION UFD
CVDHEB0

AH-T894B-MC
1 OF 2 OCT 1985
COPYRIGHT© 1985

digital
MADE IN USA

DHV11-M

DHV11-M TST ORION UFD
CVDHEB0

AH-T894B-MC
 2 OF 2 OCT 1985
 COPYRIGHT© 1985

digital
MADE IN USA

4 J2 w
A 1
1

SEQ 000

CVDHEBO DHV11 M TEST, ORION UFD MACRO Y05.02 Wednesday 08-May-85 10:16 Page 2
PROGRAM DOCUMENT

.REM 2

IDENTIFICATION

PRODUCT CODE: AC T893B-MC
PRODUCT NAME: CVDHEBO DHV11-M TEST, ORION UFD
PRODUCT DATE: MAY 1985
MAINTAINER: Bruce Ribolini MK Diagnostics Group
AUTHOR: Bert Kleinschmidt
Tony Grimshaw
MODIFIED BY: Bert Kleinschmidt

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) 1985 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

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

Original release: 06 Feb-84 Bert Kleinschmidt
Created from CVDHAB, CVDHBC, and CVDHCB.

Version 80 17 Jul-84 Bert Kleinschmidt
Fixed typographical errors discovered to date.
Modified control of processor priority and LTC throughout
the program to guarantee a less than 2 second
response to a Break request while running under APT.
Fixed bug in IAUTO Active test which caused an XON
character to arrive late on processors with cache.

Version 80 28-Sep-84 Peter ONeil
Modified VDHE.CUC to turn off clock if clock was enabled.

Version 80 6-May 85 Howard L. Marshall
Found intermittent failure in IAUTO test because with faster
CPUs, last X-ON/X-OFF character does not get into the FIFO
buffer in time to be read by routine.
Changed test 29, IAUTO test, to provide pause when FIFO buffer
is 1/2 filled (128 characters) instead of when FIFO buffer is
3/4 filled (192 characters).

Changed test 38, DMA Addressing Test to use KPAR5 instead of
KPAR6, and as to not modify KPAR0, 6 + 7.

In addition, found test 40, DMA Transmission Test, to be a
failure under the Extended monitor because the DRS and XXDP
services require more time to execute the SETPRI + GETPRI
macros. Substituted those macros 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.)	18
3.0	ERROR INFORMATION	18
3.1	TYPES OF ERROR MESSAGES	18
3.2	ERROR MESSAGES	19
4.0	PERFORMANCE AND PROGRESS REPORTS	20
5.0	TEST SUMMARIES	20
6.0	EXAMPLE ERROR FREE PASS	23

PROGRAM DOCUMENT

1.0 GENERAL PROGRAM CONSIDERATIONS

1.1 PROGRAM ABSTRACT

CVDHE is a Functional Verification Test (FVT) for the DHV11 M 8 line asynchronous multiplexer. This program has been created for inclusion within the Orion User Friendly Diagnostic (UFD) system.

The source file for Rev A of this program was created by merging the source files for CVDHAB, CVDHBC, and CVDHCB (The three parts of the DHV11 M FVT) and removing any code which required external loopback or other operator intervention. Special versions of the single character mode and DMA mode transmission and reception tests have been produced for this program to reduce the run time from that of the standard DHV FVT (at the same time, reducing the test coverage). This program requires less than 30 seconds per DHV11-M for an error free pass.

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 DHV11-M FVT:

- o LSI-11 processor with at least 56 Kbytes of RAM.
- o DHV11 boards installed on the Q-bus.
- o Appropriate program load device supporting XXDP+ media or a down-line loading system.

1.3 RELATED DOCUMENTS AND STANDARDS

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

PROGRAM DOCUMENT

1.4 DIAGNOSTIC HIERARCHY 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.

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 "DDDDD".

SWITCH	EFFECT
-----	-----
/TESTS:LIST	EXECUTE ONLY THOSE TESTS SPECIFIED IN THE LIST. LIST IS A STRING OF TEST NUMBERS, FOR EXAMPLE - /TESTS:1:5:7-10.

PROGRAM DOCUMENT

/PASS:DDDDD
/FLAGS:FLGS

/EOP:DDDDD

/UNITS:LIST

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.
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					

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 STATISTIC" 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

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>
*****
```

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( ED)/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.

2.4.7 Drop Command -

```
*****
DRO(P)/U  :<UNIT-LIST>
*****
```

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

2.4.9 Print Command -

```
*****
PRI(NT)
*****
```

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

2.4.10 Display Command -

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

2.4.10.1 Effect Of Display Command -

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

2.4.11 Flags Command -

```
*****
FLA(GS)
*****
```

2.4.11.1 Effect Of Flags Command -

The current settings of all flags are printed.

PROGRAM DOCUMENT

2.4.12 Zflags Command

ZFL(AGS)

2.4.13 Zflags Command -

All flags are cleared.

2.4.14 Control Characters -

- C A control/C (C) entered during the execution of a diagnostic causes a return to command mode.
- Z A control/Z (Z) entered during one of the two operator dialogues-- hardware P Table dialogue or software P-Table dialogue causes the defaults to be taken for the remainder of that dialogue.
- O A control/O (O) entered during the execution of a diagnostic causes all teletype output to be suppressed for the remainder of the diagnostic or until another control/O is typed, which restores normal teletype output.

2.5 HARDWARE QUESTIONS

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

1. CSR ADDRESS - This question requests the CSR address of the specified DHV11. The default answer for this question is the lowest address in the PDP-11 floating address space in which a DHV11-M can be placed (160460 Octal).
2. INTERRUPT VECTOR ADDRESS - This question requests the interrupt vector address of the specified DHV11.
3. BR Level - This questions requests the interrupt BR level of the DHV11.

2.6 SOFTWARE QUESTIONS

This program contains no software P table questions. Because of this no "CHANGE SW (L) ?" prompt is issued during program execution.

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 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>

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 (D) ? 8<CR>

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

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

UNIT 7
CSR ADDRESS (D) ? 160000<CR>
SUB-DEVICE # (D) ? 6,7<CR>
Q-FACTOR (D) 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 (D) ? 8<CR>

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

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

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

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING ONLY THE DEFAULTS FOR FLAGS. 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 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:

```
CVDHE 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:

```
CVDHE 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 FUTHER INFORMATION SEE THE SWITCHES SECTION OF THIS DOCUMENT.

5.0 TEST SUMMARIES

The following tests are included within CVDHE:

1. Device register address test - Verifies that the UUT registers will respond witht 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 consistant 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 specifi-d period. This test should signal problems that the BMP codes could cause with later tests.
10. Skip selftest test - This test verifies that if the selftest is skipped the proper codes are placed in the FIFO and that no errors are encountered.

PROGRAM DOCUMENT

11. DIAGNOSTIC.FAIL (Skip selftest) test - This test verifies, by using the skip selftest sequence, that the DIAGNOSTIC.FAIL bit will go to both the active and inactive states.
12. Selftest run test - Verifies that no errors are found by the execution of the selftest.
13. Selftest fail test - Verifies that the selftest will report errors correctly when it is forced to fail.
14. Word access read/write test - Verifies that the registers respond correctly to word read and write accesses.
15. Word access read/modify/write test - Verifies that the registers respond correctly to read/modify/write word accesses.
16. Byte access read/write test - Verifies that the registers respond correctly to byte read and write accesses.
17. Byte access read/modify/write test - Verifies that the registers respond correctly to read/modify/write byte accesses.
18. ID.BIT test - Verifies that the ID.BIT reads as a zero.
19. 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.
20. 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.
21. TX.ENABLE inactive test - Verifies that if the TX.ENABLE bit is clear no transmission occurs.
22. TX.ENABLE active test - Verifies that TX occurs if the TX.ENABLE is set.
23. Interrupts test - Verifies that the TX and RX interrupts are functioning correctly.
24. BR level test - Verifies that the UUT generates TX and RX interrupts at the correct BR level.
25. DIAG field (BMP) test - Verifies that a request for BMP code reporting is answered by the UUT within the specified time.
26. DMA.START test - Verifies that each DMA.START bit will initiate a DMA TX on a line.

PROGRAM DOCUMENT

27. DMA.ABORT test - Verifies that the DMA.ABORT bit on each line will stop a DMA transmission and return a TX.ACTION and that the DMA can then be restarted.
28. I.AUTO inactive test - Verifies that the UUT will not generate and TX XON or XOFF characters in response to the FIFO conditions if the I.AUTO bit is inactive.
29. I.AUTO active test - Verifies that the UUT will generate and TX XON and XOFF characters in response to the FIFO conditions if the I.AUTO bit is active.
30. FIFO data test - Verifies that the FIFO will hold 256 characters without corrupting data.
31. FIFO 3/4 level inactive test - Verifies that the FIFO 3/4 alarm does not become active until the FIFO becomes 3/4 full.
32. FIFO 3/4 level active test - Verifies that the FIFO 3/4 alarm becomes active, and remains active, when the FIFO is more than 3/4 full.
33. FIFO 3/4 level active/inactive test - Verifies that the FIFO 3/4 alarm, once activated, remains active until the FIFO level is reduced below 1/2.
34. FIFO 1/2 level test - Verifies that FIFO 1/2 level indicator becomes active and remains active as the FIFO level is reduced below the 1/2 full point.
35. BREAK test - Verifies that setting the BREAK bit on any line causes that line to go to a spacing state and that clearing the BREAK bit removes the line from the spacing state.
36. No OVERRUN.ERROR test - Verifies that the UUT will receive the maximum number of characters without causing an overrun error.
37. OVERRUN.ERROR test - Verifies that if more than the maximum number of characters are sent to the UUT overrun errors occur.
38. DMA Addressing test - Verifies that the UUT can access the full memory which is on the host machine via DMA accesses.
39. Short single character mode TX/RX test - Verifies that the UUT will TX and RX data correctly in single character mode.
40. Short DMA mode TX/RX test - Verifies that the UUT will TX and RX data correctly using DMA transmission.
41. 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.

PROGRAM DOCUMENT

6.0 EXAMPLE ERROR FREE PASS

The following is an example of an error free pass on two units using no optional flags or DRS options:

.R VDHEBO
VDHEBO.BIC

DRS
CVDHE-B-0
DHV11-M TEST, ORION UFD
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
INTERRUPT BR LEVEL: (0) 4? <CR>

CVDHE EOP 1
0 CUMULATIVE ERRORS

+C
DR> EXIT
.

&

1085
1086
1087 000000

.LIST SEQ,LOC,BIN,MEB
.NLIST CND
.ENABLE AMA,ABS,LC ;\$\$\$

PROGRAM DOCUMENT

```

1089 ;*****
1090 ;
1091 ;           VDHE.PHO
1092 ;
1093 ;*****
1094
1095
1096
1097 .SBTTL  Program Header
1098
1099
1100 .MCALL  SVC
1101 000000 SVC ; INITIALIZE SUPERVISOR MACROS
1102
1103 ;*****
1104 ; IF STRUCTURED MACROS ARE TO BE USED, ADD ".MCALL STRUCT" AND "STRUCT"
1105 ; TO INITIALIZE THE STRUCTURED MACROS.
1106
1107 000001 SVCINS= 1 ; LIST INSTRUCTIONS, SHIFTED RIGHT
1108 000001 SVCTST= 1 ; LIST TEST TAGS, SHIFTED RIGHT
1109 000001 SVCSUB= 1 ; LIST SUBTEST TAGS, SHIFTED RIGHT
1110 000001 SVCGBL= 1 ; LIST GLOBAL TAGS, SHIFTED RIGHT
1111 000001 SVCTAG= 1 ; LIST OTHER TAGS, SHIFTED RIGHT
1112
1113 ; CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
1114 ; TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
1115 ; SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
1116 ; CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.
1117 ;*****
1118
1119 000000 .ENABL  ABS
1120 .ENABL  AMA
1121 002000 " 2000
1122
1123 002000 BGNMOD
1124
1125 ;**
1126 ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
1127 ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
1128 ;--
1129
1130 002000 POINTER BGNSW,BGNDU,ERRTBL
1131
1132
1133
1134
1135
1136
1137
1138
1139 002000 HEADER CVDHE,B,0,15,0,PRI07,1
1140 002000
1141 002000 103
1142 002001 126
1143 002002 104
1144 002003 110
1145 002004 105
1146 002005 000
1147 002006 000
1148 002007 000
1149 002010 102
1150 002011

```

```

L$NAME::
        .ASCII /C/
        .ASCII /V/
        .ASCII /D/
        .ASCII /H/
        .ASCII /E/
        .BYTE  0
        .BYTE  0
        .BYTE  0
L$REV::
        .ASCII /B/
L$DEPO::

```

Program Header

```

002011      060
002012
002012      000000
002014
002014      000015
002016
002016      064364
002020
002020      000000
002022
002022      002250
002024
002024      002262
002026
002026      064570
002030
002030      000000
002032
002032      000000
002034
002034      000000
002036
002036      000000
002040
002040      002124
002042
002042      000340
002044
002044      000000
002046
002046      000000
002050
002050      004
002051      000
002052
002052      000000
002054      000000
002056
002056      000000
002060
002060      005534
002062
002062      000000
002064
002064      000000
002066
002066      000000
002070
002070      000000
002072
002072      042326
002074
002074      000000
002076
002076      005544
002100
002100      104035

```

```

      .ASCII /0/
L$UNIT::      .WORD 0
L$TIML::      .WORD 15
L$HPCP::      .WORD L$HARD
L$SPCP::      .WORD 0
L$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 0
L$EXP4::      .WORD 0
L$EXP5::      .WORD 0
L$AUT::      .WORD 0
L$DUT::      .WORD L$DU
L$LUN::      .WORD 0
L$DESP::      .WORD L$DESC
L$LOAD::      .WORD E$LOAD
               EMT

```


SEQ 0026

Program Header

002102
002102 005464
002104
002104 041474
002106
002106 042276
002110
002110 042274
002112
002112 041466
002114
002114 000001
002116
002116 000000
002120
002120 000000

1150

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

DISPATCH TABLE

1162
1163
1164
1165
1166
1167
1168
1169

.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 41

002122 000051
002124
002124 042444
002126 042734
002130 043150
002132 043400
002134 043562
002136 043740
002140 044142
002142 044334
002144 044526
002146 044700
002150 045100
002152 045312
002154 045560
002156 046122
002160 046332
002162 046546
002164 047012
002166 047262
002170 047370
002172 047600
002174 050034
002176 050334
002200 050672
002202 051704
002204 052556
002206 053034
002210 053422
002212 054042
002214 054436
002216 055060
002220 055350
002222 055652
002224 056344
002226 057034
002230 057500
002232 060124
002234 060624
002236 061552
002240 063350
002242 063732
002244 064302

.WORD 41
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
.WORD T28
.WORD T29
.WORD T30
.WORD T31
.WORD T32
.WORD T33
.WORD T34
.WORD T35
.WORD T36
.WORD T37
.WORD T38
.WORD T39
.WORD T40
.WORD T41

1170

DISPATCH TABLE

```

1178
1179
1180
1181
1182
1183
*****
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196 002246      BGNHW  DFPTBL
      002246      000004
      002250
      002250
1197
1198 002250      160460
1199 002252      000300
1200 002254      000377
1201 002256      001
1202 002257      004
1203
1204 002260
      002260

;*****
;
;          VDHE.DHT
;
;*****

.SBTTL  DEFAULT HARDWARE P TABLE

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

      .WORD  L10000-L$HW/2
      L$HW::
      DFPTBL::

      .WORD  160460  ;Default CSR Address
      .WORD  300     ;Default Vector Address
      .WORD  377     ;Default Active lines bit map
      .BYTE  1       ;Default Loopback mode
      .BYTE  4       ;Default BR Level

      ENDPHW

      L10000:

```

DEFAULT HARDWARE P TABLE

```

1206
1207      ;*****
1208      :
1209      :               VDHE.SWT
1210      :
1211      ;*****
1212
1213
1214
1215      .SBTTL  SOFTWARE P TABLE
1216
1217      ;**
1218      ; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
1219      ; PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
1220      ; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
1221      ; AT RUN TIME.
1222      ;--
1223
1224      002260      BGNSW   SFPTBL
1225      002260      000002
1226      002262
1227      002262
1228
1229      002262      000000
1230      002264      000000
1231
1232      002266
1233      002266

```

.WORD 110001 L\$SW/2

L\$SW::
SFPTBL::

OPTION:: .WORD 0 ;NO UNIT # OR DMA ADR WIDTH PRINTOUT.
 NDERPT:: .WORD 0 ;DISABLE ERROR SUMMARY REPORTING.

L10001:

ENDSW

SOFTWARE P-TABLE

```

1231
1232      ;*****
1233      ;
1234      ;           VDHE.EQU
1235      ;
1236      ;*****
1237
1238
1239      .SBTTL  GLOBAL EQUATES SECTION
1240
1250
1251
1252
1253      ;**
1254      ; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
1255      ; ARE USED IN MORE THAN ONE TEST.
1256      ; -
1257
1258      000010      NUMLNS==10      ;NUMBER OF LINES ON DHV11 IS 8.
1259      000377      MAPLNS==377    ;BIT MAP OF LINES ON DHV11.
1260
1261      ;***** DEVICE REGISTER OFFSETS FROM THE CSR'S ADDRESS *****
1262      000000      CSRO==0         ;CSR REGISTER OFFSET FROM THE CSR ADDRESS
1263      000002      RBUFO==2        ;RECEIVE REGISTER OFFSET FROM THE CSR ADDRESS
1264      000002      TXCHRO==2       ;TRANSMIT REGISTER OFFSET FROM THE CSR ADDRESS
1265      000004      LPRO==4         ;LINE PARAMETER REGISTER OFFSET FROM THE CSR ADDRESS
1266      000006      STATO==6        ;STATUS REGISTER OFFSET FROM THE CSR ADDRESS
1267      000010      LNCTRO==10      ;LINE CONTROL REGISTER OFFSET FROM THE CSR ADDRESS
1268      000012      TXAD10==12      ;TRANSMIT ADDRESS 1 REGISTER OFFSET FROM THE CSR ADDRESS
1269      000014      TXAD20==14      ;TRANSMIT ADDRESS 2 REGISTER OFFSET FROM THE CSR ADDRESS
1270      000016      TXBFCO==16     ;TRANSMIT COUNT REGISTER OFFSET FROM THE CSR ADDRESS
1271
1272      ;***** EQUATES USED WITH RESPECT TO THE RX BUFFER *****
1273      000020      RXBETX==16.      ;LEVEL OF RX BUFFER AT WHICH TO RE ENABLE TRANSMISSION.
1274      000030      RXBDTX==24.     ;LEVEL OF RX BUFFER AT WHICH TO DISABLE TRANSMISSION.
1275      000100      RXBFUL==64.     ;TOTAL CHARACTER CAPACITY OF THE RX BUFFER.
1276
1277
1292 002266      EQUALS
      ;
      ; BIT DIFINITIONS
      ;
      100000      BIT15== 100000
      040000      BIT14== 40000
      020000      BIT13== 20000
      010000      BIT12== 10000
      004000      BIT11== 4000
      002000      BIT10== 2000
      001000      BIT09== 1000
      000400      BIT08== 400
      000200      BIT07== 200
      000100      BIT06== 100
      000040      BIT05== 40
      000020      BIT04== 20
      000010      BIT03== 10
      000004      BIT02== 4
      000002      BIT01== 2

```

GLOBAL EQUATES SECTION

```

000001      BIT00== 1
;
001000      BIT9==  BIT09
000400      BIT8==  BIT08
000200      BIT7==  BIT07
000100      BIT6==  BIT06
000040      BIT5==  BIT05
000020      BIT4==  BIT04
000010      BIT3==  BIT03
000004      BIT2==  BIT02
000002      BIT1==  BIT01
000001      BIT0==  BIT00
;
; EVENT FLAG DEFINITIONS
; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
;
; BIT POSITION IN SECOND STATUS WORD
000040      EF.START== 32.      ; (100000) START COMMAND WAS ISSUED
000037      EF.RESTART== 31.    ; (040000) RESTART COMMAND WAS ISSUED
000036      EF.CONTINUE== 30.   ; (020000) CONTINUE COMMAND WAS ISSUED
000035      EF.NEW== 29.        ; (010000) A NEW PASS HAS BEEN STARTED
000034      EF.PWR== 28.        ; (004000) A POWER-FAIL/POWER-UP OCCURRED
;
; PRIORITY LEVEL DEFINITIONS
;
000340      PRI07== 340
000300      PRI06== 300
000240      PRI05== 240
000200      PRI04== 200
000140      PRI03== 140
000100      PRI02== 100
000040      PRI01== 40
000000      PRI00== 0
;
; OPERATOR FLAG BITS
;
000004      EVL== 4
000010      LOT== 10
000020      ADR== 20
000040      IDU== 40
000100      ISR== 100
000200      UAM== 200
000400      BOE== 400
001000      PNT== 1000
002000      PRI== 2000
004000      IXE== 4000
010000      IBE== 10000
020000      IER== 20000
040000      LOE== 40000
100000      HOE== 100000

```


GLOBAL EQUATES SECTION

```

1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315 002266 000300
1316 002270 000304
1317 002272 000377
1318 002274 000
1319 002275 004
1320 002276 000000
1321
1322
1323
1324
1325 002300
1326 002300 160000
1327 002302 160002
1328 002304 160004
1329 002306 160006
1330 002310 160010
1331 002312 160012
1332 002314 160014
1333 002316 160016
1334
1335
1336
1337
1338 002320 137660
1339 002322 177777
1340 002324 000007
1341 002326 177777
1342 002330 166051
1343 002332 000000
1344 002334 077700
1345 002336 000000
1346
1347
1348
1349
1350 002340 010644
1351 002342 010650

```

```

;*****
;
;          VDHE.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.
LOPBCK:: .BYTE 0        ;LOOPBACK MODE
BRLEVL:: .BYTE 4        ;INTERRUPT BUS REQUEST LEVEL
UNITN:: .WORD 0         ;UNIT NUMBER.

;*****
;          Device Register Address Table
;*****
DRADRT::
CSRA:: .WORD 160000     ;DHV11-M CSR ADDRESS
TXCHA:: RBUFA:: .WORD 160002 ;DHV11-M RECEIVE/TRANSMIT BUFFER ADDRESS
LPRA:: .WORD 160004     ;DHV11-M LINE PARAMETER REGISTER ADDRESS
STATA:: .WORD 160006     ;DHV11-M STATUS REGISTER ADDRESS
LNCTRA:: .WORD 160010    ;DHV11-M LINE CONTROL REGISTER ADDRESS
TXAD1A:: .WORD 160012    ;DHV11-M TRANSMIT BUFFER 1 REGISTER ADDRESS
TXAD2A:: .WORD 160014    ;DHV11-M TRANSMIT BUFFER 2 REGISTER ADDRESS
TXBFCA:: .WORD 160016    ;DHV11-M TRANSMIT BUFFER COUNT REGISTER ADDRESS

;*****
;          Bit mask table of un-used DHV11-M device register bits.
;*****
UNBTTB:: .WORD 137660    ;UNUSED BIT MASK FOR THE CSR
          .WORD 177777    ;UNUSED BIT MASK FOR THE RBUF/TX REG
          .WORD 7         ;UNUSED BIT MASK FOR THE LPR
          .WORD 177777    ;UNUSED BIT MASK FOR THE STAT
          .WORD 166051    ;UNUSED BIT MASK FOR THE LNCTRL
          .WORD 0         ;UNUSED BIT MASK FOR THE TBUFFAD1
          .WORD 77700     ;UNUSED BIT MASK FOR THE TBUFFAD2
          .WORD 0         ;UNUSED BIT MASK FOR THE TBUFFCT

;*****
;          Register Message Address Table
;*****
RMTAB:: .WORD DR00MG     ;ADDRESS OF "CSR" MESSAGE.
         .WORD DR02MG     ;ADDRESS OF "RBUF" MESSAGE.

```

GLOBAL DATA SECTION

```

1352 002344 010655      .WORD  DR04MG  ;ADDRESS OF "LPR" MESSAGE.
1353 002346 010661      .WORD  DR06MG  ;ADDRESS OF "STAT" MESSAGE.
1354 002350 010666      .WORD  DR10MG  ;ADDRESS OF "LNCTRL" MESSAGE.
1355 002352 010675      .WORD  DR12MG  ;ADDRESS OF "TBUFAD1" MESSAGE.
1356 002354 010706      .WORD  DR14MG  ;ADDRESS OF "TBUFAD2" MESSAGE.
1357 002356 010717      .WORD  DR16MG  ;ADDRESS OF "TBUFFT" MESSAGE.
1358
1359
1360
1361
1362 002360 000000      ;*****
1363 002362 000000      ; Assorted global variables:
1364 002364 000001      ;*****
1365 002366 000000      BUFPTR:: .WORD  0      ;STORAGE FOR RECEIVE CHARACTER BUFFER POINTER.
1366 002370 031463      CTRLCF:: .WORD  0      ;STORAGE FOR THE CONTROL-C FLAG.
1367 002372 146314      TSTNUM:: .WORD  1      ;STORAGE FOR THE TEST NUMBER.
1368 002374 000000      IBM:: .WORD  0      ;INACTIVE TX/RX BITS MASK.
1369 002376 000000      LGRP1M:: .WORD  31463      ;BIT MAP OF LINES IN LINE GROUP I.
1370 002400 000000      LGRP2M:: .WORD  146314      ;BIT MAP OF LINES IN LINE GROUP II.
1371 002402 000000      IESTAT:: .WORD  0      ;STORAGE FOR STATES OF THE DUT INT ENABLE BITS.
1372 002404 000000      PASCNT:: .WORD  0      ;STO'G FOR PASS COUNT USED IN ROM VERSION# TST.
1373 002406 000000      WORD1:: .WORD  0      ;LOCATION FOR PASSING INDIRECT PARAMETERS.
1374 002410 000000      RXTOUT:: .WORD  0      ;TIME-OUT VALUE FOR WAITING FOR LAST RX CHAR.
1375 002412 000000      SAVTEN:: .WORD  0      ;STORAGE FOR TX.ENABLE STATES, (TXROFF, TXRON).
1376 002414 000000      SAVPRI:: .WORD  0      ;STO'G FOR PROCESSOR PRIORITY, (TXROFF, TXRON).
1377 002416 000000      TXENBM:: .WORD  0      ;STORAGE FOR TX.ENABLE STATES, (BUFFER MGM'NT).
1378 002420 000000      RXINTC:: .WORD  0      ;STORAGE FOR RECEIVER INTERRUPT FLAGS.
1379 002422 000000      RXINTF:: .WORD  0      ;STORAGE FOR RECEIVER INTERRUPT FLAGS.
1380 002424 000000      TXINTC:: .WORD  0      ;STORAGE FOR TRANSMIT INTERRUPT COUNT.
1381 002426 177777      TXINTF:: .WORD  0      ;STORAGE FOR TRANSMIT INTERRUPT FLAGS.
1382
1383 002430 000000      TP4VEC:: .WORD  0      ;STORAGE FOR THE NORMAL 004 TRAP VECTOR.
1384 002432 000000      TP4FLG:: .WORD  0      ;FLAGS SET WHEN AN EXPECTED 004 TRAP OCCURS.
1385 002434 000000      BITLNG:: .WORD  -1      ;NUMBER OF BITS HOST USES TO DEFINE A UNIQUE
1386 002436 000000      FFREM:: .WORD  0      ;ADDR. -1= 16 BITS, 0= 18 BITS, 1= 22 BITS.
1387
1388
1389
1390
1391 002440 177546      DMTSTA:: .WORD  0      ;STO'G FOR ADR OF FIRST FREE WORD AFTER THE DIAG'TIC
1392 002442 000300      GMANWD:: .WORD  0      ;STO'G FOR DMA TEST ADDRESS (IN PAR FORM).
1393 002444 000100      PMSFLG:: .WORD  0      ;WORD FOR GMANxx CALL RETURN PARAMETERS.
1394 002446 000074      ;FLAG INDICATING WHETHER TO PRINT MODEM STATUS.
1395 002450 000000      ;*****
1396 002452 000000      ; Line Time Clock variables and storage.
1397 002454 000170      ;*****
1398 002456 000170      CLKCSR:: .WORD  177546      ;CSR ADDRESS OF THE LTC.
1399 002460 000021      CLKBRL:: .WORD  PRI06      ;INTERRUPT PRIORITY LEVEL OF THE LTC.
1400 002462 000062      CLKVEC:: .WORD  100      ;INTERRUPT VECTOR ADDRESS OF THE LTC.
1401
1402      CLKHRZ:: .WORD  60.      ;INTERRUPT FREQUENCY OF THE LTC.
1403      TIMER1:: .WORD  0      ;HARDWARE CLOCK COUNTER #1.
1404      TIMER2:: .WORD  0      ;HARDWARE CLOCK COUNTER #2.
1405 002464 177572      TIMER3:: .WORD  120.      ;HARDWARE BREAK COUNTER LOCATION.
1406 002466 172516      BCOUNT:: .WORD  120.      ;BREAK COUNT VALUE IN CLOCK TICKS.
1407 002470 000000      MSTICK:: .WORD  17.      ;NUMBER OF MILLI-SECONDS PER LTC TICK.
1408 002472 000000      MSLCNT:: .WORD  62      ;LOOP COUNT (USED BY MSLOOP) TO DELAY 1 MS.

```

```

;*****
; Memory Management Variables and Flags.
;*****

```

```

MMSR0:: .WORD  177572      ;ADDRESS OF MEM MGT STATUS REGISTER #0.
MMSR3:: .WORD  172516      ;ADDRESS OF MEM MGT STATUS REGISTER #3.
MMPRES:: .WORD  0      ;MEM MGT PRESENT FLAG (0 IF MM NOT PRESENT).
MMENAB:: .WORD  0      ;MEM MGT ENABLED FLAG (0 IF MM NOT ENABLED).

```

GLOBAL DATA SECTION

```

1409
1410 002474
1411 002474 172340
1412 002476 172342
1413 002500 172344
1414 002502 172346
1415 002504 172350
1416 002506 172352
1417 002510 172354
1418 002512 172356
1419 002514
1420
1421 002514
1422 002514 172300
1423 002516 172302
1424 002520 172304
1425 002522 172306
1426 002524 172310
1427 002526 172312
1428 002530 172314
1429 002532 172316
1430 002534
1431
1432
1433
1434
1435 002534 000001
1436 002536 000002
1437 002540 000004
1438 002542 000010
1439 002544 000020
1440 002546 000040
1441 002550 000100
1442 002552 000200
1443 002554 000400
1444 002556 001000
1445 002560 002000
1446 002562 004000
1447 002564 010000
1448 002566 020000
1449 002570 040000
1450 002572 100000
1451
1452
1453
1454
1455 002574
1456 002574 000062
1457 002576 000113
1458 002600 000156
1459 002602 000206
1460 002604 000226
1461 002606 000454
1462 002610 001130
1463 002612 002260
1464 002614 003410
1465 002616 003720

```

```

PARATB::
PAR0A:: .WORD 172340 ;BASE OF MEM MGT PAR ADDRESS TABLE.
PAR1A:: .WORD 172342 ;ADDRESS OF MEM MGT PAR #0.
PAR2A:: .WORD 172344 ;ADDRESS OF MEM MGT PAR #1.
PAR3A:: .WORD 172346 ;ADDRESS OF MEM MGT PAR #2.
PAR4A:: .WORD 172350 ;ADDRESS OF MEM MGT PAR #3.
PAR5A:: .WORD 172352 ;ADDRESS OF MEM MGT PAR #4.
PAR6A:: .WORD 172354 ;ADDRESS OF MEM MGT PAR #5.
PAR7A:: .WORD 172356 ;ADDRESS OF MEM MGT PAR #6.
PARATE:: ;ADDRESS OF MEM MGT PAR #7.
;END OF PAR ADDRESS TABLE.

PDRATB::
PDR0A:: .WORD 172300 ;BASE OF MEM MGT PDR ADDRESS TABLE.
PDR1A:: .WORD 172302 ;ADDRESS OF MEM MGT PDR #0.
PDR2A:: .WORD 172304 ;ADDRESS OF MEM MGT PDR #1.
PDR3A:: .WORD 172306 ;ADDRESS OF MEM MGT PDR #2.
PDR4A:: .WORD 172310 ;ADDRESS OF MEM MGT PDR #3.
PDR5A:: .WORD 172312 ;ADDRESS OF MEM MGT PDR #4.
PDR6A:: .WORD 172314 ;ADDRESS OF MEM MGT PDR #5.
PDR7A:: .WORD 172316 ;ADDRESS OF MEM MGT PDR #6.
PDRATE:: ;ADDRESS OF MEM MGT PDR #7.
;END OF MEM MGT PDR ADDRESS TABLE.

;*****
; 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.
;WORD 400 ;BIT 8 SET.
;WORD 1000 ;BIT 9 SET.
;WORD 2000 ;BIT 10 SET.
;WORD 4000 ;BIT 11 SET.
;WORD 10000 ;BIT 12 SET.
;WORD 20000 ;BIT 13 SET.
;WORD 40000 ;BIT 14 SET.
;WORD 100000 ;BIT 15 SET.

;*****
;* Table of DUT Baudrates
;*****
BRTBLB::
;WORD 50. ;BASE OF DUT BAUD RATE TABLE.
;WORD 75. ;BAUD RATE ENTRY FOR CODE 0.
;WORD 110. ;BAUD RATE ENTRY FOR CODE 1.
;WORD 134. ;BAUD RATE ENTRY FOR CODE 2.
;WORD 150. ;BAUD RATE ENTRY FOR CODE 3.
;WORD 300. ;BAUD RATE ENTRY FOR CODE 4.
;WORD 600. ;BAUD RATE ENTRY FOR CODE 5.
;WORD 1200. ;BAUD RATE ENTRY FOR CODE 6.
;WORD 1800. ;BAUD RATE ENTRY FOR CODE 7.
;WORD 2000. ;BAUD RATE ENTRY FOR CODE 8.
;BAUD RATE ENTRY FOR CODE 9.

```

GLOBAL DATA SECTION

```

1466 002620 004540      .WORD 2400.      ;BAUD RATE ENTRY FOR CODE 10.
1467 002622 011300      .WORD 4800.      ;BAUD RATE ENTRY FOR CODE 11.
1468 002624 016040      .WORD 7200.      ;BAUD RATE ENTRY FOR CODE 12.
1469 002626 022600      .WORD 9600.      ;BAUD RATE ENTRY FOR CODE 13.
1470 002630 045400      .WORD 19200.     ;BAUD RATE ENTRY FOR CODE 14.
1471 002632 113000      .WORD 38400.     ;BAUD RATE ENTRY FOR CODE 15.
1472 002634              ;LABEL AFTER END OF DUT BAUDRATE TABLE.
1473
1474
1475
1476 002634              ;*****
1477 002634 000000      ;*      GPR Save Areas Zero and One.
1478 002636 000000      ;*****
1479 002640 000000      ;*****
1480 002642 000000      ;*****
1481 002644 000000      ;*****
1482
1483
1484
1485 002646 000000      ;*****
1486 002650 000000      ;*****
1487 002652 000000      ;*****
1488 002654 000000      ;*****
1489 002656 000000      ;*****
1490
1491
1492
1493 002660 000000      ;*****
1494 002662      ;*****
1495 003062      ;*****
1496
1497
1498
1499 003062 000000      ;*****
1500 003064 000000      ;*****
1501 003066 000000      ;*****
1502 003070
1503 003070
1504 003270 000000      ;*****
1505
1506
1507
1508 003272
1509 003272 000000      ;*****
1510 003274 000000      ;*****
1511 003276 000000      ;*****
1512 003300 000000      ;*****
1513 003302 000000      ;*****
1514 003304 000000      ;*****
1515 003306 000000      ;*****
1516 003310 000000      ;*****
1517
1518
1519
1520 003312
1521 003352
1522 003412

```

```

      .WORD 2400.      ;BAUD RATE ENTRY FOR CODE 10.
      .WORD 4800.      ;BAUD RATE ENTRY FOR CODE 11.
      .WORD 7200.      ;BAUD RATE ENTRY FOR CODE 12.
      .WORD 9600.      ;BAUD RATE ENTRY FOR CODE 13.
      .WORD 19200.     ;BAUD RATE ENTRY FOR CODE 14.
      .WORD 38400.     ;BAUD RATE ENTRY FOR CODE 15.
      ;LABEL AFTER END OF DUT BAUDRATE TABLE.
;*****
;*      GPR Save Areas Zero and One.
;*****
;*****
GPRSOB::      ;BASE OF GPR SAVE AREA NUMBER ZERO.
      .WORD 0          ;WORD 1, STORAGE FOR R1.
      .WORD 0          ;WORD 2, STORAGE FOR R2.
      .WORD 0          ;WORD 3, STORAGE FOR R3.
      .WORD 0          ;WORD 4, STORAGE FOR R4.
      .WORD 0          ;WORD 5, STORAGE FOR R5.
;*****
;*      Transmission and Reception Variables, Pointers, and Flags.
;*****
CHRTOT:: .WORD 0      ;TOTAL RECEIVED CHARACTER COUNTER.
ERSMRF:: .WORD 0      ;"PRINT ERROR SUMMARY" FLAGS.
TXDNF:: .WORD 0       ;TRANSMISSION DONE FLAGS.
RXDNF:: .WORD 0       ;RECEPTION DONE FLAGS.
TXDBLF:: .WORD 0      ;"TX HAS BEEN DISABLED" FLAG.
;*****
;      Storage area for the BMP code queue.
;*****
BMPCQP:: .WORD 0      ;POINTER USED TO ACCESS THE NEXT CELL IN QUE.
BMPCQB:: .BLKW 64.    ;STORAGE FOR 32 CELLS, TEST0 PLUS BMP CODE.
BMPCQE::      ;LAST ADDRESS PLUS 2 OF THE BMP CODE QUEUE.
;*****
;*      Receive Buffer and Associated Variables.
;*****
RXBOPT:: .WORD 0      ;RX BUFFER OUTPUT POINTER.
RXBIPT:: .WORD 0      ;RX BUFFER INPUT POINTER.
RXBCNT:: .WORD 0      ;COUNT OF NUMBER OF CHARS IN RX BUFFER.
RXBSTA::      ;LABEL AT BEGINNING OF THE RX BUFFER.
      .BLKW RXBFUL    ;LEAVE ENOUGH ROOM FOR A FULL BUFFER.
RXBEND:: .WORD 0      ;LABEL AFTER END OF RX BUFFER.
;*****
;*      TX/RX Control Block.
;*****
CBB::      ;BASE OF TX/RX CONTROL BLOCK.
CBLPRA:: .WORD 0      ;LINE PARAMETER REGISTER CONTENTS.
CBLNCA:: .WORD 0      ;LINE CONTROL REGISTER CONTENTS.
CBDPAA:: .WORD 0      ;START ADDRESS OF DATA PATTERN.
CBDPLA:: .WORD 0      ;LENGTH OF DATA PATTERN.
CBDPNA:: .WORD 0      ;NUMBER OF REPEAT TRANSMISSIONS OF THE DATA PATTERN.
CBMAPA:: .WORD 0      ;BIT MAP OF LINES TO INITIALISE.
CBLPBA:: .WORD 0      ;LOOPBACK MODE (AS IN LOPBCK).
CBOFSA:: .WORD 0      ;AMOUNT OF OFFSET BETWEEN EACH TX START.
;*****
;*      Transmission and Reception Tables of Pointers and Counters.
;*****
OPENDB:: .BLKW 16.    ;TABLE OF END ADDRESSES OF DATA PATTERNS.
DPLENB:: .BLKW 16.    ;TABLE OF LENGTH OF DATA PATTERNS FOR LINES.
EXCNTB:: .BLKW 16.    ;EXTRA RECEIVED CHARACTER COUNTERS TABLE.

```

GLOBAL DATA SECTION

```

1523 003452      ERCNTB:: .BLKW 16.      ;CHARACTER RECEIVE ERROR COUNTERS TABLE.
1524 003512      TXPTRB:: .BLKW 16.      ;TRANSMISSION DATA POINTERS TABLE.
1525 003552      RXPTRB:: .BLKW 16.      ;RECEPTION DATA POINTERS TABLE.
1526 003612      CHCNTB:: .BLKW 16.      ;NUMBER OF CHARACTERS TO BE TXED AND RXED.
1527 003652      TXCNTB:: .BLKW 16.      ;TRANSMISSION CHARACTER COUNTERS TABLE.
1528 003712      RXCNTB:: .BLKW 16.      ;RECEPTION CHARACTER COUNTERS TABLE.
1529
1530      ;*****
1531      ;      Storage area for the contents of the DUT STAT register states.
1532      ;*****
1532 003752      STSTB::      ;BASE OF DUT STAT STORAGE TABLE.
1533 003752      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 0.
1534 003754      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 1.
1535 003756      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 2.
1536 003760      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 3.
1537 003762      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 4.
1538 003764      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 5.
1539 003766      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 6.
1540 003770      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 7.
1541 003772      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 8.
1542 003774      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 9.
1543 003776      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 10.
1544 004000      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 11.
1545 004002      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 12.
1546 004004      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 13.
1547 004006      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 14.
1548 004010      .WORD 0      ;STORAGE FOR STAT REGISTER FOR LINE 15.
1549 004012      STSTE::      ;END OF DUT STAT STORAGE TABLE.
1550      ;*****
1551      ;      General table and buffer area--513 words.
1552      ;*****
1553 004012      BUFBAS::      ;BASE OF MEMORY BUFFER.
1554 004012      ERLTBL:: .BLKW 128.      ;FIRST HALF OF GENERAL TABLE OR BUFFER.
1555 004412      BUFMID:: .BLKW 64.      ;SECOND HALF OF GENERAL TABLE OR BUFFER.
1556 004612      BUF3QT:: .BLKW 64.      ;LAST QUARTER OF THE BUFFER AREA.
1557 005012      BUFEND::      ;END OF GENERAL PURPOSE MEMORY BUFFER.
1558 005012      ENDET8:: .BLKW 16.      ;BUFFER OVERFLOW SPACE.
1559      ;*****
1560      ;      Table of Data Pattern Resync Queues.
1561      ;*****
1562 005052      DPRSQB::      ;DATA PATTERN RESYNC QUEUES TABLE BASE.
1563 005052      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 0.
1564 005062      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 1.
1565 005072      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 2.
1566 005102      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 3.
1567 005112      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 4.
1568 005122      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 5.
1569 005132      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 6.
1570 005142      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 7.
1571 005152      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 8.
1572 005162      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 9.
1573 005172      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 10.
1574 005202      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 11.
1575 005212      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 12.
1576 005222      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 13.
1577 005232      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 14.
1578 005242      .BLKW 4      ;DATA PATTERN RESYNC QUEUE FOR LINE 15.
1579 005252      DPRSQE::      ;END OF DATA PATTERN RESYNC QUEUES TABLE.

```

GLOBAL DATA SECTION

```

1580
1581
1582
1583 005252
1584 005252 000030
1585 005254
1586 005254
1587 005254 000000
1588 005256
1589 005256
1590 005256 000040
1591 005260
1592
1593
1594
1595
1596 005260 000
1597 005261 001
1598 005262 010
1599 005263 017
1600 005264 063
1601 005265 074
1602 005266 125
1603 005267 177
1604 005270 200
1605 005271 252
1606 005272 303
1607 005273 314
1608 005274 360
1609 005275 367
1610 005276 376
1611 005277 377
1612 005300
1613 005300 000
1614 005301 001
1615 005302 010
1616 005303 017
1617
1618
1619
1620
1621 005304 125
1622 005305 252
1623 005306 124
1624 005307 253
1625 005310 122
1626 005311 255
1627 005312 112
1628 005313 265
1629 005314 052
1630 005315 325
1631 005316 152
1632 005317 225
1633 005320 132
1634 005321 245
1635 005322 126
1636 005323 251

```

```

;*****
;      Single Character Mode LPR Field Tables.
;*****
SCBCTB::      ;BASE OF NUMBER OF BITS PER CHAR FIELDS TABLE.
               .WORD 30      ;8 BITS/CHAR LPR FIELD.
;CBCTE::      ;END OF NUMBER OF BITS/CHAR FIELDS TABLE.
SCNSTB::      ;BASE OF NUMBER OF STOP BITS FIELDS TABLE.
               .WORD 0      ;1 STOP BIT LPR FIELD.
SCNSTE::      ;END OF BAUDRATE FIELDS TABLE.
SCTPTB::      ;BASE OF TYPE OF PARITY FIELDS TABLE.
               .WORD 40     ;ODD PARITY LPR FIELD.
SCTPTE::      ;END OF TYPE OF PARITY FIELDS TABLE.

;*****
;      SINGLE CHARACTER DATA PATTERN TABLE.
;*****
SDPBAS::      ;START OF SINGLE CHARACTER DATA PATTERN TABLE.
               .BYTE 0
               .BYTE 1
               .BYTE 10
               .BYTE 17
               .BYTE 63
               .BYTE 74
               .BYTE 125
               .BYTE 177
               .BYTE 200
               .BYTE 252
               .BYTE 303
               .BYTE 314
               .BYTE 360
               .BYTE 367
               .BYTE 376
               .BYTE 377
SDPEND::      ;END OF SINGLE CHARACTER DATA PATTERN TABLE.
               .BYTE 0      ;START OF FIRST SHORT DATA PATTERN OVERFLOW AREA.
               .BYTE 1
               .BYTE 10
               .BYTE 17

;*****
;      SINGLE CHARACTER DATA PATTERN TABLE NUMBER TWO.
;*****
SDP2B::      ;START OF SECOND SHORT DATA PATTERN.
               .BYTE 125
               .BYTE 252
               .BYTE 124
               .BYTE 253
               .BYTE 122
               .BYTE 255
               .BYTE 112
               .BYTE 265
               .BYTE 52
               .BYTE 325
               .BYTE 152
               .BYTE 225
               .BYTE 132
               .BYTE 245
               .BYTE 126
               .BYTE 251

```


GLOBAL DATA SECTION

1637 005324		SDP2E::			
1638 005324	125		.BYTE	125	;END OF SECOND SHORT DATA PATTERN.
1639 005325	252		.BYTE	252	;START OF SECOND SHORT DATA PATTERN OVERFLOW AREA.
1640 005326	124		.BYTE	124	
1641 005327	253		.BYTE	253	
1642 005330	122		.BYTE	122	
1643 005331	255		.BYTE	255	
1644 005332	112		.BYTE	112	
1645 005333	265		.BYTE	265	
1646 005334	052		.BYTE	52	
1647 005335	325		.BYTE	325	
1648 005336	152		.BYTE	152	
1649 005337	225		.BYTE	225	
1650 005340	132		.BYTE	132	
1651 005341	245		.BYTE	245	
1652 005342	126		.BYTE	126	
1653 005343	251		.BYTE	251	
1654					
1655					
1656					
1657 005344	372				
1658 005345	252				
1659 005346	167				
1660 005347	143				
1661 005350	132				
1662 005351	062				
1663 005352	036				
1664 005353	024				
1665 005354	021				
1666 005355	020				
1667 005356	017				
1668 005357	015				
1669 005360	014				
1670 005361	014				
1671 005362	013				
1672 005363	012				
1673					
1674					
1675					
1676					
1677					
1678					
1679					
1680 005364					
1681 005364	000000				
1682 005366	000002				
1683 005370	000004				
1684 005372	000006				
1685 005374	000010				
1686 005376	000012				
1687 005400	000014				
1688 005402	000016				
1689 005404	000020				
1690 005406	000022				
1691 005410	000024				
1692 005412	000026				
1693 005414	000030				

 ; Single character safe proportional delay table.

 PROTBL:: .BYTE 250. ;DELAY IN MILLI SECONDS AT 50 BAUD
 .BYTE 170. ;DELAY IN MILLI SECONDS AT 75 BAUD
 .BYTE 119. ;DELAY IN MILLI SECONDS AT 110 BAUD
 .BYTE 99. ;DELAY IN MILLI SECONDS AT 134.5 BAUD
 .BYTE 90. ;DELAY IN MILLI SECONDS AT 150 BAUD
 .BYTE 50. ;DELAY IN MILLI SECONDS AT 300 BAUD
 .BYTE 30. ;DELAY IN MILLI SECONDS AT 600 BAUD
 .BYTE 20. ;DELAY IN MILLI SECONDS AT 1200 BAUD
 .BYTE 17. ;DELAY IN MILLI SECONDS AT 1800 BAUD
 .BYTE 16. ;DELAY IN MILLI SECONDS AT 2000 BAUD
 .BYTE 15. ;DELAY IN MILLI SECONDS AT 2400 BAUD
 .BYTE 13. ;DELAY IN MILLI SECONDS AT 4800 BAUD
 .BYTE 12. ;DELAY IN MILLI SECONDS AT 7200 BAUD
 .BYTE 12. ;DELAY IN MILLI SECONDS AT 9600 BAUD
 .BYTE 11. ;DELAY IN MILLI SECONDS AT 19200 BAUD
 .BYTE 10. ;DELAY IN MILLI SECONDS AT 38400 BAUD
 .EVEN

 ;* Table for storage of RX/TX line number associations.
 ;* The associations are stored as line number times 2 for use as offsets
 ;* when accessing a table of words.
 ;* NOTE: Do not write a non-zero value into the upper byte of any entry.

 TXRXLB:: ;BASE OF TX/RX LINE NUMBER ASSOCIATION TABLE.
 .WORD 0 ;TX/RX LINE OFFSET FOR RX/TX LINE 0.
 .WORD 2. ;TX/RX LINE OFFSET FOR RX/TX LINE 1.
 .WORD 4. ;TX/RX LINE OFFSET FOR RX/TX LINE 2.
 .WORD 6. ;TX/RX LINE OFFSET FOR RX/TX LINE 3.
 .WORD 8. ;TX/RX LINE OFFSET FOR RX/TX LINE 4.
 .WORD 10. ;TX/RX LINE OFFSET FOR RX/TX LINE 5.
 .WORD 12. ;TX/RX LINE OFFSET FOR RX/TX LINE 6.
 .WORD 14. ;TX/RX LINE OFFSET FOR RX/TX LINE 7.
 .WORD 16. ;TX/RX LINE OFFSET FOR RX/TX LINE 8.
 .WORD 18. ;TX/RX LINE OFFSET FOR RX/TX LINE 9.
 .WORD 20. ;TX/RX LINE OFFSET FOR RX/TX LINE 10.
 .WORD 22. ;TX/RX LINE OFFSET FOR RX/TX LINE 11.
 .WORD 24. ;TX/RX LINE OFFSET FOR RX/TX LINE 12.

GLOBAL DATA SECTION

```

1694 005416 000032      .WORD 26.      ;TX/RX LINE OFFSET FOR RX/TX LINE 13.
1695 005420 000034      .WORD 28.      ;TX/RX LINE OFFSET FOR RX/TX LINE 14.
1696 005422 000036      .WORD 30.      ;TX/RX LINE OFFSET FOR RX/TX LINE 15.
1697 005424              TXRXLE::      ;END OF TX/RX LINE NUMBER ASSOCIATION TABLE.
1698                      .EVEN          ;GUARANTEE THAT NEXT TABLE IS ON WORD BOUNDARY.
1699                      ;*****
1700                      ;* Table for storage of RX/TX line number associations.
1701                      ;* The associations are stored as line numbers which can be used as such or
1702                      ;* as offsets when accessing a table of bytes.
1703                      ;*****
1704 005424              TXRLNB::      ;BASE OF TX/RX LINE NUMBER ASSOCIATION TABLE.
1705 005424 000          .BYTE 0        ;TX/RX LINE FOR RX/TX LINE 0.
1706 005425 001          .BYTE 1.      ;TX/RX LINE FOR RX/TX LINE 1.
1707 005426 002          .BYTE 2        ;TX/RX LINE FOR RX/TX LINE 2.
1708 005427 003          .BYTE 3.      ;TX/RX LINE FOR RX/TX LINE 3.
1709 005430 004          .BYTE 4.      ;TX/RX LINE FOR RX/TX LINE 4.
1710 005431 005          .BYTE 5.      ;TX/RX LINE FOR RX/TX LINE 5.
1711 005432 006          .BYTE 6.      ;TX/RX LINE FOR RX/TX LINE 6.
1712 005433 007          .BYTE 7.      ;TX/RX LINE FOR RX/TX LINE 7.
1713 005434 010          .BYTE 8.      ;TX/RX LINE FOR RX/TX LINE 8.
1714 005435 011          .BYTE 9.      ;TX/RX LINE FOR RX/TX LINE 9.
1715 005436 012          .BYTE 10.     ;TX/RX LINE FOR RX/TX LINE 10.
1716 005437 013          .BYTE 11.     ;TX/RX LINE FOR RX/TX LINE 11.
1717 005440 014          .BYTE 12.     ;TX/RX LINE FOR RX/TX LINE 12.
1718 005441 015          .BYTE 13.     ;TX/RX LINE FOR RX/TX LINE 13.
1719 005442 016          .BYTE 14.     ;TX/RX LINE FOR RX/TX LINE 14.
1720 005443 017          .BYTE 15.     ;TX/RX LINE FOR RX/TX LINE 15.
1721 005444              TXRLNE::      ;END OF TX/RX LINE NUMBER ASSOCIATION TABLE.
1722                      .EVEN          ;GUARANTEE THAT NEXT TABLE IS ON WORD BOUNDARY.
1723                      ;*****
1724                      ;* Table of TX/RX line number associations in staggered loopback.
1725                      ;* The associations are stored as line number times 2 for use as offsets
1726                      ;* when accessing a table of words.
1727                      ;* This is a table of data for reading only. Use to load the above table.
1728                      ;* NOTE: Must convert from BYTES to WORDS when loading above table.
1729                      ;*****
1730 005444              STGTRB::      ;BASE OF STAGGERED TX/RX LINE NUMBER TABLE.
1731 005444 004          .BYTE 4.        ;TX/RX LINE OFFSET FOR RX/TX LINE 0.
1732 005445 006          .BYTE 6.        ;TX/RX LINE OFFSET FOR RX/TX LINE 1.
1733 005446 000          .BYTE 0        ;TX/RX LINE OFFSET FOR RX/TX LINE 2.
1734 005447 002          .BYTE 2.        ;TX/RX LINE OFFSET FOR RX/TX LINE 3.
1735 005450 014          .BYTE 12.      ;TX/RX LINE OFFSET FOR RX/TX LINE 4.
1736 005451 016          .BYTE 14.      ;TX/RX LINE OFFSET FOR RX/TX LINE 5.
1737 005452 010          .BYTE 8.        ;TX/RX LINE OFFSET FOR RX/TX LINE 6.
1738 005453 012          .BYTE 10.      ;TX/RX LINE OFFSET FOR RX/TX LINE 7.
1739 005454 024          .BYTE 20.      ;TX/RX LINE OFFSET FOR RX/TX LINE 8.
1740 005455 026          .BYTE 22.      ;TX/RX LINE OFFSET FOR RX/TX LINE 9.
1741 005456 020          .BYTE 16.      ;TX/RX LINE OFFSET FOR RX/TX LINE 10.
1742 005457 022          .BYTE 18.      ;TX/RX LINE OFFSET FOR RX/TX LINE 11.
1743 005460 034          .BYTE 28.      ;TX/RX LINE OFFSET FOR RX/TX LINE 12.
1744 005461 036          .BYTE 30.      ;TX/RX LINE OFFSET FOR RX/TX LINE 13.
1745 005462 030          .BYTE 24.      ;TX/RX LINE OFFSET FOR RX/TX LINE 14.
1746 005463 032          .BYTE 26.      ;TX/RX LINE OFFSET FOR RX/TX LINE 15.
1747                      .EVEN          ;GUARANTEE THAT NEXT TABLE IS ON WORD BOUNDARY.
1760 005464              ERRRTBL
      005464
      005464 000000      ERRRTYP::      .WORD 0

```

LERRRTBL::

GLOBAL DATA SECTION

005466 000000
005470 000000
005472 000000

ERRNBR:: .WORD 0
ERRMSG:: .WORD 0
ERRBLK:: .WORD 0

1761
1762

.EVEN

GPR HANDLING ROUTINES FOR SUBROUTINE CALLS.

```

1764 .SBTTL GPR HANDLING ROUTINES FOR SUBROUTINE CALLS.
1765 ;*****
1766 ;* There are 4 routines and macro definitions used for the handling of
1767 ;* GPR values during subroutine calls within this program. The four
1768 ;* routines/macro calls have the following names:
1769 ;*
1770 ;* SAVE Macro definition used at the beginning of a subroutine to
1771 ;* save the GPR contents for later restoration.
1772 ;* PASS - Macro definition used at the end of a subroutine to restore
1773 ;* the previously saved GPR contents and to leave the contents
1774 ;* of the specified GPR(s) intact (NOT restored).
1775 ;* PREG05 - Subroutine which is called from the SAVE and PASS macro
1776 ;* expansions which actually performs the actions on the GPRs.
1777 ;*
1778 ;* During a subroutine which uses these GPR save routines the values
1779 ;* of the GPRs are stored on the stack in the following stack frame:
1780 ;*
1781 ;* SP -> RET PC INTO PREG05 ROUTINE.
1782 ;* SP+2 -> GPR R0 CONTENTS.
1783 ;* SP+4 -> GPR R1 CONTENTS.
1784 ;* SP+6 -> GPR R2 CONTENTS.
1785 ;* SP+8 -> GPR R3 CONTENTS.
1786 ;* SP+10 -> GPR R4 CONTENTS.
1787 ;* SP+12 -> GPR R5 CONTENTS.
1788 ;* SP+14 -> RET PC INTO CALLER OF SUB'TNE WHICH CALLED .-G05.
1789 ;*
1790 ;* Each level of sub'tne calling uses 8 words of stack overhead.
1791 ;* The SAVE and PASS macros can also be used in "straight line code"
1792 ;* to save and restore the GPR values. In any case, after the
1793 ;* issuing of a PASS call the GPRs will be restored to the values
1794 ;* they had prior to the last SAVE call (except for the excepted,
1795 ;* or passed intact, GPRs specified as parameters to the PASS call)
1796 ;* and the SP will also be restored to its condition before the last
1797 ;* SAVE call. The programmer must be sure that the SP has the same
1798 ;* value when the PASS macro is called as it had immediately after
1799 ;* the SAVE macro was called.
1800 ;*****

```

GPR FRAME ACCESS EQUATES

1802		.SBTTL GPR FRAME ACCESS EQUATES		
1803		;***		
1804		;Equates that allow access to the stack frame. These are the		
1805		;offsets into the stack for registers saved during the PREG05		
1806		;routine.		
1807		;-		
1808				
1809	000036	LPCSLT==	36	;Offset for last return PC.
1810	000016	PCSL0T==	16	;Offset for return PC.
1811	000014	R5SL0T==	14	;Offset for R5.
1812	000012	R4SL0T==	12	;Offset for R4.
1813	000010	R3SL0T==	10	;Offset for R3.
1814	000006	R2SL0T==	6	;Offset for R2.
1815	000004	R1SL0T==	4	;Offset for R1.
1816	000002	R0SL0T==	2	;Offset for R0.

GLOBAL MACRO DEFINITION

- SAVE -

```

1818 .SBTTL GLOBAL MACRO DEFINITION - SAVE -
1819 ;*****
1820 ;* This macro is used at the beginning of a subroutine to save the
1821 ;* contents of the GPRs R0 thru R5.
1822 ;*
1823 ;* INPUTS: SP - Unchanged since subroutine was entered
1824 ;* RSSL0T - Offset to stack slot for R5 (Equated to 14 Octal)
1825 ;*
1826 ;* OUTPUTS: GPR save area on the stack is loaded with the contents of GPRs
1827 ;* TOP OF STACK - Loaded with the return address into PREG05
1828 ;*
1829 ;* CALLING SEQUENCE: SAVE
1830 ;*
1831 ;* COMMENTS: No arguments are allowed.
1832 ;* The PASS macro should be called to restore the GPR values.
1833 ;*
1834 ;* SUBORDINATE ROUTINES CALLED: PREG05.
1835 ;*****
1836
1837 .MACRO SAVE
1838 .LIST
1839 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
1840 .NLIST
1841 .ENDM SAVE

```

GLOBAL MACRO DEFINITION

- PASS -

```

1843 .SBTTL GLOBAL MACRO DEFINITION - PASS -
1844 ;*****
1845 ;* This macro is used in conjunction with the SAVE macro. It is
1846 ;* called at end of a subroutine to pass parameters in GPRs back to the
1847 ;* calling routine by altering the GPR save area on the stack and then
1848 ;* returning to PREG05 to restore the GPRs to their saved values.
1849 ;*
1850 ;* INPUTS: Only allowed ARGUMENTS are "R0" thru "R5".
1851 ;* ROSLOT thru R5SLOT must be equated to their respective GPR save
1852 ;* slot offsets before calling this macro.
1853 ;*
1854 ;* OUTPUTS: The GPR values are put in their respective slots on the stack.
1855 ;*
1856 ;* CALLING SEQUENCE: PASS R0,R1,...
1857 ;*
1858 ;* COMMENTS: Any combination of GPR arguments may be listed in any order.
1859 ;* For example, the following are legal:
1860 ;* PASS R1
1861 ;* PASS R4,R0,R2
1862 ;* The GPRs listed as arguments will be passed intact to the
1863 ;* calling routine, all other GPRs will be restored.
1864 ;* The SP must be at its original value when PASS is called.
1865 ;*
1866 ;* The macro call
1867 ;* PASS R0,R3
1868 ;* expands into the following assembly code:
1869 ;* MOV R0,ROSLOT(SP) ;PUT R0 IN STACK SLOT.
1870 ;* MOV R3,R3SLOT(SP) ;PUT R3 IN STACK SLOT.
1871 ;* JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
1872 ;* In this example GPRs R1, R2, R4, and R5 will be restored to
1873 ;* their values contained in the stack frame and R0 and R3
1874 ;* will be left at their values prior to this PASS call.
1875 ;*
1876 ;* SUBORDINATE ROUTINES CALLED: (PREGRT - Label within PREG05, value on stack.)
1877 ;*****
1878
1879 .MACRO PASS A,B,C,D,E,F
1880 .IRP X,<A,B,C,D,E,F>
1881 .IF NB,X
1882 .LIST
1883 MOV X,X'SLOT(SP) ;PUT X IN STACK SLOT.
1884 .NLIST
1885 .ENDC
1886 .ENDM
1887 .LIST
1888 JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
1889 .NLIST
1890 .ENDM PASS

```


GLOBAL SUBROUTINE

- PREG05 -

```

1892 .SBTTL GLOBAL SUBROUTINE - PREG05 -
1893 ;*****
1894 ;* Preserve Registers R0 through R5 for subroutine calls.
1895 ;*
1896 ;* INPUTS: The return address back into the calling routine must be in
1897 ;* GPR R5. (i.e.- Macros use "JSR R5,PREG05".)
1898 ;*
1899 ;* OUTPUTS: Registers R0 through R5 are saved on the stack.
1900 ;*
1901 ;* CALLING SEQUENCE: SAVE ;Macro expansion calls PREG05.
1902 ;* [Subroutine code]...
1903 ;* PASS ;Macro expansion recalls PREG05.
1904 ;*
1905 ;* COMMENTS: This routine is re-entrant.
1906 ;*
1907 ;* Parameters may be passed out of a subroutine by modifying the
1908 ;* register save area on the stack. Use the PASS GPRn macro
1909 ;* to return GPR values intact.
1910 ;* Use the RnSLOT offsets from the SP to pass other parameters.
1911 ;* [Example: MOV VALUE,R0SLOT(SP) ]
1912 ;* Make sure the SP is at its original value when you do this.
1913 ;*
1914 ;* SUBORDINATE ROUTINES CALLED: None.
1915 ;*****
1916
1917 005474 PREG05: ;R5 HAS BEEN LOADED ON THE STACK BY THE SUBROUTINE CALL
1918 005474 010446 MOV R4,-(SP) ;SAVE R4
1919 005476 010346 MOV R3,-(SP) ;SAVE R3
1920 005500 010246 MOV R2,-(SP) ;SAVE R2
1921 005502 010146 MOV R1,-(SP) ;SAVE R1
1922 005504 010046 MOV R0,-(SP) ;SAVE R0
1923 005506 010546 MOV R5,-(SP) ;PUSH RETURN PC ON TOP OF STACK
1924 005510 016605 000014 MOV R5SLOT(SP),R5 ;RESTORE R5 TO VALUE IT HAD BEFORE CALLS
1925
1926 005514 004736 JSR PC,0(SP)+ ;Call the subroutine at the return address
1927 ;from the PREG05 call, putting the present
1928 ;PC on the stack as a return address into
1929 ;this (PREG05) routine.
1930
1931 ;+++
1932 ;The following code is executed when the calling routine does a
1933 ;"return" [JSR PC,0(SP)+] using the PC deposited on the stack above.
1934 ;---
1935
1936 005516 012605 PREGRT: MOV (SP)+,R5 ;Put return PC in R5.
1937 005520 012600 MOV (SP)+,R0 ;Restore R0.
1938 005522 012601 MOV (SP)+,R1 ;Restore R1.
1939 005524 012602 MOV (SP)+,R2 ;Restore R2.
1940 005526 012603 MOV (SP)+,R3 ;Restore R3.
1941 005530 012604 MOV (SP)+,R4 ;Restore R4.
1942
1943 005532 000205 RTS R5 ;Return to the subroutine which called PREG05.
1944 ;restoring R5 in the process.

```


GLOBAL TEXT SECTION

Year	Address	Hex Data	Disassembly	Comment
1946			.SBTTL	GLOBAL TEXT SECTION
1948			;	*****
1949			;	
1950			;	FVTSKL1.P11
1951			;	
1952			;	*****
1954			;	
1955			;	
1956			;	
1957			;	THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
1958			;	MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
1959			;	MORE THAN ONE TEST.
1960			;	-
1961			;	
1962			;	
1963			;	NAMES OF DEVICES SUPPORTED BY PROGRAM
1964			;	
1965	005534			DEVTYPE <DHV11-M>
	005534			
	005534	104	110	126
	005537	061	061	055
	005542	115	000	
				L\$DVTYP::
				.ASCIZ <DHV11 M>
				.EVEN
1966				
1972				
1973				TEST DESCRIPTION
1974				;
1975	005544			DESCRIPT <DHV11 TEST, ORION UFD>
	005544			
	005544	104	110	126
				L\$DESC::
				.ASCIZ /DHV11 TEST, ORION U
	005547	061	061	040
	005552	124	105	123
	005555	124	054	040
	005560	117	122	111
	005563	117	116	040
	005566	125	106	104
	005571	000		
				.EVEN
1976				
1977				
1984				

GLOBAL TEXT SECTION

```
1986
1987      ;*****
1988      ;
1989      ;          VDHE.FMT
1990      ;
1991      ;*****
1992
1993
1994
1995
1996      ; FORMAT STATEMENTS USED IN PRINT CALLS
1997      ;
1998
2009
2010
```

GLOBAL TEXT SECTION

```

2019
2020
2021
2022
2023
2024
;*****
;
;          VDHE.MSG
;
;*****

2025
2026
2027 .NLIST BIN
2028 .SBTTL GLOBAL MESSAGE AREA
2029 ; ***** FORMAT STATEMENTS *****
2030 MFUNIT:: .ASCIZ /%N% TESTING UNIT :%D4%A(D)%N/
2031 EF0503:: .ASCIZ /%T%N/
2032 EF0505:: .ASCIZ /%A %D5%A ILLEGAL INTERRUPTS RECEIVED.%N/
2033 EF1401:: .ASCIZ /%N% ROM VERSION NUMBERS: PROC_1 = %D2%A(D) PROC_2 = %D2%A(D)%N/
2034 EF1402:: .ASCIZ /%T%A ROM VERSION NUMBER %T%N/
2035 EF1601:: .ASCIZ /%A %T%A ABORTED %N/
2036 EF1602:: .ASCIZ /%A EXPECTED DATA: %06%A (0).%N/
2037 EF1603:: .ASCIZ /%A ACTUAL DATA: %06%A (0).%N/
2038 EF1604:: .ASCIZ /%A BAD BIT(S) IN DEVICE %T%A REGISTER FOR LINE %D2%A (D).%N/
2039 EF3001:: .ASCIZ /%A EXPECTED OR CORRECT VALUE: %03%A%N/
2040 EF3002:: .ASCIZ /%A ACTUAL OR MEASURED VALUE: %03%A%N/
2041 EF4401:: .ASCII /%N% DMA ADDRESS TEST SUCCESSFUL, BITS 0 TO %D2%A (D) TESTED/
2042 .ASCIZ / (%D2%A BITS).%N/
2043 EF6401:: .ASCIZ /%A %D2%A(D).%N/
2044 EF7801:: .ASCIZ /%T%A ON LINE %D2%A DECIMAL.%N/
2045 EF8401:: .ASCIZ /%A %T%A FOR LINE %D2%A(D) AFFECTS OTHER MODEM SIGNALS.%N/
2046 EF8402:: .ASCII /%A CHANGING %T%A FOR LINE %D2%A(D) AFFECTED /
2047 .ASCIZ /%T%A FOR LINE %D2%A(D).%N/
2048 EF9001:: .ASCIZ /%A UNEXPECTED %T%A FOUND IN RECEIVE CHAR FIFO:%N/
2049 EF9002:: .ASCIZ /%A CODE IS ASSOCIATED WITH LINE: %D2%A(D)%N/
2050 EF9003:: .ASCIZ /%A CODE IS: %03%A(0)%N/
2051 EF9004:: .ASCIZ /%A %T%A VALUE: %03%A(0)%N/
2052 EF9005:: .ASCIZ /%A %T%A VALUE: NONE%N/
2053 EF9006:: .ASCIZ /%A %T%A %D2%A(D)%N/
2054 EF9007:: .ASCIZ /%A CHARACTER RECEIVED WITH ERROR FLAG(S) SET ON LINE %D2%A%N/
2055 EF9008:: .ASCIZ /%A CHARACTER READ AS: %03%A%N/
2056 EF9009:: .ASCIZ /%A %T%A ERROR FLAG SET.%N/
2057 EF9010:: .ASCIZ /%A NUMBER OF ERRORS DETECTED ON LINE %D2%A(D) IS %D5%A(D)%N/
2058 EF9012:: .ASCII /%A LINE%D2%A ONLY %T%D5%A BYTES OF%D5%A BYTE/
2059 .ASCIZ / DATA PAT'N TX'D FROM LINE%D2%A%N/
2060 EF9013:: .ASCIZ /%A DATA PATTERN NOT COMPLETELY %T%A%N/
2061 EF9016:: .ASCIZ /%A UNEXPECTED %T%A FOR LINE %D2%A(D) IN FIFO AFTER RESET:%N/
2062 EF9017:: .ASCIZ /%A %T%A (WITH ERROR FLAGS) IS %06%A(0)%N/
2063 EF9018:: .ASCII /%A %T%A IN SELFTEST CODE FIFO SLOT FOR LINE %D2/
2064 .ASCIZ /%A(D) AFTER RESET.%N/
2065 EF9019:: .ASCIZ /%A %T%A %06%A(0)%N/
2066 EF9020:: .ASCIZ /%A TOO FEW TX.ACTIONS GENERATED ON LINE %D2%A%N/
2067 EF9101:: .ASCIZ /%N/
2068 EF9103:: .ASCIZ /%A ERROR CONDITION ON LINE %D2%A%N/
2069 EF9301:: .ASCIZ /%A %T%D2%A(D), BMP CODE REPORTED :%03%A(0)%N/
2070 EF9302:: .ASCIZ /%A OVERFLOW OCCURRED (MORE THAN 31 BMP CODES FOUND IN QUEUE)%N/
2071
2072 ;***** MESSAGE AREA *****
2073 DR00MG:: .ASCIZ /CSR/
2074 DR02MG:: .ASCIZ /RBUF/
2075 DR04MG:: .ASCIZ /LPR/

```

GLOBAL MESSAGE AREA

```
2076 010661 DR06MG:: .ASCIZ /STAT/
2077 010666 DR10MG:: .ASCIZ /LNCTRL/
2078 010675 DR12MG:: .ASCIZ /TBUFFAD1/
2079 010706 DR14MG:: .ASCIZ /TBUFFAD2/
2080 010717 DR16MG:: .ASCIZ /TBUFFCT/
2081 010727 EM0103:: .ASCIZ /DEVICE REGISTER ACCESS ERRORS/
2082 010765 EM0201:: .ASCIZ /MASTER RESET (PERFORM SELFTEST) TEST /
2083 011033 EM0202:: .ASCIZ / MASTER RESET BIT DID NOT CLEAR AFTER BOARD RESET./
2084 011117 .ASCIZ / WAITED 5 SECONDS. BIT DEFECTIVE OR FIRMWARE HUNG./
2085 011206 EM0203:: .ASCIZ / MASTER RESET BIT CLEAR IMMEDIATELY AFTER BOARD RESET./
2086 011276 .ASCIZ / BIT DEFECTIVE OR BOARD FIRMWARE ERROR./
2087 011351 EM0204:: .ASCIZ \ MR BIT WENT CLEAR WITHIN 1/2 SECOND OF BOARD RESET.\
2088 011437 .ASCIZ / BIT DEFECTIVE OR SELFTEST WAS (INCORRECTLY) SKIPPED./
2089 011530 EM0301:: .ASCIZ /MASTER RESET (SKIP SELFTEST) TEST /
2090 011573 EM0302:: .ASCIZ / MR BIT CLR WITHIN 10 MILLISECOND AFTER BOARD RESET./
2091 011660 .ASCIZ / BIT DEFECTIVE OR BOARD FIRMWARE ERROR./
2092 011733 EM0303:: .ASCIZ \ MR BIT WENT CLEAR 1/5 TO 5 SECONDS AFTER RESET.\
2093 012015 .ASCIZ / SELFTEST DID NOT GET SKIPPED (SHOULD HAVE BEEN SKIPPED)./
2094 012112 EM0401:: .ASCIZ /RBUF REGISTER RX CHARACTER FIELD TEST /
2095 012161 EM0402:: .ASCIZ / IMPROPER CODE FOUND IN RX FIFO AFTER DUT RESET./
2096 012243 .ASCIZ / EXPECTED: SELFTEST CODE, ACTUAL: IMPROPER CODE./
2097 012331 EM0501:: .ASCIZ /RBUF REGISTER ERROR FLAGS FIELD TEST /
2098 012377 EM0502:: .ASCIZ / RX ERROR FLAG(S) FOUND CLEAR ON SELFTEST CODE./
2099 012460 .ASCIZ / EXPECTED: ALL ERROR FLAGS SET, ACTUAL: FLAG(S) CLEAR./
2100 012553 EM0509:: .ASCIZ /SET/
2101 012557 EM0525:: .ASCIZ / RX INTERRUPT(S) RECEIVED WITH RX INTERRUPTS DISABLED./
2102 012647 EM0526:: .ASCIZ / TX INTERRUPT(S) RECEIVED WITH TX INTERRUPTS DISABLED./
2103 012737 EM0601:: .ASCIZ /CSR RX.DATA.AVAIL BIT TEST /
2104 012773 EM0602:: .ASCIZ / RX.DATA.AVAIL BIT FOUND CLEAR AFTER RESET COMPLETION./
2105 013063 .ASCIZ / EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO./
2106 013153 EM0603:: .ASCIZ / RX.DATA.AVAIL BIT COULD NOT BE CLEARED BY PURGING FIFO./
2107 013245 .ASCIZ / 600 CHARS READ FROM FIFO WITHOUT R.D.A BIT CLEARING./
2108 013336 EM0701:: .ASCIZ /RBUF RX.DATA.VALID BIT TEST /
2109 013373 EM0702:: .ASCIZ / RX.DATA.VALID BIT FOUND CLEAR AFTER RESET COMPLETION./
2110 013463 .ASCIZ / EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO./
2111 013553 EM0703:: .ASCIZ / RX.DATA.VALID BIT COULD NOT BE CLEARED BY PURGING FIFO./
2112 013645 .ASCIZ / 600 CHARS READ FROM FIFO WITHOUT R.D.V BIT CLEARING./
2113 013736 EM0801:: .ASCIZ /RBUF RX.LINE.NUMBER FIELD TEST /
2114 013776 EM0802:: .ASCIZ / LINE NUMBER IS WRONG ON A SELFTEST CODE./
2115 014051 EM0901:: .ASCIZ /CHECK FOR BMP_CODES TEST/
2116 014102 EM0902:: .ASCIZ /UNEXPECTED BMP_CODES FOUND./
2117 014136 EM1001:: .ASCIZ /DIAGNOSTIC FAIL (SKP SELFTEST) TEST/
2118 014202 EM1002:: .ASCIZ / SKIP SELF-TEST TOOK TOO LONG TO COMPLETE, > 50 MS./
2119 014267 EM1003:: .ASCIZ / SKIP SELF-TEST COMPLETED TOO SOON, < 10 MS./
2120 014345 EM1101:: .ASCIZ /SKIP SELF-TEST TEST/
2121 014371 EM1201:: .ASCIZ /SELF-TEST TEST/
2122 014410 EM1202:: .ASCIZ / SELF-TEST TOOK TOO LONG TO COMPLETE, > 3 SECONDS./
2123 014474 EM1203:: .ASCIZ \ SELF-TEST COMPLETED TOO SOON, < 1/2 SECOND.\
2124 014552 EM1204:: .ASCIZ / SELF-TEST DID NOT EXECUTE/
2125 014606 EM1205:: .ASCIZ / DIAG_FAIL BIT BAD/
2126 014632 EM1301:: .ASCIZ /FAIL SELF-TEST TEST/
2127 014656 EM1302:: .ASCIZ / SELF-TEST ERROR REPORTING BAD/
2128 014715 EM1401:: .ASCIZ /ROM VERSION_NUMBER TEST/
2129 014745 EM1402:: .ASCIZ / FIFO EMPTY, ONE OR MORE ROM VERSION NUMBERS MISSING/
2130 015033 EM1403:: .ASCIZ / ROM VERSION_NUMBER FOUND OUT OF SEQUENCE/
2131 015106 EM1404:: .ASCIZ / ONE OR MORE ROM VERSION_NUMBERS MISSING/
2132 015160 EM1405:: .ASCIZ / PROC 1/
```

GLOBAL MESSAGE AREA

```

2133 015173 EM1406:: .ASCIZ / PROC_2/
2134 015206 EM1407:: .ASCIZ /NOT FOUND/
2135 015220 EM1408:: .ASCIZ /FOUND/
2136 015226 EM1601:: .ASCIZ /TIMEOUT OCCURRED WAITING FOR MASTER RESET TO CLEAR/
2137 015311 EM1604:: .ASCIZ \DEVICE REGISTER WORD READ/WRITE TEST \
2138 015357 EM1701:: .ASCIZ \DEVICE REGISTER WORD READ/MODIFY/WRITE TEST \
2139 015434 EM1801:: .ASCIZ \DEVICE REGISTER BYTE READ/WRITE TEST \
2140 015502 EM1901:: .ASCIZ \DEVICE REGISTER BYTE READ/MODIFY/WRITE TEST \
2141 015557 EM2001:: .ASCIZ /DEVICE STAT REGISTER ID BIT TEST /
2142 015621 EM2002:: .ASCIZ /ID BIT BAD. EXPECTED: CLEAR, ACTUAL: SET./
2143 015674 EM2101:: .ASCIZ \NO TX_DATA_VALID/NO TX_ACTION TEST\
2144 015737 EM2102:: .ASCIZ / TX_ACTION FOUND AFTER INVALID DATA WORD WRITTEN TO LINE: /
2145 016033 EM2201:: .ASCIZ \TX_DATA_VALID/TX_ACTION TEST\
2146 016070 EM2202:: .ASCIZ / NO TX_ACTION FOUND AFTER VALID DATA WORD TX'D ON LINE: /
2147 016162 EM2203:: .ASCIZ / INCORRECT LINE NUMBER FOUND WITH TX_ACT AFTER DATA TX'D ON LINE : /
2148 016267 EM2301:: .ASCIZ /TX_ENABLE (INACTIVE) BIT TEST/
2149 016325 EM2302:: .ASCIZ / TX_ENABLE BIT BAD ON LINE: /
2150 016363 EM2401:: .ASCIZ /TX_ENABLE (ACTIVE) BIT TEST/
2151 016417 EM2601:: .ASCIZ /RECEIVE INTERRUPT TEST /
2152 016447 EM2602:: .ASCIZ / NO RX INT GENERATED (DATA_VALID SET, RX INTS ENABLED)./
2153 016540 EM2603:: .ASCIZ / NO RX INT GENERATED (NO CODES IN FIFO AFTER RESET)./
2154 016626 EM2604:: .ASCIZ / NO RX INT GENERATED (RX_DATA_AVAIL CLR, RX INTS ENABLED)./
2155 016722 EM2605:: .ASCIZ / RX INTERRUPT GENERATED WITH RX_DATA_AVAIL CLEAR./
2156 017005 EM2606:: .ASCIZ /TRANSMIT INTERRUPT TEST ERROR:/
2157 017044 EM2607:: .ASCIZ / TX_ACTION SET REPEATEDLY AFTER BOARD RESET, NO DATA SENT./
2158 017140 EM2608:: .ASCIZ / TX_ACTION STUCK SET AFTER BOARD RESET./
2159 017211 EM2609:: .ASCIZ / TX_INTERRUPT GENERATED WITH TX_ACTION CLEAR./
2160 017270 EM2610:: .ASCIZ / NO TX INTERRUPT WITH TX_ACTION SET AND TX INTS ENABLED./
2161 017362 EM2611:: .ASCIZ / TX_ACTION NOT SET AFTER CHARS SENT ON ALL LINES./
2162 017445 EM3001:: .ASCIZ /INTERRUPT BR LEVEL TEST /
2163 017476 EM3002:: .ASCIZ / NO RX_DATA_AVAIL FROM SELFTEST CODES IN FIFO AFTER RESET./
2164 017572 EM3003:: .ASCIZ / TX INTERRUPT GENERATED AT WRONG BR LEVEL:/
2165 017646 EM3004:: .ASCIZ / RX INTERRUPT GENERATED AT WRONG BR LEVEL:/
2166 017722 EM3005:: .ASCIZ / TX INTERRUPT GIVEN PRECEDENCE OVER SIMULTANEOUS RX INT./
2167 020014 EM3101:: .ASCIZ /DIAGNOSTIC FIELD (BMP) TEST/
2168 020050 EM3102:: .ASCIZ / DIAGNOSTIC FIELD BAD ON LINE: /
2169 020111 EM4001:: .ASCIZ /DMA_START BIT TEST/
2170 020134 EM4002:: .ASCIZ /DMA_START BIT BAD ON LINE: /
2171 020170 EM4101:: .ASCIZ /DMA_ABORT BIT TEST/
2172 020213 EM4102:: .ASCIZ /DMA_ABORT BIT BAD ON LINE: /
2173 020247 EM4103:: .ASCIZ /DMA_START BIT FOUND SET AFTER DMA ABORTED ON LINE: /
2174 020333 EM4401:: .ASCIZ /DMA ADDRESS TEST /
2175 020355 EM4402:: .ASCIZ /NO SUITABLE ADDR FOUND,TEST ABANDONED/
2176 020423 EM4403:: .ASCIZ /**HOST FAILURE**WRITE FAILED TO AN ADDR WHICH HAD BEEN READ/
2177 020520 EM4404:: .ASCIZ /NO ACTIVE LINES,TEST ABANDONED/
2178 020557 EM4405:: .ASCIZ /DMA_START BIT FOUND SET BEFORE DMA INITIATED,TEST ABANDONED/
2179 020653 EM4406:: .ASCIZ /TIME-OUT OCCURED WAITING FOR DMA TO FINISH/
2180 020727 EM4407:: .ASCIZ /TOO FEW CHARACTERS FOUND IN THE RXFIFO,DMA FAILED/
2181 021011 EM4408:: .ASCIZ /TOO MANY BMP CODES FOUND IN RXFIFO/
2182 021054 EM4409:: .ASCIZ /BAD BITS BETWEEN BITS 0 AND /
2183 021111 EM4410:: .ASCIZ /RXFIFO FAILED TO PURGE/
2184 021140 EM4411:: .ASCIZ /**HOST FAILURE**WRITE ATTEMPT FAILED/
2185 021205 EM5101:: .ASCIZ /IAUTO (INACTIVE) TEST/
2186 021233 EM5102:: .ASCIZ /IAUTO BIT FOUND SET ON LINE: /
2187 021271 EM5103:: .ASCIZ /IAUTO BIT BAD ON LINE: /
2188 021321 EM5201:: .ASCIZ /IAUTO (ACTIVE) TEST/
2189 021345 EM5202:: .ASCIZ /IAUTO BIT FOUND CLR ON LINE: /

```

GLOBAL MESSAGE AREA

```

2190 021403 EM5301:: .ASCIZ /FIFO VALID DATA TEST/
2191 021430 EM5302:: .ASCIZ /FIFO BAD DATA FIELD CORRUPTED, TEST USED LINE:/
2192 021507 EM5303:: .ASCIZ /BMP CODE FOUND IN FIFO, TEST INVAILEDATED/
2193 021560 EM5401:: .ASCIZ \FIFO 3/4 ALARM (INACTIVE) TEST\
2194 021617 EM5402:: .ASCIZ /FIFO BAD, ALARM SIGNAL DEFECTIVE/
2195 021660 EM5501:: .ASCIZ \FIFO 3/4 ALARM (ACTIVE) TEST\
2196 021715 EM5601:: .ASCIZ \FIFO 3/4 ALARM (ACTIVE/INACTIVE) TEST\
2197 021763 EM5701:: .ASCIZ \FIFO 1/2 LEVEL (ACTIVE/INACTIVE) TEST\
2198 022031 EM6401:: .ASCIZ /BREAK GENERATION TEST /
2199 022060 EM6402:: .ASCIZ / BREAK NOT RECEIVED ON LINE(S):/
2200 022121 EM6601:: .ASCIZ /NO OVERRUN ERROR TEST/
2201 022147 EM6602:: .ASCIZ / OVERRUN ERROR REPORTED WHEN NONE FORCED/
2202 022221 EM6701:: .ASCIZ /OVERRUN ERROR TEST/
2203 022244 EM6702:: .ASCIZ / NO OVERRUN ERROR REPORTED, OVERRUN FORCED/
2204 022321 EM9001:: .ASCIZ /SINGLE CHARACTER MODE TEST /
2205 022355 EM9003:: .ASCIZ /MODEM STATUS CODE/
2206 022377 EM9004:: .ASCIZ /SELFTEST CODE/
2207 022415 EM9006:: .ASCIZ /CHARACTER RECEIVED ON INACTIVE LINE, LINE:/
2208 022470 EM9007:: .ASCIZ /UNEXPECTED CHAR RECEIVED AFTER RX COMPLETE ON LINE/
2209 022553 EM9008:: .ASCIZ /RECEIVED CHAR MISCOMPARE AGAINST TX DATA ON LINE/
2210 022634 EM9009:: .ASCIZ /EXPECTED OR CORRECT/
2211 022660 EM9010:: .ASCIZ /ACTUAL OR MEASURED /
2212 022704 EM9011:: .ASCIZ /OVERRUN/
2213 022714 EM9012:: .ASCIZ /FRAMING/
2214 022724 EM9013:: .ASCIZ /PARITY/
2215 022733 EM9014:: .ASCIZ /SUMMARY REPORTS FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:/
2216 023027 EM9015:: .ASCIZ /TRANSMITTED/
2217 023043 EM9016:: .ASCIZ /RECV'D/
2218 023052 EM9017:: .ASCII / FIFO WILL NOT PURGE (DATA.VALID STUCK SET),/
2219 023127 .ASCIZ / REMAINDER OF TEST SKIPPED./
2220 023163 EM9018:: .ASCIZ /NO CODE/
2221 023173 EM9019:: .ASCIZ /NON-SELFTEST/
2222 023210 EM9020:: .ASCIZ /SELFTEST ERROR CODE/
2223 023234 EM9022:: .ASCIZ /DATA CHARACTER/
2224 023257 EM9023:: .ASCIZ /MODEM STATUS CODE/
2225 023275 EM9024:: .ASCIZ /SELFTEST CODE/
2226 023313 EM9025:: .ASCIZ /MORE THAN TWICE THE EXPECTED NUMBER OF CHARACTERS RECEIVED./
2227 023407 EM9026:: .ASCIZ / LPR CONTENTS: /
2228 023433 EM9027:: .ASCIZ /EXTRA CHAR RECEIVED WITHIN DATA PATTERN ON LINE/
2229 023513 EM9028:: .ASCIZ /SINGLE CHAR MISSING FROM RECEIVED DATA ON LINE/
2230 023572 EM9030:: .ASCIZ /*A (NO TX COMPLETION INTERRUPTS RECEIVED)*N/
2231 023647 EM9101:: .ASCIZ /DMA TRANSMISSION MODE TEST /
2232 023703 EM9102:: .ASCIZ /DMA_START BIT SET AFTER RESET OR TX.ACTION ON LINE(S):/
2233 023772 EM9104:: .ASCIZ / UNEXPECTED DATA FOUND IN FIFO FROM LINE: /
2234 024046 EM9301:: .ASCIZ /BMP CODE REPORT/
2235 024066 EM9302:: .ASCIZ /BMP CODE FOUND IN TEST /
2236 024116 EM9303:: .ASCIZ /THE LAST BMP CODE WAS FOUND IN TEST /
2237 024163 EM9304:: .ASCIZ /UNEXPECTED BMP CODES FOUND DURING THIS PASS/

```

2238

2239

2240

.EVEN

.LIST BIN

GLOBAL MESSAGE AREA

2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257

```
;*****  
;  
;          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 -

```

2259 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0101 -
2260 *****
2261 ;* This is an error reporting subroutine which prints additional error
2262 ;* information if an error is detected in TEST 1 (Register Address
2263 ;* Access Test). This subroutine reports the type of access (Read or
2264 ;* Write or both) which caused a bus time-out trap (004 trap).
2265 ;* A message indicating that the DHV11-M may be at the wrong Q-bus address
2266 ;* is also printed.
2267 ;*
2268 ;* INPUTS:      R5 - Error flag word.
2269 ;*              If bit 0 is set, a read error occurred.
2270 ;*              If bit 1 is set, a write error occurred.
2271 ;*
2272 ;* OUTPUTS:     Messages are printed at the operator console.
2273 ;*
2274 ;* CALLING SEQUENCE: Include the label "ER0101" as the message pointer
2275 ;*                   parameter in the DRS error report macro call.
2276 ;*
2277 ;* COMMENTS:
2278 ;*
2279 ;* SUBORDINATE ROUTINES USED: None.
2280 *****
2281
2282 024240      BGNMSG  ER0101
2283 024240      SAVE
2284 024240 004537 005474      JSR      ;SAVE THE GPR CONTENTS.
2285 024244 032705 000001      ;CALL REGISTER SAVE SUBRT.
2286 024250 001410      BIT      #BIT0,R5
2287 024252      BEQ      2$
2288 024252 012746 024344      ;TEST FOR READ ERROR.
2289 024252 012746 000001      ;SKIP READ ERROR MSG IF NO READ ERROR.
2290 024252 010600      PRINTB  #MSG1
2291 024252 012746 024344      ;PRINT READ ERROR MESSAGE.
2292 024252 012746 000001      MOV      #MSG1, (SP)
2293 024252 010600      MOV      #1, -(SP)
2294 024252 104414      MOV      SP,R0
2295 024252 062706 000004      TRAP    C$PNTB
2296 024252 032705 000002      ADD     #4,SP
2297 024272 001410      2$: BIT      #BIT1,R5
2298 024272 001410      BEQ      4$
2299 024272 012746 024422      ;TEST FOR WRITE ERROR.
2300 024272 012746 000001      ;SKIP WRITE ERROR MSG IF NO WRITE ERROR.
2301 024272 010600      PRINTB  #MSG2
2302 024272 012746 024422      ;PRINT WRITE ERROR MESSAGE.
2303 024272 012746 000001      MOV      #MSG2, -(SP)
2304 024272 010600      MOV      #1, (SP)
2305 024272 104414      MOV      SP,R0
2306 024272 062706 000004      TRAP    C$PNTB
2307 024272 032705 000002      ADD     #4,SP
2308 024272 001410      4$: PRINTX  #MSG3
2309 024272 012746 024501      ;SUGGEST THAT DHV11-M MAY BE AT WRONG ADDRESS.
2310 024272 012746 000001      MOV      #MSG3, -(SP)
2311 024272 010600      MOV      #1, -(SP)
2312 024272 104415      MOV      SP,R0
2313 024272 062706 000004      TRAP    C$PNTX
2314 024272 032705 000002      ADD     #4,SP
2315 024300      PASS
2316 024300 004736      JSR      ;RESTORE THE GPR CONTENTS.
2317 024300      ENDMSG      PC,0(SP).
2318 024300      ;RETURN TO PREG05 SUBRT.
2319 024300
2320 024300      L10002:
2321 024300      TRAP    C$MSG
2322 024300
2323 024342
2324 024342 104423
2325 024344 045 101 102 MSG1:: .ASCIZ /#ABUS TIME-OUT TRAP CAUSED BY READ ATTEMPT.#N/

```


GLOBAL ERROR REPORTING ROUTINE

- ER0101 -

	024347	125	123	040	
	024352	124	111	115	
	024355	105	055	117	
	024360	125	124	040	
	024363	124	122	101	
	024366	120	040	103	
	024371	101	125	123	
	024374	105	104	040	
	024377	102	131	040	
	024402	122	105	101	
	024405	104	040	101	
	024410	124	124	105	
	024413	115	120	124	
	024416	056	045	116	
	024421	000			
2296	024422	045	101	102	MSG2:: .ASCIZ /#ABUS TIME OUT TRAP CAUSED BY WRITE ATTEMPT.#N/
	024425	125	123	040	
	024430	124	111	115	
	024433	105	055	117	
	024436	125	124	040	
	024441	124	122	101	
	024444	120	040	103	
	024447	101	125	123	
	024452	105	104	040	
	024455	102	131	040	
	024460	127	122	111	
	024463	124	105	040	
	024466	101	124	124	
	024471	105	115	120	
	024474	124	056	045	
	024477	116	000		
2297	024501	045	101	104	MSG3:: .ASCIZ /#ADHV11-M MAY BE AT THE WRONG Q-BUS ADDRESS.#N#N/
	024504	110	126	061	
	024507	061	055	115	
	024512	040	115	101	
	024515	131	040	102	
	024520	105	040	101	
	024523	124	040	124	
	024526	110	105	040	
	024531	127	122	117	
	024534	116	107	040	
	024537	121	055	102	
	024542	125	123	040	
	024545	101	104	104	
	024550	122	105	123	
	024553	123	056	045	
	024556	116	045	116	
	024561	000			

2298
2299 .EVEN

GLOBAL ERROR REPORTING ROUTINE

- ER0201

```

2301 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0201
2302 ;*****
2303 ;* This is an error reporting subroutine which prints 2 contiguous
2304 ;* ASCII error messages. The address of the first message is passed
2305 ;* as an input parameter and the address of the second is found by
2306 ;* searching for the end of the first message.
2307 ;*
2308 ;* INPUTS: R1 Address of the first message to print.
2309 ;*
2310 ;* OUTPUTS: A messages is printed at the operator console.
2311 ;*
2312 ;* CALLING SEQUENCE: Load the address of the first message in R1.
2313 ;* Include the label "ER0201" as the message pointer
2314 ;* parameter in the Diag Super error report macro call.
2315 ;*
2316 ;* COMMENTS: The message is printed as Basic error information.
2317 ;* The second message should follow the first one in the program
2318 ;* memory. Each message should be defined using .ASCIIZ
2319 ;*
2320 ;* SUBORDINATE ROUTINES USED: None.
2321 ;*****
2322
2323 024562 BGNMSG ER0201
2324 024562 ER0201::
2325 024562 004537 005474 SAVE JSR ;SAVE THE GPR CONTENTS.
2326 024566 010102 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2327 024570 105722 2$: MOV R1,R2 ;CHECK FOR A ZERO BYTE (END OF MESSAGE).
2328 024572 001376 TSTB (R2)+ ;LOOP UNTIL NEXT MESSAGE IS FOUND.
2329 BNE 2$
2330 024574 PRINTB #EF0503,R1 ;PRINT THE FIRST MESSAGE.
2331 024574 010146 MOV R1,(SP)
2332 024576 012746 005630 MOV #EF0503,(SP)
2333 024602 012746 000002 MOV #2,(SP)
2334 024606 010600 MOV SP,R0
2335 024610 104414 TRAP C$PNTB
2336 024612 062706 000006 ADD #6,SP
2337 024616 PRINTB #EF0503,R2 ;PRINT THE SECOND MESSAGE.
2338 024616 010246 MOV R2,-(SP)
2339 024620 012746 005630 MOV #EF0503,(SP)
2340 024624 012746 000002 MOV #2,-(SP)
2341 024630 010600 MOV SP,R0
2342 024632 104414 TRAP C$PNTB
2343 024634 062706 000006 ADD #6,SP
2344 024640 PASS JSR ;RESTORE THE GPR CONTENTS.
2345 024640 004736 PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2346 024642 ENDMSG
2347 024642 104423 L10003: TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

ER0503 -

```

2337 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0503 -
2338 ;*****
2339 ;* This is an error reporting subroutine which prints an additional error
2340 ;* message whose address is passed as an input parameter.
2341 ;*
2342 ;* INPUTS: R1 - Address of the message to print.
2343 ;*
2344 ;* OUTPUTS: A messages is printed at the operator console.
2345 ;*
2346 ;* CALLING SEQUENCE: Load the address of the message in R1.
2347 ;* Include the label "ER0503" as the message pointer
2348 ;* parameter in the Diag Super error report macro call.
2349 ;*
2350 ;* COMMENTS: The message is printed as Bas'c error information.
2351 ;*
2352 ;* SUBORDINATE ROUTINES USED: None.
2353 ;*****
2354
2355 024644 BGNMSG ER0503
2356 024644 ER0503::
2357 024644 PRINTB #EF0503,R1 ;PRINT THE MESSAGE.
2358 024644 010146 MOV R1,-(SP)
2359 024646 012746 005630 MOV #EF0503,-(SP)
2359 024652 012746 000002 MOV #2,(SP)
2359 024656 010600 MOV SP,R0
2359 024660 104414 TRAP C$PNTB
2359 024662 062706 000006 ADD #6,SP
2358
2359 024666 ENDMSG
2359 024666 L10004:
2359 024666 104423 TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER0504 -

```

2361 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER0504 -
2362 ;*****
2363 ;* This is an error reporting subroutine which prints additional error
2364 ;* messages when illegal interrupts are received.
2365 ;*
2366 ;* INPUTS: R1 Address of the message to print.
2367 ;* R2 Number of illegal interrupts received.
2368 ;*
2369 ;* OUTPUTS: Messages are printed at the operator console.
2370 ;*
2371 ;* CALLING SEQUENCE: Load the address of the message in R1.
2372 ;* Load the number of illegal ints in R2.
2373 ;* Include the label "ER0504" as the message pointer
2374 ;* parameter in the Diag Super error report macro call.
2375 ;*
2376 ;* COMMENTS:
2377 ;*
2378 ;* SUBORDINATE ROUTINES USED: None.
2379 ;*****
2380
2381 024670 BGNMSG ER0504
2382 024670
2383 024670
2384 024670 010146
2385 024672 012746 005630
2386 024676 012746 000002
2387 024702 010600
2388 024704 104414
2389 024706 062706 000006
2390 024712
2391 024714 010246
2392 024716 012746 005635
2393 024720 012746 000002
2394 024724 010600
2395 024726 104415
2396 024730 062706 000006
2397
2398 024734
2399 024734
2400 024734 104423

```

```

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

- ER1601 -

SEQ 0058

```

2388 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER1601 -
2389 ;*****
2390 ;* This an error reporting subroutine which prints additional error
2391 ;* information if an error is detected in one of the device register
2392 ;* access tests.
2393 ;* Th's subroutine reports the actual and expected from the device
2394 ;* register(s) which is(are) in faulty.
2395 ;*
2396 ;* INPUTS: R1 - Actual data (unused bits set to 0).
2397 ;* R2 - Expected data (unused bits set to 0).
2398 ;* R3 - Offset (in bytes) to the register being tested.
2399 ;* R5 - Line number of register being tested.
2400 ;* RMatBB - Label at base of register message address table.
2401 ;*
2402 ;* OUTPUTS: Messages are printed at the operators console.
2403 ;*
2404 ;* CALLING SEQUENCE: Include the lable "ER1601" as the message pointer
2405 ;* parameter in the DRS error report macro call.
2406 ;*
2407 ;* COMMENTS:
2408 ;*
2409 ;* SUBORDINATE ROUTINES CALLED: NONE
2410 ;*****
2411 024736 BGNMSG ER1601
2412 024736
2412 ER1601::
2413 024736 016304 002340 MOV RMatBB(R3),R4 ;FETCH ADDRESS OF REGISTER NAME MESSAGE.
2414
2415 024742 PRINTB #EF1604,R4,R5 ;REPORT BASIC MESSAGE (REG NAME AND LINE #).
2416 024742 010546 MOV R5,-(SP)
2417 024744 010446 MOV R4,-(SP)
2418 024746 012746 006177 MOV #EF1604,-(SP)
2419 024752 012746 000003 MOV #3,-(SP)
2420 024756 010600 MOV SP,R0
2421 024760 104414 TRAP C:PNTB
2422 024762 062706 000010 ADD #10,SP
2423 024766 PRINTX #EF1602,R2 ;PRINT THE EXPECTED DATA.
2424 024766 010246 MOV R2,-(SP)
2425 024770 012746 006073 MOV #EF1602,-(SP)
2426 024774 012746 000002 MOV #2,-(SP)
2427 025000 010600 MOV SP,R0
2428 025002 104415 TRAP C:PNTX
2429 025004 062706 000006 ADD #6,SP
2430 025010 PRINTX #EF1603,R1 ;PRINT THE ACTUAL DATA.
2431 025010 010146 MOV R1,-(SP)
2432 025012 012746 006135 MOV #EF1603,-(SP)
2433 025016 012746 000002 MOV #2,-(SP)
2434 025022 010600 MOV SP,R0
2435 025024 104415 TRAP C:PNTX
2436 025026 062706 000006 ADD #6,SP
2437 025032 ENDMSG
2438 025032
2439 025032 104423 L10006:
2440 TRAP C:MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER1603

```

2420 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER1603 -
2421 ;*****
2422 ;* This error reporting routine is used to print out a basic error
2423 ;* message, along with a message informing the operator which test is
2424 ;* about to be aborted.
2425 ;*
2426 ;* INPUTS: R1 - Contains the address of the message to be printed.
2427 ;* ERRMSG - Contains the address of the message that indicates
2428 ;* the test that is being performed, eg DMA, BREAK etc.
2429 ;*
2430 ;* OUTPUTS: Messages are printed at the operators console.
2431 ;* "testname TEST ABORTED"
2432 ;*
2433 ;* CALLING SEQUENCE: Include the label "ER1603" as the message pointer
2434 ;* parameter in the DRS error report macro call.
2435 ;*
2436 ;* COMMENTS:
2437 ;*
2438 ;*
2439 ;* SUBORDINATE ROUTINES CALLED: None.
2440 ;*****
2441 025034 BGNMSG ER1603
2442 025034 ER1603::
2443 025034 004537 005474 SAVE JSR ;SAVE THE CONTENTS OF THE GPRS.
2444 025040 PRINTB #EF0503,R1 ;PRINT BASIC MESSAGE ON OPERATORS CONSOLE.
2445 025040 010146 MOV R1,-(SP)
2446 025042 012746 005630 MOV #EF0503,-(SP)
2447 025046 012746 000002 MOV #2,-(SP)
2448 025052 010600 MOV SP,R0
2449 025054 104414 TRAP C#PNTB
2450 025056 062706 000006 ADD #6,SP
2451 025062 013702 005470 MOV ERRMSG,R2 ;GET THE "TEST MESSAGE".
2452 025066 010246 PRINTB #EF1601,R2 ;PRINT "TEST ABORTED" MESSAGE.
2453 025070 012746 006047 MOV R2,-(SP)
2454 025074 012746 000002 MOV #EF1601,-(SP)
2455 025100 010600 MOV #2,-(SP)
2456 025102 104414 MOV SP,R0
2457 025104 062706 000006 TRAP C#PNTB
2458 025110 PASS ;RESTORE THE CONTENTS OF THE GPRS.
2459 025110 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
2460 025112 ENDMSG
2461 025112 104423 L10007: TRAP C#MSG

```


GLOBAL ERROR REPORTING ROUTINE

- ER3001 -

```

2452 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER3001 -
2453 ;*****
2454 ;* This is an error reporting subroutine which is intended for use in the
2455 ;* Interrupt BR Level Test. It reports additional information when an
2456 ;* interrupt has occurred at the wrong BR level.
2457 ;*
2458 ;* INPUTS: R1 - Address of message to print first.
2459 ;* R4 - BR level at which the int request occurred.
2460 ;* R5 - Expected or correct BR level for the DUT.
2461 ;*
2462 ;* OUTPUTS: A messages is printed at the operator console.
2463 ;*
2464 ;* CALLING SEQUENCE: Include the label "ER3001" as the message pointer
2465 ;* parameter in the Diag Super error report macro call.
2466 ;*
2467 ;* COMMENTS: The message is printed as Basic and Extended error information.
2468 ;*
2469 ;* SUBORDINATE ROUTINES USED: None.
2470 ;*****
2471
2472 025114 BGNMSG ER3001
2473 025114 ER3001::
2474 025114 PRINTB #EF0503,R1 ;PRINT THE FIRST LINE OF THE MESSAGE.
2475 025114 010146 MOV R1,-(SP)
2476 025116 012746 005630 MOV #EF0503,-(SP)
2477 025122 012746 000002 MOV #2,-(SP)
2478 025126 010600 MOV SP,R0
2479 025130 104414 TRAP C$PNTB
2480 025132 062706 000000 ADD #6,SP
2481 025136 PRINTX #EF3001,R5 ;REPORT EXPECTED BR LEVEL.
2482 025136 010546 MOV R5,-(SP)
2483 025140 012746 006274 MOV #EF3001,-(SP)
2484 025144 012746 000002 MOV #2,-(SP)
2485 025150 010600 MOV SP,R0
2486 025152 104415 TRAP C$PNTX
2487 025154 062706 000006 ADD #6,SP
2488 025160 PRINTX #EF3002,R4 ;REPORT ACTUAL BR LEVEL.
2489 025160 010446 MOV R4,-(SP)
2490 025162 012746 006343 MOV #EF3002,-(SP)
2491 025166 012746 000002 MOV #2,-(SP)
2492 025172 010600 MOV SP,R0
2493 025174 104415 TRAP C$PNTX
2494 025176 062706 000006 ADD #6,SP
2495
2496 025202 ENDMSG
2497 025202
2498 025202 104423 L10010: TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER6401 -

```

2480 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER6401 -
2481 ;*****
2482 ;* This is an error reporting subroutine which prints additional error
2483 ;* information after the error message header.
2484 ;* This subroutine is passed a GPR containing flags which indicate
2485 ;* the line(s) for which the error condition should be reported.
2486 ;*
2487 ;* INPUTS: R1 Address of the message to be printed by this routine.
2488 ;* R5 - Contains the error flags, (1 flag per line).
2489 ;*
2490 ;* OUTPUTS: Messages are printed at the operator console.
2491 ;*
2492 ;* CALLING SEQUENCE: Load the address of the message in R1.
2493 ;* Include the label "ER6401" as the message pointer
2494 ;* parameter in the Diag Super error report macro call.
2495 ;*
2496 ;* COMMENTS: The output format of this message is:
2497 ;* TEXT MESSAGE
2498 ;* @nn
2499 ;* @nn
2500 ;*
2501 ;* Where each "@nn" is the number of a line with the error.
2502 ;*
2503 ;* SUBORDINATE ROUTINES USED: None.
2504 ;*****
2505
2506 025204 BGNMSG ER6401
2507 025204 SAVE
2508 025204 004537 005474 JSR ER6401::
2509 025210 005002 ;SAVE THE CONTENTS OF THE GPRS.
2510 025212 012703 000010 R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2511 025216 CLR R2 ;CLEAR LINE NUMBER TO ZERO.
2512 025216 010146 MOV @NUMLNS,R3 ;SET UP MAX LINE COUNT.
2513 025220 012746 005630 PRINTB @EF0503,R1 ;PRINT MESSAGE.
2514 025224 012746 000002 MOV R1,-(SP)
2515 025230 010600 MOV @EF0503,-(SP)
2516 025232 104414 MOV @2,-(SP)
2517 025234 062706 000006 MOV SP,R0
2518 025240 000241 TRAP C#PNTB
2519 025242 006205 ADD @6,SP
2520 025244 103011 CLC ;CLEAR CARRY.
2521 025246 010246 2$: ASR R5 ;SHIFT FLAG OUT INTO CARRY F.T.
2522 025250 012746 006527 BCC 4$ ;SKIP ERROR REPORT IF CLEAN.
2523 025254 012746 000002 PRINTB @EF6401,R2 ;PRINT MESSAGE.
2524 025260 010600 MOV R2,-(SP)
2525 025262 104414 MOV @EF6401,(SP)
2526 025264 062706 000006 MOV @2,-(SP)
2527 025270 005202 MOV SP,R0
2528 025272 020302 TRAP C#PNTB
2529 025274 001362 ADD @6,SP
2530 025276 004736 4$: INC R2 ;INCREMENT LINE COUNT.
2531 025300 025300 CMP R3,R2 ;CHECK IF MAX LINE COUNT EXCEEDED.
2532 025300 BNE 2$ ;LOOP IF NOT DONE.
2533 025300 PASS 60$: ;RESTORE THE SAVED CONTENTS OF THE GPRS.
2534 025300 JSR PC,@(SP) ;RETURN TO PREG05 SUBRT.
2535 025300 ENDMSG

```

L10011:

GLOBAL ERROR REPORTING ROUTINE

- ER6401 -

SEQ 0062

025300 104423

TRAP C\$MSG

GLOBAL ERROR REPORTING ROUTINE

- ER7801 -

```

2522 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER7801 -
2523 ;*****
2524 ;* This is an error reporting subroutine which prints an additional error
2525 ;* message whose address is passed as an input parameter. A line number
2526 ;* is included at the end of the message.
2527 ;*
2528 ;* INPUTS: R1 - Address of the message to print.
2529 ;* R3 - Number of line on which error occurred.
2530 ;*
2531 ;* OUTPUTS: A messages is printed at the operator console.
2532 ;*
2533 ;* CALLING SEQUENCE: Load the address of the message in R1.
2534 ;* Load the line number into R3.
2535 ;* Include the label "ER7801" as the message pointer
2536 ;* parameter in the Diag Super error report macro call.
2537 ;*
2538 ;* COMMENTS: The message is printed as Basic error information.
2539 ;*
2540 ;* SUBORDINATE ROUTINES USED: None.
2541 ;*****
2542
2543 025302 BGNMSG ER7801
2544 025302
2545 025302 PRINTB #EF7801,R1,R3 ;PRINT THE MESSAGE.
2546 025302 010346
2547 025304 010146
2548 025306 012746 006604
2549 025312 012746 000003
2550 025316 010600
2551 025320 104414
2552 025322 062706 000010
2553
2554 025326 ENDMSG
2555 025326
2556 025326 104423
2557
2558 ER7801::
2559 MOV R3,-(SP)
2560 MOV R1,-(SP)
2561 MOV #EF7801,-(SP)
2562 MOV #3,-(SP)
2563 MOV SP,R0
2564 TRAP C#PNTB
2565 ADD #10,SP
2566
2567 L10012:
2568 TRAP C#MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER9001 -

```

2549 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9001 -
2550 ;*****
2551 ;* This is an error reporting subroutine which reports an unexpected
2552 ;* code which has been found in the DUT CSR. This code can be a BMP
2553 ;* code, a Self-test code, or a Modem Status code.
2554 ;*
2555 ;* INPUTS: R1 - Address of message to print first.
2556 ;* R2 - Single byte code which has been read from the DUT.
2557 ;* R4 - Line number associated with the code.
2558 ;*
2559 ;* OUTPUTS: A messages is printed at the operator console.
2560 ;*
2561 ;* CALLING SEQUENCE: Include the label "ER9001" as the message pointer
2562 ;* parameter in the Diag Super error report macro call.
2563 ;*
2564 ;* COMMENTS: The message is printed as Basic and Extended error information.
2565 ;*
2566 ;* SUBORDINATE ROUTINES USED: None.
2567 ;*****
2568
2569 025330 BGNMSG ER9001
2570 025330
2571 025330 ER9001::
2572 025330 PRINTB #EF9001,R1 ;REPORT TYPE OF CODE FOUND.
2573 025330 010146 MOV R1,-(SP)
2574 025332 012746 007051 MOV #EF9001,-(SP)
2575 025336 012746 000002 MOV #2,-(SP)
2576 025342 010600 MOV SP,R0
2577 025344 104414 TRAP C#PNTB
2578 025346 062706 000006 ADD #6,SP
2579 025352 PRINTX #EF9002,R4 ;REPORT THE LINE NUMBER OF THE CODE.
2580 025352 010446 MOV R4,-(SP)
2581 025354 012746 007133 MOV #EF9002,-(SP)
2582 025360 012746 000002 MOV #2,-(SP)
2583 025364 010600 MOV SP,R0
2584 025366 104415 TRAP C#PNTX
2585 025370 062706 000006 ADD #6,SP
2586 025374 PRINTX #EF9003,R2 ;REPORT THE CODE WHICH WAS FOUND.
2587 025374 010246 MOV R2,-(SP)
2588 025376 012746 007212 MOV #EF9003,-(SP)
2589 025402 012746 000002 MOV #2,-(SP)
2590 025406 010600 MOV SP,R0
2591 025410 104415 TRAP C#PNTX
2592 025412 062706 000006 ADD #6,SP
2593 025416 ENDMSG
2594 025416
2595 025416 104423 L10013: TRAP C#MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER9002 -

SEQ 0065

```

2577 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9002 -
2578 ;*****
2579 ;* This is an error reporting subroutine which is intended for use in the
2580 ;* transmission and reception tests. It reports the type of error which
2581 ;* has occurred when incorrect data is received from the DUT. This
2582 ;* routine also reports the read and expected data values.
2583 ;*
2584 ;* INPUTS: R1 - Address of message to print first.
2585 ;* R2 - Data byte read from the DUT.
2586 ;* R3 - Line number multiplied by 2.
2587 ;* R4 - Expected data byte, bit 15 set if "NONE".
2588 ;*
2589 ;* OUTPUTS: A message is printed at the operator console.
2590 ;*
2591 ;* CALLING SEQUENCE: Include the label "ER9002" as the message pointer
2592 ;* parameter in the Diag Super error report macro call.
2593 ;*
2594 ;* COMMENTS: The message is printed as Basic and Extended error information.
2595 ;*
2596 ;* SUBORDINATE ROUTINES USED: PRTLPR.
2597 ;*****
2598
2599 025420 BGNMSG ER9002
2600 025420 ER9002::
2601 025420 006203 ASR R3 ;CALCULATE THE LINE NUMBER.
2602 025422 042702 177400 BIC #177400,R2 ;MASK OUT ALL BUT DATA IN READ CHAR.
2603 025426 PRINTB #EF9006,R1,R3 ;PRINT THE FIRST LINE OF THE MESSAGE.
2604 025426 010346 MOV R3,-(SP)
2605 025430 010146 MOV R1,-(SP)
2606 025432 012746 007334 MOV #EF9006,-(SP)
2607 025436 012746 000003 MOV #3,-(SP)
2608 025442 010600 MOV SP,R0
2609 025444 104414 TRAP C#PNTB
2610 025446 062706 000010 ADD #10,SP
2611 025452 PRINTX #EF9004,#EM9010,R2 ;PRINT ACTUAL DATA.
2612 025452 010246 MOV R2,-(SP)
2613 025454 012746 022660 MOV #EM9010,-(SP)
2614 025460 012746 007246 MOV #EF9004,-(SP)
2615 025464 012746 000003 MOV #3,-(SP)
2616 025470 010600 MOV SP,R0
2617 025472 104415 TRAP C#PNTX
2618 025474 062706 000010 ADD #10,SP
2619 025500 005704 TST R4 ;CHECK FOR "NONE" CODE SET IN EXPECTED DATA.
2620 025502 100414 BMI 2# ;BRANCH TO PRINT "NONE" MESSAGE IF FLAG SET.
2621 025504 PRINTX #EF9004,#EM9009,R4 ;PRINT EXPECTED DATA.
2622 025504 010446 MOV R4,-(SP)
2623 025506 012746 022634 MOV #EM9009,-(SP)
2624 025512 012746 007246 MOV #EF9004,-(SP)
2625 025516 012746 000003 MOV #3,-(SP)
2626 025522 010600 MOV SP,R0
2627 025524 104415 TRAP C#PNTX
2628 025526 062706 000010 ADD #10,SP
2629 025532 000412 BR 60# ;EXIT THIS ROUTINE.
2630 025534 PRINTX #EF9005,#EM9009 ;PRINT MESSAGE INDICATING NO EXPECTED DATA.
2631 025534 012746 022634 MOV #EM9009,-(SP)
2632 025540 012746 007303 MOV #EF9005,-(SP)

```

GLOBAL ERROR REPORTING ROUTINE

- ER9002

SEQ 0066

	025544	012746	000002
	025550	010600	
	025552	104415	
	025554	062706	000006
2610	025560	004737	032640
2611	025564		
	025564		
	025564	104423	

603:

JSR
ENDMSG

PC.PRTLPR

```
ADD  
;PRINT CONTENTS OF THE LPR REGISTER.
```

```
MOV    #2, (SP)
MOV    SP, R0
TRAP   C$PNTX
ADD    #6, SP
```

L10014:

TRAP CMSG

GLOBAL ERROR REPORTING ROUTINE

- ER9003 -

```

2613 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9003 -
2614 ;*****
2615 ;* This is an error reporting subroutine which is intended for use in the
2616 ;* transmission and reception tests. It reports error information when
2617 ;* a character has been read from the DUT with an error flag or flags
2618 ;* set (ie. over-run, framing, or parity flag).
2619 ;*
2620 ;* INPUTS: R2 Data byte read from the DUT, including error flags.
2621 ;* R3 Line number multiplied by 2.
2622 ;*
2623 ;* OUTPUTS: A messages is printed at the operator console.
2624 ;*
2625 ;* CALLING SEQUENCE: Include the label "ER9003" as the message pointer
2626 ;* parameter in the Diag Super error report macro call.
2627 ;*
2628 ;* COMMENTS: The message is printed as Basic and Extended error information.
2629 ;* The contents of the Indirect address register field of the DUT
2630 ;* CSR may be altered.
2631 ;*
2632 ;* SUBORDINATE ROUTINES USED: None.
2633 ;*****
2634
2635 025566 BGNMSG ER9003
2636 025566
2637 025566 006203
2638 025570 ASR R3 ;CALCULATE THE LINE NUMBER.
2639 025570 010346 PRINTB #EF9007,R3 ;REPORT THE ERROR TYPE AND LINE NUMBER.
2640 025572 012746 007360 MOV R3,-(SP)
2641 025576 012746 000002 MOV #EF9007,-(SP)
2642 025602 010600 MOV #2,-(SP)
2643 025604 104414 MOV SP,R0
2644 025606 062706 000006 TRAP C:PNTB
2645 025612 010201 MOV R2,R1 ;EXTRACT THE RECEIVED CHARACTER FROM THE
2646 025614 042701 177400 BIC #177400,R1 ; PASSED IN CHAR VALUE WITH FLAGS.
2647 025620 010146 PRINTX #EF9008,R1 ;REPORT THE VALUE OF THE RECEIVED CHAR.
2648 025622 012746 007454 MOV R1,-(SP)
2649 025626 012746 000002 MOV #EF9008,-(SP)
2650 025632 010600 MOV #2,-(SP)
2651 025634 104415 MOV SP,R0
2652 025636 062706 000006 TRAP C:PNTX
2653
2654 ;* Report OVERRUN flag set if necessary.
2655 ;*
2656 025642 012701 022704 MOV #EM9011,R1 ;SELECT THE OVERRUN ERROR MESSAGE.
2657 025646 032702 040000 BIT #BIT14,R2 ;CHECK OVERRUN ERROR FLAG IN PASSED IN CHAR.
2658 025652 001402 BEQ 2$ ;SKIP ERROR IF OVERRUN ERROR FLAG WAS CLEAR.
2659 025654 004737 025716 JSR PC,50$ ;REPORT THE OVERRUN ERROR FLAG WAS SET.
2660
2661 ;* Report FRAMING flag set if necessary.
2662 ;*
2663 2$: MOV #EM9012,R1 ;SELECT THE FRAMING ERROR MESSAGE.
2664 BIT #BIT13,R2 ;CHECK FRAMING ERROR FLAG IN PASSED IN CHAR.
2665 BEQ 4$ ;SKIP ERROR IF FRAMING ERROR FLAG WAS CLEAR.
2666 JSR PC,50$ ;REPORT THE FRAMING ERROR MESSAGE.
2667

```


GLOBAL ERROR REPORTING ROUTINE

- ER9003 -

```

2657      ; Report PARITY flag set if necessary.
2658      ;
2659 025676 012701 022724      4$:      MOV      #EM9013,R1      ;SELECT THE PARITY ERROR MESSAGE.
2660 025702 032702 010000      BIT      #BIT12,R2      ;CHECK PARITY ERROR FLAG IN PASSED IN CHAR.
2661 025706 001415      BEQ      60$      ;EXIT ROUTINE IF PARITY ERRO FLAG WAS CLEAR.
2662 025710 004737 025716      JSR      PC,50$      ;REPORT THE PARITY ERROR MESSAGE.
2663 025714 000412      BR      60$      ;EXIT THIS ROUTINE.
2664
2665      ;
2666      ; Local subroutine to report an error flag status.
2667      ;
2668 025716      50$:      PRINTX  #EF9009,R1
2669 025716 010146
2670 025720 012746 007513      MOV      R1,(SP)
2671 025724 012746 000002      MOV      #EF9009,-(SP)
2672 025730 010600      MOV      #2,(SP)
2673 025732 104415      MOV      SP,R0
2674 025734 062706 000006      TRAP     C$PNTX
2675 025740 000207      ADD      #6,SP
2676
2677      RTS      PC
2678
2679 025742 004737 032640      60$:      JSR      PC,PRTLPR      ;REPORT THE LPR CONTENTS FOR THIS LINE.
2680 025746      ENDMMSG
2681 025746 104423      L10015:      TRAP     C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER9004 -

```

2674 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9004
2675 ;*****
2676 ;* This is an error reporting subroutine which reports error summaries
2677 ;* for lines which have exceeded the specified maximum number of
2678 ;* individual reception errors.
2679 ;*
2680 ;* INPUTS: R1 Address of message to print first.
2681 ;* ERCNTB - Label at base of line error counters table.
2682 ;* ERSMRF - "Report error summary for line" flags.
2683 ;*
2684 ;* OUTPUTS: A message is printed at the operator console.
2685 ;*
2686 ;* CALLING SEQUENCE: Include the label "ER9004" as the message pointer
2687 ;* parameter in the Diag Super error report macro call.
2688 ;*
2689 ;* COMMENTS: The message is printed as Basic and Extended error information.
2690 ;* The contents of GPR's R2, R3, R4, and R5 are destroyed.
2691 ;*
2692 ;* SUBORDINATE ROUTINES USED: None.
2693 ;*****
2694
2695 025750 BGNMSG ER9004
2696 025750
2697 025750 ER9004::
2698 025750 PRINTB #EF0503,#EM9014 ;REPORT THE SECONDARY ERROR MESSAGE.
2699 025750 012746 022733 MOV #EM9014,(SP)
2700 025754 012746 005630 MOV #EF0503,-(SP)
2701 025760 012746 000002 MOV #2,-(SP)
2702 025764 010600 MOV SP,R0
2703 025766 104414 TRAP C#PNTB
2704 025770 062706 000006 ADD #6,SP
2705 025774 005002 CLR R2 ;CLEAR THE LINE COUNTER.
2706 025776 013703 002650 MOV ERSMRF,R3 ;GET THE ERROR SUMMARY FLAGS.
2707 026002 005004 CLR R4 ;CLEAR "LINE COUNTER TIMES 2" OFFSET.
2708 026004 000241 2$: CLC ;CLEAR THE CARRY FOR THE FOLLOWING ROTATE.
2709 026006 006003 ROR R3 ;SHIFT ANOTHER ERROR SUMMARY FLAG INTO CARRY.
2710 026010 103013 BCC 4$ ;SKIP PRINTING MESSAGE IF FLAG FOR LINE CLEAR.
2711 026012 PRINTX #EF9010,R2,ERCNTB(R4)
2712 026012 016446 003452 MOV ERCNTB(R4),-(SP)
2713 026016 010246 MOV R2,-(SP)
2714 026020 012746 007552 MOV #EF9010,-(SP)
2715 026024 012746 000003 MOV #3,-(SP)
2716 026030 010600 MOV SP,R0
2717 026032 104415 TRAP C#PNTX
2718 026034 062706 000010 ADD #10,SP
2719 026040 012405 4$: MOV (R4)+,R5 ;INCREMENT THE LINE OFFSET BY 2.
2720 026042 005202 INC R2 ;INCREMENT THE LINE COUNTER.
2721 026044 005703 TST R3 ;CHECK THE ERROR SUMMARY FLAGS.
2722 026046 001356 BNE 2$ ;IF MORE FLAGS SET, LOOP TO DO OTHER LINES.
2723 026050 ENDMSG
2724 026050
2725 026050 104423 L10016: TRAP C#MSG

```


GLOBAL ERROR REPORTING ROUTINE

- ER9005

```

2712 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9005
2713 ;*****
2714 ;* This is an error reporting subroutine which reports incomplete data
2715 ;* transmissions or receptions.
2716 ;*
2717 ;* INPUTS: R1 - Either "TRANSMITTED" or "RECEIVED" to indicate TX or RX.
2718 ;* R2 - Bit map of lines which did not complete TX or RX.
2719 ;* R4 - Address of base of the correct character counters table.
2720 ;* DPLEN8 - Label at base of data pattern length table.
2721 ;* EM9015 - Symbolic address of the "TRANSMITTED" message.
2722 ;*
2723 ;* OUTPUTS: A message is printed at the operator console.
2724 ;*
2725 ;* CALLING SEQUENCE: Include the label "ER9005" as the message pointer
2726 ;* parameter in the Diag Super error report macro call.
2727 ;*
2728 ;* COMMENTS: The message is printed as Basic and Extended error information.
2729 ;* The contents of the indirect address field in the DUT CSR may
2730 ;* be altered.
2731 ;*
2732 ;* SUBORDINATE ROUTINES USED: PRTLPR.
2733 ;*****
2734
2735 026052 BGNMSG ER9005
2736 026052 ER9005::
2737 026052 004537 005474 SAVE ;SAVE THE CONTENTS OF THE GPR'S.
2738 026056 PRINTB #EF9013,R1 ;CALL REGISTER SAVE SUBRT.
2739 026056 010146 ;REPORT THE SECONDARY ERROR MESSAGE.
2740 026060 012746 007765 MOV R1,-(SP)
2741 026064 012746 000002 MOV #EF9013,-(SP)
2742 026070 010600 MOV #2,-(SP)
2743 026072 104414 MOV SP,R0
2744 026074 062706 000006 TRAP C#PNTB
2745 026100 005003 CLR R3 ;CLEAR THE LINE COUNTER.
2746 026102 022701 023027 CMP #EM9015,R1 ;CHECK IF ADDRESS CORRESPONDS TO TX MESSAGE.
2747 026106 001032 BNE 6$ ;BRANCH IF RECEPTION MESSAGE TO BE PRINTED.
2748
2749 ;*
2750 ; Perform TX incomplete error message reporting.
2751 ;*
2752 PRINTX #EM9030 ;PRINT "NO TX COMPLETION INTERRUPTS RECEIVED"
2753
2754 026110 012746 023572 MOV #EM9030,-(SP)
2755 026114 012746 000001 MOV #1,-(SP)
2756 026120 010600 MOV SP,R0
2757 026122 104415 TRAP C#PNTX
2758 026124 062706 000004 ADD #4,SP
2759
2760 2$: CLC ;CLEAR THE CARRY FOR THE FOLLOWING ROTATE.
2761 ROR R2 ;SHIFT "TX NOT DONE" FLAG INTO CARRY.
2762 BCC 4$ ;SKIP PRINTING MESSAGE IF FLAG FOR LINE CLEAR.
2763 PRINTX #EF9020,R3 ;PRINT "TOO FEW TX ACTIONS GENERATED" MSG.
2764
2765 026136 010346 MOV R3,(SP)
2766 026140 012746 010334 MOV #EF9020,-(SP)
2767 026144 012746 000002 MOV #2,-(SP)
2768 026150 010600 MOV SP,R0
2769 026152 104415 TRAP C#PNTX

```

GLOBAL ERROR REPORTING ROUTINE

- ER9005 -

```

026154 062706 000006
2751 026160 004737 032640
2752 026164 005203
2753 026166 005702
2754 026170 001357
2755 026172 000440
2756
2757
2758
2759 026174 000241
2760 026176 006002
2761 026200 103031
2762 026202 006303
2763 026204 016305 005364
2764 026210 010246
2765 026212 010502
2766 026214 016505 003612
2767 026220 006202
2768 026222 006203
2769 026224
026224 010246
026226 010546
026230 011446
026232 010146
026234 010346
026236 012746 007651
026242 012746 000006
026246 010600
026250 104415
026252 062706 000016
2770 026256 012602
2771 026260 004737 032640
2772 026264 005724
2773 026266 005203
2774 026270 005702
2775 026272 001340
2776 026274
026274 004736
2777 026276
026276
026276 104423

```

```

4$: JSR PC,PRTLPR ;REPORT CONTENTS OF LPR REGISTER FOR THIS LINE.
    INC R3 ;INCREMENT LINE COUNTER.
    TST R2 ;CHECK THE "TX NOT DONE FLAGS".
    BNE 2$ ;IF MORE FLAGS SET, LOOP TO DO OTHER LINES.
    BR 10$ ;EXIT THIS ROUTINE.

; Perform RX incomplete error message reporting.
6$: CLC ;CLEAR THE CARRY FOR THE FOLLOWING ROTATE.
    ROR R2 ;SHIFT "RX NOT DONE" FLAG INTO CARRY.
    BCC 8$ ;SKIP PRINTING MESSAGE IF FLAG FOR LINE CLEAR.
    ASL R3 ;SHIFT LINE # TO GIVE CORRECT TABLE OFFSET.
    MOV TXRXLB(R3),R5 ;GET THE "ASSOCIATED" RECEIVE LINE OFFSET.
    MOV R2,(SP) ;SAVE THE "RX NOT DONE" FLAGS ON THE STACK.
    MOV R5,R2 ;COPY THE ASSOCIATED TX LINE OFFSET.
    MOV CHCNTB(R5),R5 ;GET THE TOTAL NUMBER OF EXPECTED CHARS.
    ASR R2 ;SHIFT THE TABLE OFFSET TO GIVE A LINE NUMBER.
    ASR R3 ;SHIFT TABLE OFFSET TO GIVE LINE NUMBER.
    PRINTX #EF9012,R3,R1,(R4),R5,R2 ;REPORT NUMBER OF CHARS ON LINE.
    MOV R2,-(SP)
    MOV R5,(SP)
    MOV (R4),-(SP)
    MOV R1,(SP)
    MOV R3,-(SP)
    MOV #EF9012,-(SP)
    MOV #6,-(SP)
    MOV SP,R0
    TRAP C$PNTX
    ADD #16,SP

    MOV (SP)+,R2 ;RESTORE THE "RX NOT DONE" FLAGS.
    JSR PC,PRTLPR ;REPORT CONTENTS OF LPR REGISTER FOR THIS LINE.
8$: TST (R4)+ ;INCREMENT THE CHARACTER COUNTER TABLE.
    INC R3 ;INCREMENT THE LINE COUNTER.
    TST R2 ;CHECK THE "RX NOT DONE FLAGS".
    BNE 6$ ;IF MORE FLAGS SET, LOOP TO DO OTHER LINES.
10$: PASS ;RESTORE THE CONTENTS OF THE GPRS.
    JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

ENDMSG

```

L10017: TRAP C\$MSG

GLOBAL ERROR REPORTING ROUTINE

- ER9007 -

SEQ 0072

```

2779 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9007 -
2780 ;*****
2781 ;* This is an error reporting subroutine which is used to report that
2782 ;* something other than a selftest code was found in a selftest code
2783 ;* FIFO slot during the removal of the selftest codes from the FIFO.
2784 ;* This routine is used by the RSTRPT routine.
2785 ;*
2786 ;* INPUTS: R1 - Address of error message qualifier string.
2787 ;* R2 - Incorrect code as read from the selftest code FIFO slot.
2788 ;* R3 - Line number associated with the selftest FIFO slot.
2789 ;*
2790 ;* OUTPUTS: A message is printed at the operator console.
2791 ;*
2792 ;* CALLING SEQUENCE: Include the label "ER9007" as the message pointer
2793 ;* parameter in the Diag Super error report macro call.
2794 ;*
2795 ;* COMMENTS: The message is printed as Basic and Extended error information.
2796 ;*
2797 ;* SUBORDINATE ROUTINES USED: None.
2798 ;*****
2799
2800 026300 BGNMSG ER9007
2801 026300
2802 026300 042703 177760
2803 026304 BIC #177760,R3 ;REMOVE ALL BUT LINE # BITS FROM LINE # WORD.
2803 026304 PRINTB #EF9018,R1,R3 ;REPORT SECONDARY ERROR MESSAGE.
2803 026304 010346
2803 026306 010146
2803 026310 012746 010203
2803 026314 012746 000003
2803 026320 010600
2803 026322 104414
2803 026324 062706 000010
2804 026330 PRINTX #EF9017,R1,R2 ;REPORT THE ACTUAL INCORRECT CODE.
2804 026330 010246
2804 026332 010146
2804 026334 012746 010127
2804 026340 012746 000003
2804 026344 010600
2804 026346 104415
2804 026350 062706 000010
2805
2806 026354 ENDMSG
2806 026354
2806 026354 104423

```

```

ER9007::
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

```

```

L10020:
TRAP C$MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER9008 -

```

2808 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9008 -
2809 ;*****
2810 ; This is an error reporting subroutine which is used to report that
2811 ; an unexpected code or character has been found in the DUT receive
2812 ; character FIFO.
2813 ;
2814 ; INPUTS: R1 - Address of partial error message string.
2815 ; R2 - Incorrect code as read from the selftest code FIFO slot.
2816 ;
2817 ; OUTPUTS: A message is printed at the operator console.
2818 ;
2819 ; CALLING SEQUENCE: Include the label "ER9008" as the message pointer
2820 ; parameter in the Diag Super error report macro call.
2821 ;
2822 ; COMMENTS: The message is printed as Basic and Extended error information.
2823 ;
2824 ; SUBORDINATE ROUTINES USED: None.
2825 ;*****
2826
2827 026356 BGNMSG ER9008
2828 026356
2829
2830 ;*
2831 ; Extract the line number from the incorrect code or character which was read
2832 ; from the selftest code FIFO slot.
2833 ;
2833 026356 010203
2834 026360 000303
2835 026362 042703 177760
2836 026366
2836 026366 010346
2836 026370 010146
2836 026372 012746 010032
2836 026376 012746 000003
2836 026402 010600
2836 026404 104414
2836 026406 062706 000010
2837 026412
2837 026412 010246
2837 026414 010146
2837 026416 012746 010127
2837 026422 012746 000003
2837 026426 010600
2837 026430 104415
2837 026432 062706 000010
2838
2839 026436
2839 026436
2839 026436 104423

```

```

                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

```

```

L10021:
                TRAP C#MSG

```

GLOBAL ERROR REPORTING ROUTINE

- ER9101 -

```

2841 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9101 -
2842 ;*****
2843 ;* This is a general error reporting subroutine which reports a message
2844 ;* which takes a single, 2 digit decimal argument after the end of an
2845 ;* ASCII message.
2846 ;*
2847 ;* INPUTS: R1 - Value to be printed after msg as 2 decimal digits.
2848 ;* R2 - Address of message to print first.
2849 ;*
2850 ;* OUTPUTS: A messages is printed at the operator console.
2851 ;*
2852 ;* CALLING SEQUENCE: Include the label "ER9101" as the message pointer
2853 ;* parameter in the Diag Super error report macro call.
2854 ;*
2855 ;* COMMENTS: The message is printed as Basic error information.
2856 ;*
2857 ;* SUBORDINATE ROUTINES USED: None.
2858 ;*****
2859
2860 026440 BGNMSG ER9101
2861 026440 ER9101::
2862 026440 PRINTB #EF9006,R2,R1 ;REPORT THE STRING FOLLOWED BY THE NUMBER.
2863 026440 010146 MOV R1,-(SP)
2864 026442 010246 MOV R2,-(SP)
2865 026444 012746 007334 MOV #EF9006,-(SP)
2866 026450 012746 000003 MOV #3,-(SP)
2867 026454 010600 MOV SP,R0
2868 026456 104414 TRAP C$PNTB
2869 026460 062706 000010 ADD #10,SP
2870
2871 ENDMSG
2872
2873 L10022:
2874 TRAP C$MSG
2875
2876 026464 104423

```

GLOBAL ERROR REPORTING ROUTINE

- ER9102 -

```

2866 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9102 -
2867 ;*****
2868 ;* This is an error reporting subroutine which prints additional error
2869 ;* information after the error message header.
2870 ;* This routine is passed a bit map which specifies the lines for which
2871 ;* the error condition should be reported.
2872 ;*
2873 ;* INPUTS: R1 - Address of the message to be printed by this routine.
2874 ;* R2 Bit map of lines for which to report errors.
2875 ;*
2876 ;* OUTPUTS: Messages are printed at the operator console.
2877 ;*
2878 ;* CALLING SEQUENCE: Load the address of the message in R1.
2879 ;* Load the bit map of lines with errors in R2.
2880 ;* Include the label "ER9102" as the message pointer
2881 ;* (ERRBLK) in the Diag Super error report macro call.
2882 ;*
2883 ;* COMMENTS: The output format of this message is:
2884 ;* "TEXT MESSAGE POINTED TO BY R1"
2885 ;* "ERROR CONDITION ON LINE nn"
2886 ;* "ERROR CONDITION ON LINE ..."
2887 ;* The top message, and the message for each line are printed
2888 ;* as basic error information.
2889 ;*
2890 ;* SUBORDINATE ROUTINES USED: None.
2891 ;*****
2892
2893 026466 BGNMSG ER9102
2894 026466
2895 026466 004537 005474
2896 026472
2897 026472 010146
2898 026474 012746 005630
2899 026500 012746 000002
2900 026504 010600
2901 026506 104414
2902 026510 062706 000006
2903 026514 005003
2904 026516 000241
2905 026520 006002
2906 026522 103011
2907 026524
2908 026524 010346
2909 026526 012746 010420
2910 026532 012746 000002
2911 026536 010600
2912 026540 104414
2913 026542 062706 000006
2914 026546 005203
2915 026550 005702
2916 026552 001361
2917 026554
2918 026554 012746 010415
2919 026560 012746 000001
2920 026564 010600

```

```

ER9102::
;SAVE THE CONTENTS OF THE GPRS.
R5,PREG05 ;CALL REGISTER SAVE SUBRT.

;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

2$: CLR R3 ;CLEAR THE LINE NUMBER.
CLC ;PREPARE TO ROTATE NEXT BIT OUT OF MAP.
ROR R2 ;GET THE NEXT BIT OF THE BIT MAP.
BCC 4$ ;SKIP PRINTING MESSAGE IF THE BIT IS CLEAR.
PRINTB #EF9103,R3 ;REPORT THIS LINE HAD THE ERROR.
MOV R3,-(SP)
MOV #EF9103,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C:PNTB
ADD #6,SP

4$: INC R3 ;INCREMENT THE LINE COUNTER.
TST R2 ;CHECK THE BIT MAP.
BNE 2$ ;LOOP IF NOT ALL SET BITS REMOVED FROM BIT MAP.
PRINTB #EF9101 ;PRINT A BLANK LINE.
MOV #EF9101,-(SP)
MOV #1,-(SP)
MOV SP,R0

```


GLOBAL ERROR REPORTING ROUTINE

- ER9102

SEQ 0076

026566 104414
026570 062706 000004
2906 026574
026574 004736
2907 026576
026576
026576 104423

60\$: PASS
ENDMSG

JSR

,RESTORE THE SAVED CONTENTS OF THE GPRS.
PC,0(SP)+

TRAP C:PNTB
ADD 04,SP

;RETURN TO PREG05 SUBRT.

L10023:

TRAP C:MSG

GLOBAL ERROR REPORTING ROUTINE

- ER9301 -

SEQ 0077

```

2909 .SBTTL GLOBAL ERROR REPORTING ROUTINE - ER9301 -
2910 ;*****
2911 ;* This is an error reporting subroutine which prints any BMP codes
2912 ;* that are found in the BMP code queue, together with the the number of
2913 ;* the test that was executing at the time the BMP code was logged.
2914 ;*
2915 ;* INPUTS: R1 - The address of the first message to be reported.
2916 ;* R2 - The address of the next empty cell in the queue.
2917 ;*
2918 ;* OUTPUTS: The test number followed by the BMP code are printed at the
2919 ;* operator console.
2920 ;*
2921 ;* CALLING SEQUENCE: Include the label "ER9301" as the message pointer
2922 ;* parameter in the Diag Super error report macro call.
2923 ;*
2924 ;* COMMENTS: The message is printed as Basic error information.
2925 ;*
2926 ;* SUBORDINATE ROUTINES USED: None.
2927 ;*****
2928
2929 026600 BGNMSG ER9301
2930 026600 ER9301::
026600 004537 005474 SAVE ;SAVE THE GPRS ON THE STACK.
2931 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
2932 026604 PRINTB #EF0503,R1 ;REPORT UNEXPECTED BMP CODES FOUND.
026604 010146 MOV R1,-(SP)
026606 012746 005630 MOV #EF0503,-(SP)
026612 012746 000002 MOV #2,-(SP)
026616 010600 MOV SP,R0
026620 104414 TRAP C#PNTB
026622 062706 000006 ADD #6,SP
2933 026626 012703 002662 MOV #BMPCQB,R3 ;GET THE START ADDRESS OF THE BMP CODE QUEUE.
2934 026632 012705 024066 MOV #EM9302,R5 ;GET THE MESSAGE TO BE REPORTED.
2935 026636 012301 2#: MOV (R3)+,R1 ;GET THE NUMBER OF THE TEST THAT WAS EXECUTING.
2936 026640 012304 MOV (R3)+,R4 ;GET BMP CODE THAT WAS REPORTED OFF THE QUEUE.
2937 026642 004737 026724 JSR PC,50# ;GO REPORT THE BMP CODE.
2938 026646 020302 CMP R3,R2 ;CHECK IF ALL CODES HAVE BEEN REPORTED.
2939 026650 103772 BLO 2# ;IF IT IS NOT THE LAST BMP CODE THEN LOOP.
2940 ;*
2941 ; Check if overflow has occurred.
2942 ; The conditions for overflow are: the pointer contains the address of the
2943 ; last cell in the queue, and a bmp code has already been written into that
2944 ; cell.
2945 ;-
2946 026652 020227 003056 CMP R2,#BMPCQE-4 ;CHECK IF THE POINTER IS AT THE LAST LOCATION.
2947 026656 001036 BNE 60# ;EXIT IF NOT AT THE LAST LOCATION.
2948 026660 005762 000002 TST 2(R2) ;CHECK FOR A BMP CODE IN THE LAST CELL
2949 026664 001433 BEQ 60# ;EXIT IF NO OVERFLOW HAS OCCURED, CELL EMPTY.
2950 026666 012301 MOV (R3)+,R1 ;GET THE TEST NUMBER OFF THE QUEUE.
2951 026670 011304 MOV (R3),R4 ;GET THE BMP CODE OFF THE QUEUE.
2952 026672 012705 024116 MOV #EM9303,R5 ;SELECT THE MESSAGE TO BE REPORTED.
2953 026676 PRINTX #EF9302 ;REPORT OVERFLOW CONDITION.
026676 012746 010544 MOV #EF9302,-(SP)
026702 012746 000001 MOV #1,-(SP)
026706 010600 MOV SP,R0
026710 104415 TRAP C#PNTX

```


GLOBAL ERROR REPORTING ROUTINE

- ER9301 -

SEQ 0078

```

026712 062706 000004
2954 026716 004737 026724      JSR    PC,50$      ;REPORT THE LAST BMP CODE PLACED ON THE QUEUE.
2955 026722 000414              BR      60$      ;EXIT.
2956
2957 026724      50$: PRINTX  #EF9301,R5,R1,R4 ;PRINT THE MESSAGE.
026724 010446
026726 010146
026730 010546
026732 012746 010466
026736 012746 000004
026742 010600
026744 104415
026746 062706 000012
2958 026752 000207
2959 026754      60$: RTS    PC      ;RETURN.
026754 004736      PASS           ;RESTORE THE GPR CONTENTS.
2960
2961 026756      JSR    PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
026756
026756 104423
L10024: TRAP    C#MSG

```

GLOBAL SUBROUTINES SECTION

2963
2965
2966
2967
2968
2969
2971
2972
2973
2974
2975
2976

```
.C3TTL GLOBAL SUBROUTINES SECTION
;*****
;
;          FVTSKL3.P11
;*****
;
;
; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
; THAT ARE USED IN MORE THAN ONE TEST.
; -
```

GLOBAL SUBROUTINE

- ALTFLD

SEQ 0080

```

2978 .SBTTL GLOBAL SUBROUTINE ALTFLD -
2979 ;* *****
2980 ;* Alter Device Register Fields Routine -
2981 ;* This subroutine alters the specified field of the specified device
2982 ;* register for the specified lines. This routine can be used to set
2983 ;* or clear bits within selected fields of selected registers.
2984 ;* Use examples: Set RX.BAUD.RATE fields on lines 3 and 6.
2985 ;* Clear TX.DMA bits on all lines.
2986 ;*
2987 ;* INPUTS: R1 Address of the registers to alter.
2988 ;* R2 - Bit fields set to desired states.
2989 ;* R3 - Bit map of lines for which to alter register.
2990 ;* R4 Mask of bits to alter (1 indicates change bit).
2991 ;* CSRA - Contains the address of the device CSR.
2992 ;* IESTAT Saved states of the interrupt enable bits.
2993 ;*
2994 ;* OUTPUTS: DEVICE REGISTERS - Specified register fields altered.
2995 ;* CSR IND.ADR.REG field - Destroyed.
2996 ;*
2997 ;* CALLING SEQUENCE: JSR PC,ALTFLD
2998 ;*
2999 ;* COMMENTS: This routine reads the specified registers for all lines
3000 ;* with numbers lower than the highest specified line.
3001 ;* This routine does not read the CSR.
3002 ;*
3003 ;* SUBROUTINES CALLED: None.
3004 ;*
3005 ;* *****
3006 026760 004537 005474 ALTFLD:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3007 026760 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3008
3009 ; Set up to loop for each line:
3010 ; Prepare the word to be ORed into the register contents.
3011 ; Set up the word to write into the IND.ADR.REG field of the CSR.
3012 ;
3013 026764 010400 MOV R4,R0 ;CALCULATE THE NEW CONTENTS OF THE
3014 026766 005100 COM R0 ; REGISTER FIELDS WHICH ARE TO BE
3015 026770 040002 BIC R0,R2 ; ALTERED BY THIS ROUTINE.
3016 026772 013705 002374 MOV IESTAT,R5 ;SET UP TO WRITE IND.ADR.REG FIELD TO 0.
3017 ;
3018 ; Loop once for each line, altering the specified field in the specified
3019 ; register if the line has been selected for altering.
3020 ; Exit the loop if no more lines to alter, or if we have altered the max
3021 ; allowable number of lines (as specified by NUMLNS).
3022 ;
3023 026776 000241 CLC ;PREPARE FOR ROTATE, "TST R5" DOES THIS BELOW.
3024 027000 006003 2$: ROR R3 ;GET THE LINE SELECT BIT FOR THIS LINE.
3025 027002 103006 BCC 4$ ;SKIP SETUP IF LINE IS NOT SELECTED.
3026 027004 010577 153270 MOV R5,@CSRA ;SET OUT CSR IND.ADR.REG FIELD TO THIS LINE.
3027 027010 011100 MOV (R1),R0 ;GET THE PRESENT CONTENTS OF THE REG TO ALTER.
3028 027012 040400 BIC R4,R0 ;CLEAR THE BIT FIELDS WE ARE TO ALTER.
3029 027014 050200 BIS R2,R0 ;OR IN THE NEW STATES OF THE FIELDS.
3030 027016 010011 MOV R0,(R1) ;WRITE THE NEW REGISTER CONTENTS TO THE REG.
3031 027020 005205 4$: INC R5 ;SET LINE NUMBER TO THE NEXT LINE.
3032 027022 005703 TST R3 ;CHECK FOR UNHANDLED LINES, CLEAR CARRY FLAG.
3033 027024 001365 BNE 2$ ;LOOP IF SELECTED LINE(S) IS NOT HANDLED.

```

GLOBAL SUBROUTINE

ALTFLD

3034

3035 027026

603: PASS

;RESTORE GPRS.

027026 004736

JSR

PC,@(SP).

;RETURN TO PREG05 SUBRT.

3036 027030 000207

RTS PC

;RETURN TO CALLING ROUTNE.

GLOBAL SUBROUTINE

CALMSL -

```

3038 .SBTTL GLOBAL SUBROUTINE CALMSL -
3039 ;* *****
3040 ;* - Calibrate Milli Second Loop count subroutine -
3041 ;* This subroutine calibrates the timing loop which is used in the MSLOOP
3042 ;* routine. This subroutine calculates a value for the MSLCNT variable
3043 ;* which is the number of software loops which takes 1 ms to execute in
3044 ;* the MSLOOP routine. This routine calibrates the count by using the
3045 ;* Line Time Clock (LTC), so if no LTC is available the default value for
3046 ;* the delay count must be used.
3047 ;*
3048 ;*
3049 ;* INPUTS: MSLCNT - Default 1 ms delay loop count value, or
3050 ;* value from previous calibration.
3051 ;* MSTICK - Number of MS per LTC clock tick.
3052 ;* TIMER1 - Timer counter changed by LTC interrupt service rtn.
3053 ;* CLKHRZ - Number of LTC clicks per second (50 or 60).
3054 ;*
3055 ;* OUTPUTS: CARRY Set if LTC is available, and new calibration performed.
3056 ;* MSLCNT - New 1 ms delay loop count value if LTC available, or
3057 ;* unchanged if no LTC is available.
3058 ;*
3059 ;* CALLING SEQUENCE: JSR PC,CALMSL
3060 ;*
3061 ;* COMMENTS:
3062 ;*
3063 ;* SUBORDINATE ROUTINES CALLED: UNSDIV,OOPS.
3064 ;* - *****
3065
3066 027032 CALMSL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3067 027032 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3068 027036 005037 027252 CLR 62$ ;CLEAR THE 2ND TIME FLAG.
3069 ;*
3070 ; Synchronize with the LTC.
3071 027042 012705 000001 2$: MOV #1,R5 ;SET OUTER LOOP COUNTER TO 1 LOOP.
3072 ;INCREASE THE VALUE LOADED INTO THIS COUNTER IF THE <***
3073 ;FOLLOWING LOOP FAILS ON FUTURE, FASTER PROCESSORS. <***
3074 027046 005000 CLR R0 ;CLEAR THE WAIT FOR CLOCK INT COUNTER.
3075 027050 012737 000001 002450 MOV #1,TIMER1 ;SET UP COUNT OF 1 TO SYNCH WITH LTC.
3076 027056 005737 002450 4$: TST TIMER1 ;CHECK FOR COUNTER HAVING GONE TO ZERO.
3077 027062 001410 BEQ 6$ ;JUMP OUT OF LOOP IF LTC HAS INTERRUPTED.
3078 027064 005200 INC R0 ;COUNT THIS ITERATION OF THE INNER LOOP.
3079 027066 001373 BNE 4$ ;LOOP IF COUNTER HAS NOT TURNED OVER.
3080 027070 005305 DEC R5 ;DECREMENT THE INNER LOOP COUNTER.
3081 027072 003371 BGT 4$ ;LOOP IF OUTER LOOP COUNT NOT UP.
3082 ;*
3083 ; If we got no LTC interrupt, indicate that there is no LTC available.
3084 ; LTC must be flakey, or not really an LTC at all.
3085 ;*
3086 027074 005037 002446 CLR CLKHRZ ;CLEAR LTC FREQUENCY WORD TO INDICATE NO LTC.
3087 027100 000241 CLC ;INDICATE FAILURE FOR RETURN.
3088 027102 000461 BR 60$ ;BYPASS THE FOLLOWING CALIBRATION PROCEDURES.
3089 ;*
3090 ; We are now synchronized with the LTC.
3091 ; Set up for the calibration loop.
3092 ;*
3093 027104 012704 002450 6$: MOV #TIMER1,R4 ;WILL TEST TIMER1 IN THE LOOP BELOW.

```

GLOBAL SUBROUTINE

CALMSL

```

3094 027110 005001          CLR      R1          ;CLEAR THE OUTER LOOP COUNTER.
3095 027112 005002          CLR      R2          ;INDICATE TO CHECK ALL BITS OF TIMER1.
3096 027114 005003          CLR      R3          ;INDICATE TO CHECK FOR TIMER1 CLEAR.
3097 027116 012714 000001    MOV      #1,(R4)    ;LOAD TIMER1 WITH COUNT OF 1.
3098
3099 027122 013705 002462    8$:  MOV      MSLCNT,R5    ;LOAD MS LOOP COUNT.
3100 027126 011400          10$: MOV      (R4),R0    ;GET THE TIMER1 VALUE.
3101 027130 010037 027254    MOV      R0,64$    ;SAVE WORD (LIKE IN THE REAL LOOP).
3102 027134 040200          BIC      R2,R0        ;LEAVE ALL THE BITS.
3103 027136 020003          CMP      R0,R3        ;COMPARE AGAINST ZERO.
3104 027140 000261          SEC                  ;SET CARRY IN CASE OF SUCCESS.
3105 027142 001406          BEQ      12$          ;EXIT LOOP IF TIMER1 HAS CLEARED.
3106 027144 005305          DEC      R5          ;COUNT DOWN THE INSIDE MS LOOP COUNT.
3107 027146 001367          BNE      10$          ;LOOP IF MS NOT UP.
3108 027150 005301          DEC      R1          ;DECREMENT THE MS TIME COUNT.
3109 027152 001363          BNE      8$          ;KEEP LOOPING.
3110 027154 004737 032412    JSR      PC,00PS    ;WE OVERFLOWED, SOMETHING IS WRONG, ABORT.
3111
3112      ;+
3113      ; We have now have loop count information for one clock tick.
3114      ; We have negative of number of outer loops in R1, each is MSLCNT inner loops.
3115      ; We have the portion of the last outer loop not executed, in R5.
3116      ; Now we calculate the total number of inner loops executed.
3117
3117 027160 005401          12$: NEG      R1          ;GET NUMBER OF OUTER LOOPS.
3118 027162 013702 002462    MOV      MSLCNT,R2    ;GET THE NUMBER OF INNER LOOPS PER OUTER LOOP.
3119 027166 010203          MOV      R2,R3        ;COPY NUMBER OF LOOPS FOR MULTIPLY.
3120 027170 160502          SUB      R5,R2        ;CALC # OF INNER LOOPS DONE IN LAST OUTER LOOP
3121 027172 010204          MOV      R2,R4        ; AND ADD TO ACCUMULATOR LSWORD.
3122 027174 005005          CLR      R5          ;CLEAR ACCUMULATOR MSWORD.
3123 027176 005301          14$: DEC      R1          ;CHECK R1 FOR 0 CONDITION
3124 027200 100403          BMI      16$          ; SKIP MULTIPLICATION IF ZERO
3125 027202 060304          ADD      R3,R4        ;MULTIPLY NUMBER OF INNER
3126 027204 005505          ADC      R5          ; LOOPS PER OUTER LOOP BY
3127 027206 000773          BR       14$          ;NUMBER OF OUTER LOOPS PERFORMED.
3128
3129      ;+
3130      ; Divide the total number of inner loops by the number of MS per LTC tick.
3131
3131 027210 013701 002460    16$: MOV      MSTICK,R1    ;# OF MS PER LTC TICK IS DIVISOR.
3132 027214 010403          MOV      R4,R3        ;LSWORD OF LOOP COUNT IS LSWORD OF DIVIDEND.
3133 027216 010502          MOV      R5,R2        ;MSWORD OF LOOP COUNT IS MSWORD OF DIVIDEND.
3134 027220 004737 037234    JSR      PC,UNSDIV    ;DIVIDE NUMBER OF LOOPS BY MS PER LTC TICK.
3135 027224 103402          BCS      18$          ;BYPASS OOPS IF WE'RE OK.
3136 027226 004737 032412    JSR      PC,00PS    ;CLOCK ROUTINES ARE NOT LONG ENOUGH, OR BUG.
3137 027232 010137 002462    18$: MOV      R1,MSLCNT ;SET NEW VALUE FOR MS LOOP COUNT.
3138 027236 005137 027252    COM      62$        ;SET THE 2ND ITERATION FLAGS IF 1ST ITERATION.
3139 027242 001277          BNE      2$          ;BRANCH IF ONLY ONE ITERATION DONE.
3140 027244 000261          SEC                  ;SET THE SUCCESS FLAG FOR EXIT.
3141
3142 027246          60$:  PASS          ;RESTORE GPRS,
3143      027246 004736          RTS      PC          PC,0(SP); ;RETURN TO PREG05 SUBRT.
3144      027250 000207          ; CARRY SUCCESS FLAG. SET IF SUCCESS.
3145
3145 027252 000000          62$: .WORD      0          ;2ND CALIBRATION ITERATION FLAGS.
3146 027254 000000          64$: .WORD      0          ;DUMMY WORD FOR STORAGE OF THE READ WORD.

```


GLOBAL SUBROUTINE

- CHKBMP -

```

3148 .SBTTL GLOBAL SUBROUTINE - CHKBMP -
3149 ;* *****
3150 ;* - CHECK IF CHARACTER IS A BMP CODE -
3151 ;* This subroutine is used to check for BMP codes.
3152 ;* If a BMP code is detected, it will be saved on the queue to be reported
3153 ;* later. The carry is used as a flag to indicate a code has been found.
3154 ;*
3155 ;* INPUTS: R2 - Contains the data to be checked.
3156 ;*
3157 ;* OUTPUTS: R1 - Contains the message to be reported.
3158 ;* ERRBLK - Contains the error reporting routine.
3159 ;* Carry bit is used to indicate a BMP code found, Carry set.
3160 ;*
3161 ;* CALLING SEQUENCE: JSR PC,CHKBMP
3162 ;*
3163 ;* COMMENTS:
3164 ;*
3165 ;* SUBORDINATE ROUTINES CALLED: SAVBMP.
3166 ;* *****
3167
3168 027256 CHKBMP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3169 027256 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3170 027262 012700 170301 MOV #170301,R0 ;SET UP THE FLAGS OF A BMP CODE.
3171 027270 001011 BIC R2,R0 ;TRY TO CLEAR THE BMP CODE FLAGS.
3172 027272 004737 035624 BNE 2$ ;IF NOT A BMP CODE, EXIT WITH FAILURE.
3173 027276 012701 021507 JSR PC,SAVBMP ;SAVE THE BMP CODE ON THE QUEUE.
3174 027302 012737 025034 005472 MOV #EM5303,R1 ;PASS THE MESSAGE TO BE REPORTED.
3175 027310 000261 MOV #ER1603,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
3176 027312 000401 SEC ;PASS FLAG TO INDICATE SUCCESS, BMP CODE FOUND.
3177 027314 000241 BR 60$ ;EXIT.
3178 027316 2$: CLC ;PASS FLAG TO INDICATE FAILURE.
3179 027316 60$: PASS R1 ;RESTORE GPRS, EXCEPT
3180 027316 010166 000004 MOV R1,R1$LOT(SP) ;PUT R1 IN STACK SLOT.
3181 027322 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3181 027324 000207 RTS PC ;R1 - CONTAINS THE ADDRESS OF ERROR MESSAGE.
;CARRY BIT - SET INDICATES SUCCESS.

```

GLOBAL SUBROUTINE

- CHKEXT -

SEQ 0085

```

3183 .SBTTL GLOBAL SUBROUTINE - CHKEXT -
3184 ;* *****
3185 ;* - Check For Extra Character Routine -
3186 ;* This subroutine checks for the condition which indicates that an extra
3187 ;* character has been received during the reception of a data pattern.
3188 ;* If this routine determines that it is likely that an extra character
3189 ;* has been received it indicates this in the status information returned
3190 ;* to the calling routine.
3191 ;*
3192 ;* INPUTS: R3 - RX line number multiplied by 2 (offset into word tables).
3193 ;* R4 - Base address of resync que containing RX chars.
3194 ;* R5 - Mask of "inactive" (non-data) bits of RX and TX chars.
3195 ;* CHCNTB - Base of number of chars to TX on each line table.
3196 ;* RXCNTB - Base of the RX character counters table.
3197 ;* RXPTRB - Base of the RX character pointers table.
3198 ;* TXRXLB - Base of TX/RX line number association table.
3199 ;*
3200 ;* OUTPUTS: CARRY - Set if extra character condition is verified.
3201 ;*
3202 ;* CALLING SEQUENCE: JSR PC,CHKEXT
3203 ;*
3204 ;* COMMENTS: The following symbols are used in line comments:
3205 ;* CHR0 - Character at bottom of resync que (first received).
3206 ;* CHR1, CHR2 - 2 characters received after CHR0.
3207 ;* EXP0 - Character expected to be received next.
3208 ;* EXP1, EXP2 - Character expected to be received after EXP0, etc.
3209 ;*
3210 ;* SUBORDINATE ROUTINES CALLED: None.
3211 ;*
3212 ;* *****
3212 027326 CHKEXT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3213 027326 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3214 027332 016302 003552 MOV RXPTRB(R3),R2 ;GET THE RX DATA POINTER.
3215 027340 012400 TST (R4)+ ;INCREMENT R4 BY 2 TO POINT TO CHR1.
3216 027342 100026 MOV (R4)+,R0 ;GET CHR1 FROM THE QUE, DATA.VALID INTO N FLAG.
3217 027344 040500 BPL 52$ ;EXIT WITH "FAILURE" IF CHR1 NOT VALID.
3218 027346 112201 BIC R5,R0 ;REMOVE INACTIVE BITS FROM CHR1 VALUE.
3219 027350 040501 MOVB (R2)+,R1 ;GET EXP0 FROM THE DATA PATTERN.
3220 027352 120100 BIC R5,R1 ;REMOVE INACTIVE BITS FROM EXP0 VALUE.
3221 027354 001021 CMPB R1,R0 ;COMPARE CHR1 AND EXP0.
3222 027356 016300 003712 BNE 52$ ;EXIT WITH "FAILURE" IF CHR1 <> EXP0.
3223 027362 005200 MOV RXCNTB(R3),R0 ;COMPARE THE PRESENT RX CHARACTER COUNT PLUS 1
3224 027364 016301 005364 INC R0 ; WITH THE EXPECTED NUMBER OF CHARS TO RX ON
3225 027370 020061 003612 MOV TXRXLB(R3),R1 ; LINE (NUMBER TRANSMITTED AND LOOPED BACK) TO
3226 027374 001407 CMP R0,CHCNTB(R1) ; DETERMINE IF CHR1 IS LAST EXPECTED CHAR.
3227 027376 011400 BEQ 50$ ;EXIT WITH "SUCCESS" IF CHR1 IS LAST CHAR.
3228 027400 100005 MOV (R4),R0 ;GET CHR2 FROM THE QUE, DATA.VALID INTO N FLAG.
3229 027402 040500 BPL 50$ ;EXIT WITH "SUCCESS" IF CHR1 WAS LAST IN QUE.
3230 027404 111201 BIC R5,R0 ;REMOVE INACTIVE BITS FROM CHR2 VALUE.
3231 027406 040501 MOVB (R2),R1 ;GET THE EXP1 VALUE.
3232 027410 020001 BIC R5,R1 ;REMOVE INACTIVE BITS FROM EXP1 VALUE.
3233 027412 001002 CMP R0,R1 ;COMPARE CHR2 AND EXP1.
3234 BNE 52$ ;EXIT WITH "FAILURE" IF CHR2 <> EXP1.
3235 ;*
3236 ;* It is likely that we received an extra character within the data pattern.
3237 ;* Indicate "success" and exit.
3238 ;*

```


GLOBAL SUBROUTINE

- CHKEXT -

```
3239 027414 000261      50$:   SEC           ;SET THE SUCCESS FLAG.
3240 027416 000401      BR      60$       ;EXIT THE ROUTINE.
3241
3242
3243      ;*
3244      ; We didn't receive a single extra character at this point in the data pattern.
3245      ; Indicate "failure" and exit.
3246 027420 000241      52$:   CLC           ;CLEAR THE SUCCESS FLAG.
3247
3248 027422      60$:   PASS           ;RESTORE GPRS.
      027422 004736      JSR      PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
3249 027424 000207      RTS      PC       ;CARRY - SET IF SUCCESS (EXTRA CHAR RXED).
```

GLOBAL SUBROUTINE

- CHKLOS

SEQ 0087

```

3251 .SBTTL GLOBAL SUBROUTINE - CHKLOS -
3252 ;** *****
3253 ;* - Check For Lost Character Routine -
3254 ;* This subroutine checks for the condition which indicates that a char
3255 ;* has been "lost" from the looped back data pattern during a transmission
3256 ;* and reception test. If this routine determines that it is likely that
3257 ;* a character has been lost, it indicates this in the status information
3258 ;* returned to the calling routine.
3259 ;*
3260 ;* INPUTS: R3 - RX line number multiplied by 2 (offset into word tables).
3261 ;* R4 - Base address of resync que containing RX chars.
3262 ;* R5 - Mask of "inactive" (non data) bits of RX and TX chars with
3263 ;* all set bits in a single, left justified group.
3264 ;* CHCNTB - Base of number of chars to TX on each line table.
3265 ;* RXCNTB - Base of the RX character counters table.
3266 ;* RXPTRB - Base of the RX character pointers table.
3267 ;* TXRXLB - Base of TX/RX line number association table.
3268 ;*
3269 ;* OUTPUTS: CARRY - Set if lost character condition is verified.
3270 ;*
3271 ;* CALLING SEQUENCE: JSR PC,CHKLOS
3272 ;*
3273 ;* COMMENTS: The following symbols are used in line comments:
3274 ;* CHR0 - Character at bottom of resync que (first received).
3275 ;* CHR1, CHR2 - 2 characters received after CHR0.
3276 ;* EXP0 - Character expected to be received next.
3277 ;* EXP1, EXP2 - Character expected to be received after EXP0, etc.
3278 ;*
3279 ;* SUBORDINATE ROUTINES CALLED: None.
3280 ;-- *****
3281 027426 004537 005474 CHKLOS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3282 027426 016301 003712 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3283 027436 005201 MOV RXCNTB(R3),R1 ;COMPARE THE PRESENT RX CHARACTER COUNT PLUS 1
3284 027440 016300 005364 INC R1 ; WITH THE EXPECTED NUMBER OF CHARS TO RX ON
3285 027444 016002 003612 MOV TXRXLB(R3),R0 ; LINE (NUMBER TXED AND LOOPED BACK) TO
3286 027450 020102 MOV CHCNTB(R0),R2 ; DETERMINE IF THE POSSIBLE LOST CHAR
3287 027452 001423 CMP R1,R2 ; WOULD BE THE LAST EXPECTED RX CHAR.
3288 027454 005201 BEQ 52$ ;EXIT WITH "FAILURE" IF LOST CHR WOULD BE LAST.
3289 027456 160201 INC R1 ;DETERMINE (AS ABOVE) IF CHR0 WOULD BE THE LAST
3290 027460 016302 003552 SUB R2,R1 ; RX CHAR AND SAVE RESULT FOR LATER.
3291 027464 005202 MOV RXPTRB(R3),R2 ;GET THE RX DATA POINTER.
3292 027466 112200 INC R2 ;CALCULATE POINTER TO EXP1 LOCATION.
3293 027470 162400 MOVB (R2)+,R0 ;GET EXP1 VALUE FROM DATA PATTERN.
3294 027472 040500 SUB (R4)+,R0 ;COMPARE CHR0 AND EXP1 VALUES.
3295 BIC R5,R0 ;REMOVE INACTIVE BITS FROM RESULT. (NO ACTIVE
3296 027474 001012 BNE 52$ ; BITS ALLOWED TO LEFT OF ANY INACTIVE BITS.)
3297 027476 005701 TST R1 ;EXIT WITH "FAILURE" IF CHR0 <> EXP1.
3298 027500 001406 BEQ 50$ ;CHECK CHR0 TEST RESULT SAVED ABOVE.
3299 027502 011401 MOV (R4),R1 ;EXIT WITH "SUCCESS" IF CHR0 IS LAST CHAR.
3300 027504 100004 BPL 50$ ;GET CHR1 FROM THE QUE, DATA.VALID INTO N FLAG.
3301 027506 111200 MOVB (R2),R0 ;EXIT WITH "SUCCESS" IF CHR0 WAS LAST QUE CHAR.
3302 027510 160001 SUB R0,R1 ;GET THE EXP2 VALUE FROM THE DATA PATTERN.
3303 027512 040501 BIC R5,R1 ;COMPARE THE EXP2 AND THE CHR1 VALUES.
3304 ;REMOVE INACTIVE BITS FROM RESULT OF COMPARE.
3305 027514 001002 BNE 52$ ; (NO ACTIVE BITS LEFT OF INACTIVE BITS.)
3306 ;EXIT WITH "FAILURE" IF CHR1 <> EXP2.

```

GLOBAL SUBROUTINE

CHKLOS -

```
3307
3308      ;+
3309      ; It is likely that we lost a character from the data pattern.
3310      ; Indicate "success" and exit.
3311      ;-
3311 027516 000261      50$:      SEC          ;SET THE SUCCESS FLAG.
3312 027520 000401      BR          60$      ;EXIT THE ROUTINE.
3313
3314      ;+
3315      ; We didn't lose a single extra character at this point in the data pattern.
3316      ; Indicate "failure" and exit.
3317      ;-
3318 027522 000241      52$:      CLC          ;CLEAR THE SUCCESS FLAG.
3319
3320 027524          60$:      PASS          ;RESTORE GPRS.
3321 027526 000207          RTS      PC      JSR      PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
                                     ;CARRY - SET IF SUCCESS (LOST CHAR LIKELY).
```

GLOBAL SUBROUTINE

CHRMASK -

SEQ 0089

```

3323 .SBTTL GLOBAL SUBROUTINE - CHRMASK -
3324 ;* *****
3325 ;* - Form a Bit Mask of Unused TX/RX Bits Routine -
3326 ;* This subroutine constructs a bit mask of character bits which are not
3327 ;* used during transmission and reception. This mask can be used
3328 ;* to remove the flags, line number, DATA.VALID bits, and unused data bits
3329 ;* from a character word which has been read from the DUT FIFO.
3330 ;*
3331 ;* INPUTS: R1 - DUT LPR contents used to determine character length.
3332 ;*
3333 ;* OUTPUTS: IBM - Bit mask of unused TX/RX bits (including upper byte):
3334 ;* Examples: 177400 returned for 8 bits/char.
3335 ;* 177700 returned for 6 bits/char.
3336 ;*
3337 ;* CALLING SEQUENCE: JSR PC,CHRMASK
3338 ;*
3339 ;* COMMENTS: If this mask is to be used to just remove the inactive bits
3340 ;* within the data byte of a word read from the DUT FIFO, the
3341 ;* upper byte of the mask must be cleared.
3342 ;*
3343 ;* SUBORDINATE ROUTINES CALLED: None.
3344 ;* -- *****
3345
3346 027530 CHRMASK:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3347 027530 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3348 027534 052701 177740 BIS #177740,R1 ;PREPARE TO COUNT BITS SHIFTED INTO MASK BY
3349 027540 012703 177400 MOV #177400,R3 ; USING THE LPR BITS/CHAR FIELD CONTENTS.
3350 027544 062701 000010 2$: ADD #10,R1 ;CLEAR THE UNUSED BIT MAP LOWER BYTE.
3351 027550 103402 BCS 4$ ;DETERMINE IF ANOTHER BIT WOULD BE TOO MANY.
3352 027552 006203 ASR R3 ;EXIT THE SHIFT LOOP IF IT WOULD BE TOO MANY.
3353 027554 000773 BR 2$ ;SHIFT A BIT INTO THE UNUSED BIT MASK LOW BYTE.
3354 ;LOOP TO CHECK FOR DONE.
3355 027556 010337 002366 4$: MOV R3,IBM ;LOAD THE INACTIVE BITS MASK STORAGE IN MEMORY.
3356
3357 027562 60$: PASS ;RESTORE GPRS.
3358 027564 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3358 027564 000207 RTS PC

```

GLOBAL SUBROUTINE

- CKCHR -

```

3360 .SBTTL GLOBAL SUBROUTINE - CKCHR -
3361 ;* *****
3362 ;* - Check Character For Errors Routine -
3363 ;* This subroutine checks the character at the bottom of the resync queue
3364 ;* to determine if it is correct. Pointers and counters which are related
3365 ;* to the reception of the character are updated. If the character is
3366 ;* incorrect, an analysis of the error is done and parameters are set up
3367 ;* for the reporting of the correct error.
3368 ;*
3369 ;* INPUTS: R3 - Line offset for access of word tables of line variables.
3370 ;* R4 - Base address of the resync queue for this line.
3371 ;* R5 - Mask of the inactive bits in a TX or RX char byte.
3372 ;* BITTBL - Table of words with bits set for use in forming maps.
3373 ;* DPRSQ - Data Pattern Resync Que with valid char at bottom.
3374 ;* EXCNTB - Base of the extra character counters table.
3375 ;* RXDNF - Receive done flags.
3376 ;* RXPTRB - Base of the RX character pointers table.
3377 ;* Error Message Labels - EM9007,EM9008,EM9027,EM9028
3378 ;*
3379 ;* OUTPUTS: R1 - Contains the address of the error message to be reported.
3380 ;* R2 - Contains the actual received data.
3381 ;* R4 - Contains the expected data.
3382 ;* CARRY - "Success" flag (set if no error is found).
3383 ;* Following variables updated for line on which char was received:
3384 ;* EXCNT - Count of the number of extra chars received on line.
3385 ;* RXCNT - Count of the number of characters received on line.
3386 ;* RXPTR - Updated to point to the next expected char on line.
3387 ;* ERRBLK - Contents destroyed.
3388 ;*
3389 ;* CALLING SEQUENCE: JSR PC,CKCHR
3390 ;*
3391 ;* COMMENTS:
3392 ;*
3393 ;* SUBORDINATE ROUTINES CALLED: CHKEXT,CHKLOS,UPDCHR.
3394 ;* *****
3395 027566 CKCHR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3396 027566 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3397 ;*
3398 ;* Check for the RX of a char after RX should be complete on this line.
3399 027572 036337 002534 002654 BIT BITTBL(R3),RXDNF ;TEST THE RX DONE FLAG FOR THIS LINE.
3400 027600 001407 BEQ 2$ ;SKIP ERROR REPORT IF RX NOT COMPLETE ON LINE.
3401 ;*
3402 ;* We have received an extra character on this line.
3403 ;* Set up for error report and exit to report the error.
3404 ;* Count the extra character.
3405 ;* Exit to report "UNEXPECTED CHAR RECEIVED AFTER RX COMPLETE ON LINE nn"
3406 ;*
3407 027602 012701 022470 MOV #EM9007 R1 ;SELECT "EXTRA CHAR ON LINE" ERROR MESSAGE.
3408 027606 011402 MOV (R4),R2 ;GET THE ACTUAL DATA FOR ERROR REPORT.
3409 027610 040502 BIC R5,R2 ;REMOVE THE INACTIVE BITS.
3410 027612 052704 100000 BIS #BIT15,R4 ;INDICATE "NONE" EXPECTED DATA FOR ERROR RPT.
3411 027616 000452 BR 12$ ;GO COUNT EXTRA CHAR AND EXIT WITH "FAILURE".
3412 ;*
3413 ;* Get the pointer to the next expected receive data character.
3414 ;*
3415 027620 016302 003552 2$: MOV RXPTRB(R3),R2

```

GLOBAL SUBROUTINE

- CKCHR -

```

3416
3417
3418
3419 027624 011400
3420 027626 040500
3421 027630 111201
3422 027632 040501
3423 027634 120001
3424 027636 001003
3425 027640 004737 037370
3426 027644 000446
3427
3428
3429
3430
3431 027646 004737 027326
3432 027652 103010
3433
3434
3435
3436
3437
3438 027654 012701 023433
3439 027660 111200
3440 027662 040500
3441 027664 011402
3442 027666 040502
3443 027670 010004
3444 027672 000424
3445
3446
3447
3448
3449
3450 027674 004737 027426
3451 027700 103012
3452
3453
3454
3455
3456
3457
3458 027702 012701 023513
3459 027706 111200
3460 027710 040500
3461 027712 011402
3462 027714 040502
3463 027716 010004
3464 027720 004737 037370
3465 027724 000404
3466
3467
3468
3469
3470 027726 010002
3471 027730 010104
3472 027732 012701 022553

```

```

;+
; Compare the actual data with the expected data.
;-
      MOV      (R4),R0      ;GET THE ACTUAL DATA.
      BIC      R5,R0      ;REMOVE THE INACTIVE BITS.
      MOVB     (R2),R1      ;GET THE EXPECTED DATA.
      BIC      R5,R1      ;REMOVE THE INACTIVE BITS.
      CMPB     R0,R1      ;COMPARE ACTUAL AND EXPECTED.
      BNE      4$          ;CHECK FURTHER IF DATA MISCOMPARE.
      JSR      PC,UPDCHR    ;UPDATE PTRS AND COUNTERS FOR THE CHAR.
      BR       50$         ;EXIT WITH "SUCCESS", NO ERROR FOUND.

;+
; Actual and expected data miscompare.
; Determine if it's likely we received an extra char within the data pattern.
;-
4$:   JSR      PC,CHKEXT    ;CHECK FOR EXTRA CHAR RX'ED IN PATTERN.
      BCC      6$          ;GO CHECK FOR LOST CHAR IF NO EXTRA CHAR.

;+
; It is likely that we received an extra character within the data pattern.
; Count the char as an extra char, don't count as a standard char.
; Report "EXTRA CHAR RECEIVED WITHIN DATA PATTERN ON LINE nn"
;-
      MOV      #EM9027,R1   ;SELECT "EXTRA CHAR ON LINE" ERROR MSG.
      MOVB     (R2),R0      ;GET THE EXPECTED RECEIVE DATA.
      BIC      R5,R0      ;REMOVE THE INACTIVE BITS FROM EXPECTED DATA.
      MOV      (R4),R2      ;GET THE ACTUAL RECEIVE DATA.
      BIC      R5,R2      ;REMOVE THE INACTIVE BITS FROM ACTUAL DATA.
      MOV      R0,R4      ;PASS EXPECTED DATA TO ERROR REPORT ROUTINE.
      BR       12$         ;GO COUNT EXTRA CHAR AND EXIT WITH "FAILURE".

;+
; Actual and expected data miscompare.
; Not likely that we received an extra character within the data pattern.
; Determine if it's likely we lost a character from the data pattern.
;-
6$:   JSR      PC,CHKLOS    ;CHECK FOR A LOST CHAR CONDITION.
      BCC      8$          ;GO REPORT BAD RX DATA IF NOT LOST CHAR.

;+
; It is likely that we lost a character from the data pattern.
; Count the char in the RX char count as if it had been received.
; Also, count CHRO as a valid char, because we have verified it above.
; Report "SINGLE CHAR MISSING FROM RECEIVED DATA ON LINE nn"
;-
      MOV      #EM9028,R1   ;SELECT "LOST CHAR ON LINE" ERROR MSG. +++++
      MOVB     (R2),R0      ;GET THE EXPECTED RECEIVE DATA.
      BIC      R5,R0      ;REMOVE THE INACTIVE BITS FROM EXPECTED DATA.
      MOV      (R4),R2      ;GET THE ACTUAL RECEIVE DATA.
      BIC      R5,R2      ;REMOVE THE INACTIVE BITS FROM ACTUAL DATA.
      MOV      R0,R4      ;PASS EXPECTED DATA TO ERROR REPORT ROUTINE.
      JSR      PC,UPDCHR    ;UPDATE PTRS AND COUNTERS FOR THE CHAR.
      BR       10$         ;GO EXIT WITH "FAILURE".

;+
; Did not lose or gain a single character from/to the data pattern.
; Report "RECEIVED CHAR MISCOMPARE AGAINST TX DATA ON LINE nn"
;-
8$:   MOV      R0,R2      ;PASS ACTUAL DATUM TO ERROR REPORT ROUTINE.
      MOV      R1,R4      ;PASS EXPECTED DATUM TO ERROR REPORT ROUTINE.
      MOV      #EM9008,R1   ;SELECT THE "DATA MISCOMPARE" MESSAGE.

```


GLOBAL SUBROUTINE

CKCHR -

```

3473      ;*
3474      ; Update the character counter and RX data pattern pointer for this line.
3475      ;
3476 027736 004737 037370      10$: JSR    PC,UPDCHR      ;UPDATE RX PTR AND COUNTER FOR THIS LINE.
3477 027742 000405              BR      14$      ;GO EXIT WITH "FAILURE".
3478      ;*
3479      ; Count the character as an extra character.
3480      ;
3481 027744 005263 003412      12$: INC     EXCNTB(R3)      ;INCREMENT THE EXTRA CHAR COUNT FOR THIS LINE.
3482 027750 001002              BNE     14$      ;EXIT WITH FAILURE IF NO OVERFLOW.
3483 027752 005363 003412      DEC     EXCNTB(R3)      ;DECREMENT BACK TO -1 (MAX VALUE) IF OVERFLOW.
3484      ;*
3485      ; Indicate "failure" and exit.
3486      ;
3487 027756 000241              14$: CLC              ;CLEAR THE "SUCCESS" FLAG.
3488 027760 000401              BR      60$      ;EXIT THE ROUTINE.
3489      ;*
3490      ; No error was found.
3491      ; Set "success" flag and exit.
3492      ;
3493      ;
3494 027762 000261              50$: SEC              ;SET THE "SUCCESS" FLAG.
3495      ;
3496 027764              60$: PASS    R1,R2,R4      ;RESTORE GPRS, EXCEPT
                                MOV     R1,R1SLOT(SP)      ;PUT R1 IN STACK SLOT.
                                MOV     R2,R2SLOT(SP)      ;PUT R2 IN STACK SLOT.
                                MOV     R4,R4SLOT(SP)      ;PUT R4 IN STACK SLOT.
                                JSR     PC,0(SP)          ;RETURN TO PREG05 SUBRT.
3497      ;R1 - CONTAINS THE ADDRESS OF THE ERROR REPORT.
3498      ;R2 - CONTAINS THE ACTUAL DATA RECEIVED
3499      ;R4 - CONTAINS THE EXPECTED DATA.
3500 030002 000207              RTS     PC

```

GLOBAL SUBROUTINE

CKINAC

```

3502 .SBTTL GLOBAL SUBROUTINE          - CKINAC -
3503 ;.. *****
3504 ;*      - Check for New Character on Inactive Line Routine -
3505 ;*      This subroutine checks a character to determine if the character
3506 ;*      was received on an active line.  If the character was received on
3507 ;*      an inactive line this routine records the fact that the character
3508 ;*      was received on an inactive line, prepares an error message for
3509 ;*      the calling routine, and returns a "failure" status.
3510 ;*
3511 ;* INPUTS:      R2 - The RX character including error flags and line number.
3512 ;*              ACTLNS - Bit map of active DUT lines.
3513 ;*              BITTBL - Table of words with bits set for forming bit maps.
3514 ;*              EM9006 - Label at "RX ON INACTIVE LINE" error message.
3515 ;*              EXCNTB - Base of the extra character counters table.
3516 ;*              TXRXLB - Base of TX/RX line number association table.
3517 ;*
3518 ;* OUTPUTS:     CARRY "Success" flag (set if no error found).
3519 ;*              R1 - If error found, address of error message.
3520 ;*              R3 - Line number offset of passed in character.
3521 ;*              R4 - If error found, expected data indication for error rpt.
3522 ;*              EXCNT - Extra character count for line (Updated if error).
3523 ;*
3524 ;* CALLING SEQUENCE:  JSR      PC,CKINAC
3525 ;*
3526 ;* COMMENTS:
3527 ;*
3528 ;* SUBORDINATE ROUTINES CALLED: NONE.
3529 ;.. *****
3530
3531 030004      030004      004537      005474      CKINAC:: SAVE
3532 ;..
3533 ; Extract the line number from the passed in character and use the line
3534 ; number to form an offset for accessing tables of line variables.
3535 ;
3536 030010      010203      MOV      R2,R3          ;EXTRACT THE LINE NUMBER
3537 030012      000303      SWAB     R3              ; FROM THE CHARACTER WE
3538 030014      042703      BIC      0177760,R3     ; ARE COMPARING.
3539 030020      006303      ASL      R3              ;FORM OFFSET INTO WORD TABLE FROM LINE NUMBER.
3540 ;
3541 ; If the character in question is not a valid character, exit with "success".
3542 ;..
3543 030022      005702      TST      R2              ;CHECK DATA.VALID BIT.
3544 030024      100021      BPL      50$             ;EXIT WITH SUCCESS IF CHAR IS NOT VALID.
3545 ;
3546 ; If the TX line which is associated with this RX line is an active line,
3547 ; exit the routine with "success".
3548 ;
3549 030026      016301      005364      MOV      TXRXLB(R3),R1  ;GET THE TX LINE # OFFSET FOR THIS RX LINE.
3550 030032      036137      002534      002272      BIT      BITTBL(R1),ACTLNS ;DETERMINE IF TX LINE IS AN ACTIVE LINE.
3551 030040      001013      BNE      50$             ;EXIT ROUTINE WITH SUCCESS IF LINE IS ACTIVE.
3552 ;
3553 ; The character in question was received on an inactive line.
3554 ; Count this character as an extra char.
3555 ; Set up error information.
3556 ; Exit routine with "failure" indication.
3557 ;..

```


GLOBAL SUBROUTINE

CKINAC -

```

3558 030042 005263 003412          INC    EXCNTB(R3)      ;INCREMENT THE EXTRA CHAR COUNT FOR THIS LINE.
3559 030046 001002                  BNE     2$           ;SKIP SETTING TO MAX VALUE IF NO OVERFLOW.
3560 030050 005363 003412          DEC     EXCNTB(R3)      ;DECREMENT BACK TO -1 (MAX VALUE) IF OVERFLOW.
3561 030054 012701 022415          2$:  MOV     @EM9006,R1    ;SET UP RX ON INACTIVE LINE MESSAGE.
3562 030060 012704 100000          MOV     @BIT15,R4       ;SET UP "NONE" EXPECTED DATA INDICATION.
3563 030064 000241                  CLC                    ;CLEAR THE "SUCCESS" FLAG.
3564 030066 000401                  BR      60$           ;GO REPORT RX CHAR ON INACTIVE LINE.
3565
3566                               ;*
3567                               ; We have not found a "char on inactive line" error situation.
3568                               ; Set the "success" flag and exit the routine.
3569                               ;
3570 030070 000261          50$:  SEC                        ;SET THE "SUCCESS" FLAG.
3571
3572 030072                  60$:  PASS     R1,R3,R4         ;RESTORE GPRS, EXCEPT OUTPUT GPRS.
                                MOV     R1,R1SLOT(SP)       ;PUT R1 IN STACK SLOT.
                                MOV     R3,R3SLOT(SP)       ;PUT R3 IN STACK SLOT.
                                MOV     R4,R4SLOT(SP)       ;PUT R4 IN STACK SLOT.
                                JSR     PC,@(SP)+           ;RETURN TO PREG05 SUBRT.
3573 030110 000207          RTS     PC                    ;CARRY - SUCCESS FLAG (SET IF NO ERROR).

```

GLOBAL SUBROUTINE

- CKTRAP -

```

3575 .SBTTL GLOBAL SUBROUTINE - CKTRAP -
3576 ;*****
3577 ;* Check Trap Routine -
3578 ;* This subroutine is used to check for a bus time-out trap (004 trap)
3579 ;* which is caused by an access to a non-existent memory or I/O location.
3580 ;* If the trap does not occur, this routine returns a success indication.
3581 ;*
3582 ;* INPUTS: R0 - Source address for move.
3583 ;* R1 - Destination address for move.
3584 ;* (R0) - Source for the move.
3585 ;*
3586 ;* OUTPUTS: (R1) Written to the contents of (R0).
3587 ;* Carry flag - Set on return if no 004 trap detected.
3588 ;* TP4FLG - Nonzero if trap occurred, cleared otherwise.
3589 ;*
3590 ;* CALLING SEQUENCE: JSR PC,CKTRAP
3591 ;*
3592 ;* COMMENTS: If this subroutine causes a trap, either the address which
3593 ;* is labeled ADRPTR will be the trap PC address on the stack.
3594 ;*
3595 ;* SUBORDINATE ROUTINES CALLED: None.
3596 ;*****
3597
3598 030112 CKTRAP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3599 030112 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3600 030116 005037 002424 CLR TP4FLG ;CLEAR THE 004 TRAP FLAGS.
3601 030122 011011 MOV (R0),(R1) ;PERFORM THE MOVE IN QUESTION.
3602 030124 005737 002424 ADRPTR:: TST TP4FLG ;CHECK FOR OCCURENCE OF TRAP.
3603 030130 000261 SEC ;INDICATE SUCCESS.
3604 030132 001401 BEQ 60$ ;EXIT WITH SUCCESS IF TRAP DID NOT OCCUR.
3605 030134 000241 CLC ;INDICATE FAILURE.
3606 030136 004736 60$: PASS ;RESTORE GPRS.
3607 030140 000207 RTS PC JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.

```

GOBAL SUBROUTINE

- CKTRPB -

```

3608 .SBTTL GOBAL SUBROUTINE - CKTRPB -
3609 ;*****
3610 ;* CHECK FOR TRAP -
3611 ;* THIS SUBROUTINE IS USED TO CHECK FOR A BUS TIME-OUT TRAP (004 TRAP)
3612 ;* WHICH IS CAUSED BY AN ACCESS TO A NON-EXISTENT MEMORY OR I/O LOCATION
3613 ;* IF A TRAP DOES NOT OCCUR, THIS ROUTINE RETURNS A SUCCESS INDICATION.
3614 ;*
3615 ;* INPUTS: R0 - SOURCE ADDRESS FOR MOVE
3616 ;* R1 - DESTINATION ADDRESS FOR MOVE
3617 ;* (R0) - SOURCE FOR THE MOVE
3618 ;*
3619 ;* OUTPUTS: (R1) - WRITEN TO THE CONTENTS OF (R0)
3620 ;* CARRY FLAG - SET ON RETURN IF NO 004 TRAP DETECTED
3621 ;* TP4FLG - NONZERO IF TRAP OCCURED, CLEARED OTHERWISE.
3622 ;*
3623 ;* CALLING SEQUENCE: JSR PC,CKTRPB
3624 ;*
3625 ;* COMMENTS: IF THIS SUBROUTINE CAUSES A TRAP, EITHER THE ADDRESS
3626 ;* WHICH IS LABELED TRPAD2 WILL BE THE TRAP PC ADDRESS ON
3627 ;* THE STACK OR SOME OTHER ADDRESS WHICH WAS PLACED ON
3628 ;* THE STACK BY AN UNEXPECTED TRAP.
3629 ;* THIS ROUTINE PERFORMS A BYTE MOV .
3630 ;*
3631 ;* SUBORDINATE ROUTINES CALLED: NONE.
3632 ;*****
3633
3634 030142 CKTRPB:: SAVE JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3635 030142 004537 005474
3636 030146 005037 002424 CLR TP4FLG ;CLEAR THE 004 TRAP FLAGS
3637 030152 111011 MOV (R0),(R1) ;PERFORM THE BYTE MOVE
3638 030154 005737 002424 TRPAD2:: TST TP4FLG ;CHECK FOR OCCURENCE OF TRAP
3639 030160 000261 SEC ;INDICATE SUCCESS
3640 030162 001401 BEQ 60$ ;EXIT WITH SUCCESS IF TRAP DID NOT OCCUR
3641 030164 000241 CLC ;INDICATE FAILURE
3642 030166 004736 60$: PASS
3643 030170 000207 RTS PC JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
;RETURN

```

GLOBAL SUBROUTINE

- CLNRST -

```

3645 .SBTTL GLOBAL SUBROUTINE - CLNRST -
3646 ;*****
3647 ;* Clean Reset of the Device Under Test -
3648 ;* This subroutine is used to reset the DUT to a known state.
3649 ;* The DUT's self-test is skipped, and the fifo is purged of any error
3650 ;* codes, etc.
3651 ;* If the reset does not successfully complete, then the carry bit is
3652 ;* passed back to the calling routine (clear).
3653 ;*
3654 ;* INPUTS: CSRA - Contains the address of the CSR
3655 ;* TXBFCA - Contains address of DUT DMA Buffer Count register.
3656 ;* ERRNBR Error number for possible error report.
3657 ;* ERRTBL ERRTP,ERNBR,and ERRMSG set up correctly.
3658 ;*
3659 ;* OUTPUTS: The DUT performs its reset function into a known state.
3660 ;* CARRY - Clear indicates the test is to be aborted.
3661 ;* ERRBLK - value may be destroyed.
3662 ;* IESTAT - TX and RX interrupt flags are cleared.
3663 ;* TX and RX interrupt enable bits in the DUT's CSR are cleared.
3664 ;*
3665 ;* CALLING SEQUENCE: JSR PC,CLNRST
3666 ;*
3667 ;* COMMENTS: This subroutine can report errors with numbers ERRNBR.
3668 ;* This routine does not destroy the value of ERRNBR.
3669 ;*
3670 ;* SUBORDINATE ROUTINES CALLED: DELAY,MSLGET,PUFIFO,RESETT.
3671 ;*****
3672
3673 030172 CLNRST:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3674 030172 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3675 ;*
3676 ; Reset the DUT.
3677 ; This routine reports errors with numbers from ERRNBR thru ERRNBR+2.
3678 ;
3678 030176 004737 034530 JSR PC,RESETT ;RESET THE DUT TO A KNOWN STATE.
3679 030202 103002 BCC 60$ ;EXIT ROUTINE WITH ABORT TEST INDICATOR.
3680 ;*
3681 ; Purge the FIFO of error codes, save any BMP codes found.
3682 ;
3683 030204 004737 032722 JSR PC,PUFIFO ;PURGE THE FIFO.
3684
3685 030210 60$: ;EXIT THE TEST USING RESETT OR PUFIFO STATUS.
3686 030210 PASS ;RESTORE GPRS, PASS THE FOLLOWING INTACT:
3687 030210 004736 JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
3688 030212 000207 RTS PC ;CARRY BIT:IF CLEAR, THEN ABORT THE TEST.

```

GLOBAL SUBROUTINE

- CLR16W -

```

3690 .SBTTL GLOBAL SUBROUTINE - CLR16W -
3691 ;* *****
3692 ;* - Clear Sixteen Words Routine -
3693 ;* This subroutine clears 16 words starting with the specified word.
3694 ;*
3695 ;* INPUTS: R0 - Address of the first word to clear.
3696 ;*
3697 ;* OUTPUTS: (R0) to (R0+15) - 16 words of memory are cleared to 0.
3698 ;*
3699 ;* CALLING SEQUENCE: JSR PC,CLR16W
3700 ;*
3701 ;* COMMENTS:
3702 ;*
3703 ;* SUBORDINATE ROUTINES CALLED: None.
3704 ;* -- *****
3705
3706 030214 CLR16W:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
      030214 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3707 030220 012701 000020      MOV #16.,R1 ;SET THE LOOP COUNTER TO 16.
3708 030224 005020      2$: CLR (R0)+ ;CLEAR A WORD OF MEMORY.
3709 030226 005301      DEC R1 ;COUNT THIS LOOP.
3710 030230 001375      BNE 2$ ;LOOP IF NOT 16 WORD CLEARED.
3711 030232      60$: PASS ;RESTORE GPRS.
      030232 004736      JSR PC,8(SP)+ ;RETURN TO PREG05 SUBRT.
3712 030234 000207      RTS PC

```

GLOBAL SUBROUTINE

- CNTERR -

```

3714 .SBTTL GLOBAL SUBROUTINE - CNTERR -
3715 ;* *****
3716 ;* - Count Error Routine -
3717 ;* This subroutine is used to count a "data" error on the specified
3718 ;* line. It checks whether error summary reporting is active, or should
3719 ;* be made active on this line, and activates it if necessary.
3720 ;*
3721 ;* INPUTS: R5 - Line number of line under consideration.
3722 ;* ERCNTB - Label at base of error counters table.
3723 ;* ERSMRF - Error summary flags (bit set if line in summary mode).
3724 ;* NDERPT - Number of individual data errors to report on a line.
3725 ;*
3726 ;* OUTPUTS: CARRY - Set if line is in error summary mode.
3727 ;* ERCNT - Error counter incremented for specified line.
3728 ;* ERSMRF - Bit set if line should be in summary mode.
3729 ;*
3730 ;* CALLING SEQUENCE: JSR PC,CNTERR
3731 ;*
3732 ;* COMMENTS:
3733 ;*
3734 ;* SUBORDINATE ROUTINES CALLED: None.
3735 ;* - *****
3736
3737 030236 CNTERR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
030236 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3738
3739 ; Count the error on the counter for the specified line.
3740 ; -
3741 030242 006305 ASL R5 ;FORM WORD OFFSET FROM LINE NUMBER.
3742 030244 016501 003452 MOV ERCNTB(R5),R1 ;GET THE PRESENT ERROR COUNT FOR THIS LINE.
3743 030250 005201 INC R1 ;COUNT ERROR.
3744 030252 103402 BCS 2$ ;OVERFLOW? YES, DON'T UPDATE COUNTER IN TABLE.
3745 030254 010165 003452 MOV R1,ERCNTB(R5) ;UPDATE ERROR COUNTER TABLE ENTRY.
3746 030260 005737 002264 2$: TST NDERPT
3747 030264 001411 BEQ 60$ ;SUMMARYS DISABLED? YES, EXIT WITH CARRY 0.
3748 030266 020137 002264 CMP R1,NDERPT ;NO, CHECK FOR ENOUGH ERRORS FOR SUMMARY USE.
3749 030272 101002 BHI 4$ ;ENOUGH ERRORS TO USE SUMMARY? YES, GO HANDLE.
3750 030274 000241 CLC ;INDICATE NOT TO USE SUMMARY REPORT YET.
3751 030276 000404 BR 60$ ;EXIT WITH CARRY 0.
3752 030300 056537 002534 002650 4$: BIS BITTBL(R5),ERSMRF ;SET THE ERROR SUMMARY FLAG FOR LINE.
3753 030306 000261 SEC ;INDICATE TO USE SUMMARY REPORT.
3754 030310 004736 60$: PASS ;RESTORE GPRS.
030310 000207 JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
3755 030312 000207 RTS PC

```


GLOBAL SUBROUTINE

CONMAP -

```

3757 .SBTTL GLOBAL SUBROUTINE - CONMAP -
3758 ;* *****
3759 ;* - Convert Line bit map.
3760 ;* This subrouitne is used to convert a bit map passed to it , into
3761 ;* another line bit map that is based upon the associated TX/RX line
3762 ;* number/offset table.
3763 ;*
3764 ;* INPUTS: R5 Contains the line bit map to be transformed.
3765 ;* TXRXLB - Base address of associated TX/RX line number table.
3766 ;*
3767 ;* OUTPUTS: R5 Contains an associated line bit map.
3768 ;*
3769 ;* CALLING SEQUENCE: JSR PC,CONMAP
3770 ;*
3771 ;* COMMENTS: The TX/RX association table must be initialised before this
3772 ;* routine is called.
3773 ;*
3774 ;* SUBORDINATE ROUTINES CALLED: NONE.
3775 ;* *****
3776
3777 030314 CONMAP::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
030314 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3778 030320 012702 005364 MOV #TXRXLB,R2 ;GET THE BASE ADDRESS OF THE LINE ASSOC TABLE.
3779 030324 010503 MOV R5,R3 ;COPY THE BIT MAP TO BE TRANSFORMED.
3780 030326 012704 000010 MOV #NUMLNS,R4 ;SET MAX LINE COUNTER.
3781 030332 005005 CLR R5 ;CLEAR ASSOCIATED LINE BIT MAP.
3782 030334 006203 2$: ASR R3 ;SHIFT ACTLNS BIT MAP INT BOOLEAN REGISTER.
3783 030336 103005 BCC 4$ ;SKIP SETTING ASSOCIATED LINE NUMBER BIT MAP.
3784 030340 011201 MOV (R2),R1 ;GET ASSOCIATED LINE NUMBER OFFSET FROM TABLE.
3785 030342 006201 ASR R1 ;SHIFT RIGHT TO GET LINE NUMB FROM OFFSET.
3786 030344 004737 031646 JSR PC,LINBIT ;GENERATE AN SINGLE BIT MAP FOR THIS LINE.
3787 030350 050005 BIS R0,R5 ;SET BIT FOR THIS LINE IN ASSOCIATED BIT MAP.
3788 030352 005722 4$: TST (R2)+ ;INCREMENT ADDRESS FOR THE NEXT LINE NUMBER.
3789 030354 005304 DEC R4 ;DECREMENT LINE COUNT.
3790 030356 001366 BNE 2$ ;LOOP IF NOT DONE.
3791 030360 60$: PASS R5 ;RESTORE GPRS, EXCEPT
030360 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
030364 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
3792 ;R5 CONTAINS THE ASSOCIATED LINE BIT MAP.
3793 030366 000207 RTS PC

```

GLOBAL SUBROUTINE

- DELAY

```

3795      .SBTTL GLOBAL SUBROUTINE                      DELAY
3796      ;*****
3797      ;*      - DELAY SUBROUTINE -
3798      ;*      This subroutine is used to delay a variable number of milli seconds.
3799      ;*
3800      ;* INPUTS:      R4 - Contains the number of ms to delay.
3801      ;*              MSLCNT.
3802      ;*
3803      ;* OUTPUTS:     None.
3804      ;*
3805      ;* CALLING SEQUENCE:  JSR      PC,DELAY
3806      ;*
3807      ;* COMMENTS:      If no hardware clock interrupts are occurring, control-Cs will
3808      ;*                  not be honored for the duration of the delay.
3809      ;*
3810      ;* SUBORDINATE ROUTINES CALLED: None.
3811      ;*****
3812
3813 030370      DELAY:: SAVE                                ;SAVE CONTENTS OF GPRS R0 THRU R5.
3814      030370      004537      005474                    R5,PREG05      ;CALL REGISTER SAVE SUBRT.
3815      030374      010401                    MOV      R4,R1      ;PASS NUMBER OF MS DELAY AS TIME-OUT VALUE.
3816      030376      012702      177777                    MOV      #-1,R2      ;TELL MSLOOP ROUTINE TO CHECK ALL BITS.
3817      030402      005003                    CLR      R3          ;TELL MSLOOP RTN TO CHECK FOR ALL BITS CLEAR.
3818      030404      012704      030426                    MOV      #62$,R4      ;TELL MSLOOP TO CHECK DUMMY NON-ZERO WORD.
3819      030410      004737      032042                    JSR      PC,MSLOOP      ;DELAY THE REQUESTED # OF MS.
3820      030414      103002                    BCC      60$        ;EXIT ROUTINE IF WE TIMED-OUT.]
3821      030416      004737      032412                    JSR      PC,00PS      ;IF NO TIME-OUT, BAD PROGRAM OR HOST MACHINE.
3822      030422      004736                    60$: PASS          ;RESTORE GPRS.
3823      030424      000207                    RTS      PC          PC,8(SP)+      ;RETURN TO PREG05 SUBRT.
3824      030426      177777                    62$: .WORD      -1      ;DUMMY, NON-ZERO WORD.

```


GLOBAL SUBROUTINE

DM16B

SEQ 0102

```

3826 .SBTTL GLOBAL SUBROUTINE      - DM16B -
3827 ;+ *****
3828 ;*          CONVERT TO A 16 BIT PHYSICAL ADDRESS -
3829 ;*          THIS ROUTINE CONVERTS FROM PAR FORM TO A 16-BIT PHYSICAL ADDRESS,
3830 ;*          OF ALTERNATE 1'S AND 0'S.
3831 ;*
3832 ;*          INPUTS:          DMTSTA: - CONTAINS THE ADDRESS IN PAR FORM
3833 ;*
3834 ;*          OUTPUTS:         RO - CONTAINS THE 16 BIT PHYSICAL ADDRESS
3835 ;*
3836 ;*          CALLING SEQUENCE:      JSR      PC,DM16B
3837 ;*
3838 ;*          COMMENTS:          USED IN THE DMA ADDRESS TEST
3839 ;*
3840 ;*          SUBROUTINES CALLED:    NONE.
3841 ;* *****
3842 ;- *****
3843 030430 004537 005474  DM16B:: SAVE
3844 030434 013700 002432      MOV      DMTSTA,RO      JSR      R5,PREG05      ;CALL REGISTER SAVE SUBRT.
3845 030440 012702 000006      MOV      #6,R2      ;SHIFT THE DMA TEST ADDRESS
3846 030444 006300      2$:  ASL      RO      ;SIX PLACES LEFT , TO
3847 030446 005302      DEC      R2      ;CONVERT IT INTO A ,
3848 030450 001375      BNE      2$      ;16-BIT PHYSICAL ADDRESS
3849 ;
3850 030452 012701 000052      MOV      #52,R1      ;SET UP THE 6 LSB'S
3851 030456 032700 000100      BIT      #100,RO      ;IF BIT #6 OF THE PHYSICAL
3852 030462 001402      BEQ      4$      ;ADDRESS IS CLEAR THEN BRANCH
3853 030464 012701 000025      MOV      #25,R1      ;OTHERWISE CORRECT THE LSB'S
3854 ;
3855 030470 060100      4$:  ADD      R1,RO      ;MREGE THE LSB'S WITH THE PHY ADDR
3856 ;
3857 030472      PASS      RO      ;RETURN WITH THE PHY ADDR.
3858 030500 000207      RTS      PC      MOV      RO,ROSL0T(SP)      ;PUT RO IN STACK SLOT.
3859 ;                               JSR      PC,@(SP)+      ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE - DM16R -

SEQ 0103

```

3861
3862 .SBTTL GLOBAL SUBROUTINE - DMRW -
3863 ;* *****
3864 ;* - READ/WRITE DATA FROM/TO (DMTSTA) -
3865 ;* THIS ROUTINE READS DATA BYTES FROM OR WRITES DATA BYTES TO AN ADDR OF
3866 ;* ALTERNATE 1'S AND 0'S . BITS 21 TO 6 OF THE ADDR ARE CONTAINED AT
3867 ;* DMTSTA. THE ROUTINE APPENDS THE 6 LSB'S TO PRODUCE AN ADDR OF ALTERNATE
3868 ;* 1'S AND 0'S. THIS ROUTINE IS CALLED FROM THE DMA ADDRESS TEST.
3869 ;*
3870 ;* INPUTS:          R0 - ADDRESS OF THE DATA TO BE WRITTEN TO (DMTSTA),
3871 ;*                  IF A WRITE IS SPECIFIED.
3872 ;*                  R1 - ADDRESS OF THE AREA IN WHICH DATA FROM (DMTSTA),
3873 ;*                  IS TO BE SAVED, IF A READ IS SPECIFIED.
3874 ;*                  R3 - NUMBER OF DATA BYTES TO BE READ/WRITTEN
3875 ;*                  R5 - CLEAR , SPECIFIES A READ FROM (DMTSTA)
3876 ;*                  SET , SPECIFIES A WRITE TO (DMTSTA).
3877 ;*                  DMTSTA - CONTAINS BITS 21 TO 6 OF THE ADDR.
3878 ;*                  MMSRO - ADDRESS OF MEM MGT STATUS REG #0
3879 ;*                  MMPRES - BIT #0 SET, INDICATES MEM MGT PRESENT
3880 ;*                  PARA6 - ADDRESS OF MEM MGT PAR #6
3881 ;*                  TP4FLG - 004 TRAP FLAGS
3882 ;*
3883 ;* OUTPUTS:         DATA AT (DMTSTA) SAVED OR WRITTEN
3884 ;*                  PAR #6 - CONTENTS SET TO CONTENTS OF DMTSTA
3885 ;*                  TP4FLG - CLEAR IF READ/WRITE SUCCESSFUL
3886 ;*                  SET IF FAIL.
3887 ;*
3888 ;* CALLING SEQUENCE:      JSR      PC,DMRW
3889 ;*
3890 ;* COMMENTS:            IF MEM MGT IS PRESENT THE SUBROUTINE USES (DMTSTA)
3891 ;*                      AS THE PAGE ADDRESS , PLACING IT IN PAR #6, AND CREATES
3892 ;*                      A VIRTUAL ADDR IN THE RANGE OF PAR #6 WHICH CONTAINS
3893 ;*                      THE SIX LSB'S.
3894 ;*                      IF IT IS NOT PRESENT THE (DMTSTA) IS CONVERTED INTO
3895 ;*                      THE EQUIVALENT 16 BIT PHYSICAL ADDRESS.
3896 ;*
3897 ;* SUBORDINATE ROUTINES CALLED: CKTRAP,DM16B.
3898 ;*
3899 ;* *****
3900 030502 DMRW:: SAVE
3901 030502 004537 005474 JSR      R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3902 030506 010004 MOV      R0,R4 ;SAVE THE SOURCE ADDR
3903 030510 005737 TST      MMPRES ;IF MEM MGT IS PRESENT THEN
3904 030514 001003 BNE      6# ;JUMP AND SET UP THE PAR #6
3905 030516 004737 JSR      PC,DM16B ;OTHERWISE CONVERT DMTSTA INTO A 16-BIT
3906 ;PHYSICAL ADDRESS, IN R0.
3907 030522 000416 BR      10# ;JUMP TO PERFORM THE MOVE
3908 030524 013777 002432 151756 6#: MOV      DMTSTA,PAR6A ;SET PAR #6
3909 030532 012700 140052 MOV      #140052,R0 ;SET THE SIX LSB'S AND CONVERT TO
3910 ;A VIRTUAL ADDRESS WITHIN THE INFLUENCE
3911 ;OF PAR #6.
3912 030536 032737 000001 002432 BIT      #1,DMTSTA ;IF BIT #0 OF DMTSTA IS CLEAR THEN
3913 030544 001402 BEQ      8# ;AVOID CHANGING THE LSB'S
3914 030546 012700 140025 MOV      #140025,R0 ;CHANGE THE LSB'S
3915 030552 012777 000001 151704 8#: MOV      #BIT0,MMSRO ;ENABLE MEM MGT.
3916 030560 005705 10#: TST      R5 ;IF A READ IS SPECIFIED THEN

```

GLOBAL SUBROUTINE

- DMRW -

```

3917 030562 001402      BEQ      12#      ;AVOID SWAPING THE SOURCE AND DESTINATION.
3918 030564 010001      MOV      R0,R1    ;SWAP
3919 030566 010400      MOV      R4,R0    ;RESTORE THE ORIGINAL SOURCE FOR THE MOVE.
3920 030570 004737 030142 12# : JSR      PC,CKTRPB ;PERFORM THE BYTE MOVE.
3921 030574 103004      BCC      14#      ;EXIT IF A TRAP OCCURED.
3922 030576 005201      INC      R1        ;INCREMENT THE DESTINATION ADDRESS
3923 030600 005200      INC      R0        ;INCREMENT THE SOURCE ADDR.
3924 030602 005303      DEC      R3        ;DECREMENT THE DATA
3925 030604 001371      BNE      12#      ;REPEAT UNTIL ALL DATA READ/WITTEN
3926 030606 005737 002470 14# : TST      MMPRES ;IF MEM MGT IS PRESENT THEN
3927 030612 001402      BEQ      16#      ;
3928 030614 005077 151644 16# : CLR      @MMSR0 ;DISABLE IT.
3929 030620 004736      PASS
3930 030622 000207      RTS      PC      JSR      PC,@(SP), ;RETURN TO PREG05 SUBRT.
3931

```

GLOBAL SUBROUTINE

DODMA -

```

3933 .SBTTL GLOBAL SUBROUTINE - DODMA -
3934 ;* *****
3935 ;* - Initiate DMA Transmission Routine -
3936 ;* This routine writes the DMA parameter to the specified device and
3937 ;* initiates the DMA transmission.
3938 ;*
3939 ;* INPUTS: R1 - Line number on which to initiate the DMA.
3940 ;* R2 - Start address of the DMA buffer (16 bit virtual).
3941 ;* R3 - Character count of the DMA buffer.
3942 ;* CSRA - Contains address of the DUT CSR.
3943 ;* IESTAT - Storage for states of the interrupt enable bits.
3944 ;* MMENAB - Memory management flag (0 if MEM MGT not enabled).
3945 ;* HOST MEM MGT PAR REGISTERS - If MEM MGT is in use.
3946 ;* TXAD1A - Contains address of DMA TX buffer address reg #1.
3947 ;* TXAD2A - Contains address of DMA TX buffer address reg #2.
3948 ;* TXBFCA - Contains address of DMA character count register.
3949 ;*
3950 ;* OUTPUTS: CARRY - Success flag (set if DMA_START found clear).
3951 ;* DUT TBUFFAD1 - LS 16 bits of DMA buffer address (initialized).
3952 ;* DUT TBUFFAD2 - MS 6 bits of DMA buffer address (initialized).
3953 ;* DMA_START bit set.
3954 ;* DUT TBUFFCT - DMA buffer character count (initialized).
3955 ;*
3956 ;* CALLING SEQUENCE: JSR PC,DODMA
3957 ;*
3958 ;* COMMENTS: This routine determines if Memory Management is being used
3959 ;* and sets up the full 22 bit physical address if necessary.
3960 ;*
3961 ;* SUBORDINATE ROUTINES CALLED: None.
3962 ;* *****
3963
3964 030624 DODMA:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
3965 030624 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
3966 030630 012704 000200 MOV #200,R4 ;PREPARE TO CLEAR UPPER 6 BITS OF DMA BUFF ADR.
3967 030634 005737 002472 TST MMENAB ;CHECK FOR MEMORY MANAGEMENT IN USE.
3968 030640 001427 BEQ 6$ ;GOTO SET UP DEVICE IF MEM MGT NOT IN USE.
3969 ;*
3970 ;* Memory management is in use.
3971 ;* Construct 22 bit physical address from the 16 bit virtual address.
3972 ;*
3973 030642 010205 MOV R2,R5 ;STRIP THE MOST SIGNIFICANT 3 BITS OF THE
3974 030644 012700 000005 MOV #5,R0 ;DMA BUFFER VIRTUAL ADDRESS AND MULTIPLY
3975 030650 006105 2$: ROL R5 ;THEIR VALUE BY TWO TO GET AN OFFSET INTO
3976 030652 005300 DEC R0 ;THE TABLE OF MEMORY MANAGEMENT PAGE
3977 030654 001375 BNE 2$ ;ADDRESS REGISTERS (PAR).
3978 030656 042705 177761 BIC #177761,R5 ;
3979 030662 063705 002474 ADD PAR0A,R5 ;ADD IN THE BASE VALUE OF THE MM PAR REGISTERS.
3980 030666 011505 MOV (R5),R5 ;GET THE 16 BIT PHYSICAL ADDRESS BLOCK COUNT.
3981 030670 012700 000006 MOV #6,R0 ;SHIFT UPPER 6 BITS OF THE PHYSICAL ADDRESS
3982 030674 006305 4$: ASL R5 ;BLOCK COUNT (GOTTEN FROM THE PROPER PAR)
3983 030676 006104 ROL R4 ;INTO THE LS 6 BITS OF THE WORD TO WRITE
3984 030700 005300 DEC R0 ;INTO THE DUT TBUFFAD2 REGISTER.
3985 030702 001374 BNE 4$ ;
3986 030704 042702 160000 BIC #160000,R2 ;ADD THE 13 BIT DISPLACEMENT FIELD FROM VIRTUAL
3987 030710 060502 ADD R5,R2 ;ADR TO THE SHIFTED BLOCK NUMBER FROM THE
3988 030712 005504 ADC R4 ;MEMORY MANAGEMENT PAR.
3989 030714 052704 000200 BIS #200,R4 ;SET THE DMA START BIT IN WORD FOR TBUFFAD2.

```

GLOBAL SUBROUTINE

DODMA -

```

3989
3990      ;
3991      ; Write the DMA parameters out to the DUT DMA registers.
3992      ; Disable interrupts.
3993      ; Set up DUT CSR IND.ADR.REG field.
3994      ; Write the DMA transmit character count.
3995      ; Write the least significant 16 bits of the DMA buffer start address.
3996      ; Write the most significant 6 bits of the address,
3997      ; setting the DMA_START bit, and initiating the DMA transmission.
3998 030720 106705
3999 030722 106427 000340
4000 030726 053701 002374
4001 030732 010177 151342
4002 030736 105777 151352
4003 030742 000241
4004 030744 100410
4005 030746 010377 151344
4006 030752 010277 151334
4007 030756 110477 151332
4008 030762 106405
4009 030764 000261
4010
4011 030766
      030766 004736
4012 030770 000207

6$:      MFPS      R5      ;GET THE PRESENT PROCESSOR PRIORITY.
      MTPS      @PRI07    ;DISABLE ALL HARDWARE INTERRUPTS.
      BIS      IESTAT,R1  ;PREPARE FOR SETUP OF LINE NUMBER IN DUT CSR.
      MOV      R1,@CSRA   ;SET UP THE DUT CSR IND.ADR.REG FIELD.
      TSTB     @TXAD2A    ;TEST THE DUT DMA_START BIT.
      CLC
      BMI      60$        ;INDICATE FAILURE IN CASE DMA.HO BIT IS SET.
      MOV      R3,@TXBFCA ;EXIT WITH FAILURE IF DMA.HO BIT IS SET.
      MOV      R2,@TXAD1A ;WRITE THE DMA CHARACTER COUNT.
      MOVB     R4,@TXAD2A ;WRITE THE LS 16 BITS OF BUFFER ADDRESS.
      MTPS     R5         ;WRITE MS 6 BITS OF ADR AND START DMA TX.
      SEC      ;RESTORE THE PROCESSOR PRIORITY.
      ;INDICATE SUCCESS.

60$:      PASS
      RTS      PC      JSR      ;RESTORE GPRS,
      ;PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
      ; CARRY - SUCCESS FLAG (SET IF SUCCESS).

```

GLOBAL SUBROUTINE

- FINACT

```

4014 .SBTTL GLOBAL SUBROUTINE FINACT -
4015 ;* *****
4016 ;* - FIND FIRST ACTIVE LINE -
4017 ;* This subroutine calculates the number of the first active line that
4018 ;* is found in the active line bit map ACTLNS.
4019 ;*
4020 ;* INPUTS: ACTLNS Contains the active line bit map.
4021 ;*
4022 ;* OUTPUTS: R1 - Contains the number of the first active line.
4023 ;* R5 - Contains the bit map representation of the active line.
4024 ;* Carry set indicates success.
4025 ;*
4026 ;* CALLING SEQUENCE: JSR PC,FINACT
4027 ;*
4028 ;* COMMENTS:
4029 ;*
4030 ;* SUBORDINATE ROUTINES CALLED: NONE.
4031 ;* *****
4032
4033 030772 FINACT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
030772 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4034 ;*
4035 ;* Find an active line on which to perform the test.
4036 ;*
4037 030776 005001 CLR R1 ;CLEAR THE LINE NUMBER COUNTER.
4038 031000 012703 000010 MO' #NUMLNS,R3 ;GET MAX LINE NUMBER.
4039 031004 013700 002272 MOV ACTLNS,R0 ;GET THE ACTIVE LINE BIT MAP.
4040 031010 012705 000001 MOV #1,R5 ;SET UP A LINE BIT MASK.
4041 031014 030500 2$: BIT R5,R0 ;LOOK FOR AN ACTIVE LINE.
4042 031016 001006 BNE 4$ ;BRANCH TO BEGIN TEST IF A LINE HAS BEEN FOUND.
4043 031020 006305 ASL R5 ;SHIFT THE BIT MASK FOR THE NEXT LINE.
4044 031022 005201 INC R1 ;INCREMENT THE LINE NUMBER COUNTER.
4045 031024 020103 CMP R1,R3 ;CHECK IF ALL LINES HAVE BEEN TRIED.
4046 031026 002772 BLT 2$ ;LOOP TO TRY THE NEXT LINE.
4047 031030 000241 CLC ;CLEAR CARRY BIT, NO ACTIVE LINE FOUND.
4048 031032 000401 BR 60$ ;EXIT WITH FAILURE.
4049 031034 000261 4$: SEC ;SET CARRY, SUCCESS.
4050
4051 031036 60$: PASS R1,R5 ;RESTORE GPRS, EXCEPT
031036 010166 000004 MOV R1,R1SLOT(SP) ;PUT R1 IN STACK SLOT.
031042 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
031046 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4052 ;R1 CONTAINS THE NUMBER OF FIRST ACTIVE LINE.
4053 ;R5 - CONTAINS THE BIT MAP OF THE ACTIVE LINE.
4054 ;CARRY - SET INDICATES SUCCESS.
4055 031050 000207 RTS PC

```


GLOBAL SUBROUTINE

- GETCHR -

```

4057 .SBTTL GLOBAL SUBROUTINE GETCHR -
4058 ;* *****
4059 ;* - Get a Character From the RX Buffer Routine -
4060 ;* This subroutine gets a character from the RX buffer which is in the
4061 ;* host system memory. If the buffer is empty upon entry of this routine
4062 ;* this routine returns a null character with DATA.VALID clear and a
4063 ;* buffer empty indication.
4064 ;*
4065 ;* INPUTS: RXBCNT - RX buffer character count.
4066 ;*          RXBEND - Label after end of the RX buffer area in memory.
4067 ;*          RXBETX - Equated to RX buffer level at which to enable TX.
4068 ;*          RXBOPT - Pointer to next available input slot of RX buffer.
4069 ;*          RXBSTA - Label at start of RX buffer area in memory.
4070 ;*
4071 ;* OUTPUTS: R2 - Character which is read from the buffer.
4072 ;*          RXBOPT - Updated to point to next input slot of RX buffer.
4073 ;*          RXBCNT - RX buffer character count (Updated).
4074 ;*          CARRY "Success" flag (Set if buffer is not empty on entry).
4075 ;*
4076 ;* CALLING SEQUENCE: JSR PC,GETCHR
4077 ;*
4078 ;* COMMENTS:
4079 ;*
4080 ;* SUBORDINATE ROUTINES CALLED: None.
4081 ;* *****
4082
4083 GETCHR:: SAVE
4084 031052 004537 005474 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
4085 031056 005000 CLR R0 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4086 031060 005002 CLR R2 ;CLEAR THE "RE-ENABLE" TX FLAG (SUBRTN OUTPUT).
4087 031062 005737 003066 TST RXBCNT ;GET NULL CHAR IN CASE BUFFER IS EMPTY.
4088 031070 013704 003062 BEQ 60$ ;CHECK FOR RX BUFFER EMPTY, CLEAR CARRY.
4089 031074 011402 MOV RXBOPT,R4 ;EXIT THE ROUTINE IF BUFFER IS EMPTY.
4090 031076 005024 MOV (R4),R2 ;GET THE BUFFER OUTPUT POINTER.
4091 031100 020427 003270 CLR (R4)+ ;GET A CHARACTER FROM THE BUFFER.
4092 031104 103402 CMP R4,RXBEND ;DELETE THE READ CHARACTER FROM THE BUFFER.
4093 031106 012704 003070 BLO 2$ ;CHECK IF POINTER SHOULD WRAP AROUND.
4094 031112 010437 003062 MOV RXBSTA,R4 ;SKIP WRAPAROUND IF POINTER IS NOT AT END.
4095 2$ MOV R4,RXBOPT ;WRAP INPUT POINTER AROUND.
4096 031116 005337 003066 DEC RXBCNT ;UPDATE THE OUTPUT POINTER STORAGE.
4097 031122 000261 SEC ;REMOVE THIS CHAR FROM THE BUFFER COUNT.
4098 4099 031124 010266 000006 60$: PASS R2 ;SET SUCCESS FLAG, BUFFER WAS NOT EMPTY.
4099 031124 004736 MOV R2,R2SLOT(SP) ;RESTORE GPRS, EXCEPT
4100 031130 000207 JSR PC,0(SP)+ ;PUT R2 IN STACK SLOT.
4101 ;R2 CONTAINS THE CHARACTER READ FROM BUFFER.
4102 RTS PC ;RETURN TO PREG05 SUBRT.
;CARRY "SUCCESS" FLAG, SET IF BUFFER NOT EMPTY.

```

GLOBAL SUBROUTINE

- GETLP2 -

```

4104 .SBTTL GLOBAL SUBROUTINE - GETLP2 -
4105 ;* *****
4106 ;* - Get Line Parameters Routine Number Two -
4107 ;* This routine is used to repeatedly get combinations of line parameter
4108 ;* contents for the Single Character Mode TX/RX Test (long data pattern).
4109 ;* Each time this routine is called it gets another combination of the
4110 ;* paramters in the paramter tables until all combinations have been
4111 ;* returned at which point it returns a "failure" indication.
4112 ;*
4113 ;* INPUTS: Single character mode, short data pattern TX/RX tables:
4114 ;*          SCBCT - Number of bits per char table (4 entries).
4115 ;*          SCNST - Number of stop bits bits table (2 entries).
4116 ;*          SCTPT - Type of parity table (3 entries).
4117 ;*          Each table has a base and end label consisting of the name of
4118 ;*          the table with a "B" and "E" appended respectively.
4119 ;*          R1 thru R3 - Pointers into SCBCT, SCNST, SCTPT tables
4120 ;*          R1 is clear if this is the first call of GETLP2.
4121 ;*
4122 ;* OUTPUTS: R0 Composed LPR contents, clear if failure (Done),
4123 ;*          38.4K baudrate is selected.
4124 ;*          R1 thru R3 - Table pointers (Updated).
4125 ;*
4126 ;* CALLING SEQUENCE: JSR PC,GETLP2
4127 ;*
4128 ;* COMMENTS: This routine should be used in congunction with a SWAPx
4129 ;*          routine to avoid destroying the GPR contents.
4130 ;*
4131 ;* SUBORDINATE ROUTINES CALLED: None.
4132 ;* *****
4133
4134 GETLP2:: SAVE
4135         TST R1
4136         BNE 2$
4137         MOV #SCBCTB,R1
4138         MOV #SCNSTB,R2
4139         MOV #SCTPTB,R3
4140
4141 2$: CMP R3,#SCTPTE
4142     BLO 4$
4143     MOV #SCTPTB,R3
4144     TST (R2)+
4145     CMP R2,#SCNSTE
4146     BLO 4$
4147     MOV #SCNSTB,R2
4148     TST (R1)+
4149     CMP R1,#SCBCTE
4150     BLO 4$
4151     CLR R0
4152     CLC
4153     BR 60$
4154
4155 4$: MOV #177400,R0
4156     BIS (R1),R0
4157     BIS (R2),R0
4158     BIS (R3)+,R0
4159
;SAVE CONTENTS OF GPRS R0 THRU R5.
;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
;TEST FOR THIS BEING FIRST CALL OF GETLP2.
;SKIP ORIGINAL SET UP IF NOT FIRST CALL.
;INITIALIZE BITS PER CHAR TABLE POINTER.
;INITIALIZE # OF STOP BITS TABLE POINTER.
;INITIALIZE TYPE OF PARITY TABLE POINTER.

;CHECK FOR POINTER AT END OF TABLE.
;GO GET LPR CONTENTS IF NOT AT END OF TABLE.
;RESET POINTER TO BEGINNING OF TABLE.
;INC THE # OF STOP BITS TABLE POINTER BY 2.
;CHECK FOR POINTER AT END OF TABLE.
;GO GET LPR CONTENTS IF NOT AT END OF TABLE.
;RESET POINTER TO BEGINNING OF TABLE.
;INC BAUD RATES TABLE POINTER BY 2.
;CHECK FOR POINTER AT END OF TABLE.
;GO GET LPR CONTENTS IF NOT AT END OF TABLE.
;PREPARE TO PASS OUT CLEAR LPR FIELDS.
;INDICATE "FAILURE" FOR EXIT.
;EXIT WITH "FAILURE", WE'RE DONE.

;SET BAUD RATE FIELDS FOR 38.4 K BAUD.
;GET THE BITS/CHAR FIELD OF NEW LPR CONTENTS.
;INCLUDE THE NUMBER OF STOP BITS FIELD.
;INCLUDE THE TYPE OF PARITY FIELD.

```


GLOBAL SUBROUTINE

- GETLP2 -

```
4160 031236 000261          SEC          ;INDICATE "SUCCESS" FOR EXIT.
4161
4162 031240          60$: PASS  R0,R1,R2,R3 ;RESTORE GPRS R4 & R5, LEAVE FOLLOWING INTACT:
      031240 010066 000002          MOV  R0,R0SLOT(SP) ;PUT R0 IN STACK SLOT.
      031244 010166 000004          MOV  R1,R1SLOT(SP) ;PUT R1 IN STACK SLOT.
      031250 010266 000006          MOV  R2,R2SLOT(SP) ;PUT R2 IN STACK SLOT.
      031254 010366 000010          MOV  R3,R3SLOT(SP) ;PUT R3 IN STACK SLOT.
      031260 004736          JSR    PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4163 031262 000207          RTS    PC          ; R1 THRU R3 - POINTERS, R0 - NEW LPR FIELDS.
```

GLOBAL SUBROUTINE

- GETTIM -

```

4165 .SBTTL GLOBAL SUBROUTINE - GETTIM -
4166 ;* *****
4167 ;* Get Time-out Value Based on Minimum Baudrate Routine -
4168 ;* This subroutine gets the necessary time-out value to verify that all
4169 ;* chars have been received at the completion of the TX/RX of a data
4170 ;* pattern. This uses the slowest baudrate which is specified in the
4171 ;* passed in DUT LPR contents to calculate this time-out value.
4172 ;*
4173 ;* INPUTS: R1 - DUT LPR contents.
4174 ;*
4175 ;* OUTPUTS: RXTOUT - Time-out value for waiting for last RX char.
4176 ;*
4177 ;* CALLING SEQUENCE: JSR PC,GETTIM
4178 ;*
4179 ;* COMMENTS:
4180 ;*
4181 ;* SUBORDINATE ROUTINES CALLED: None.
4182 ;* -- *****
4183
4184 031264 GETTIM:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4185 031264 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4186 031270 000301 SWAB R1 ;PUT THE BAUD RATE FIELDS IN THE LOW BYTE.
4187 031272 042701 177400 BIC #177400,R1 ;CLEAR STOP,PARITY,AND CHAR FIELDS.
4188 031276 010102 MOV R1,R2 ;COPY BAUD RATE FIELDS.
4189 031300 042701 000360 BIC #360,R1 ;SELECT RX BAUD RATE FIELD ONLY.
4190 031306 006202 ASR R2 ;SHIFT TX BAUD RATE FIELD
4191 031310 006202 ASR R2 ; TO OCCUPY THE LOW FOUR BYTES.
4192 031312 006202 ASR R2 ;
4193 031314 020102 CMP R1,R2 ;CHECK IF SAME BAUD RATE IN EACH FIELD.
4194 031316 101401 BLOS 2$ ;BRANCH IF RX BAUD RATE IS LOWER OR SAME.
4195 031320 010201 MOV R2,R1 ;TX BAUD RATE IS THE SLOWER OF THE TWO.
4196 031322 116102 005344 2$: MOVB PROTB(R1),R2 ;GET PROPORTIONAL DELAY FROM TABLE.
4197 031326 042702 177400 BIC #177400,R2 ;CLEAR UPPER BYTE BECAUSE OF SIGN EXTENSION.
4198 031332 010237 002402 MOV R2,RXTOUT ;LOAD THE RX TIME-OUT VARIABLE.
4199
4200 031336 60$: PASS ;RESTORE GPRS.
4201 031336 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4201 031340 000207 RTS PC

```

GLOBAL SUBROUTINE

INDATP -

```

4203      .SBTTL GLOBAL SUBROUTINE          - INDATP -
4204      ;* *****
4205      ;* - INITIALISE DATA PATTERN -
4206      ;* This subroutine is used to initialise an incremental byte data pattern
4207      ;* in the general buffer area.
4208      ;* The data pattern will be sequential from 0 to 255 (decimal).
4209      ;*
4210      ;* INPUTS:      BUFBAS - Address of the start of the general buffer area.
4211      ;*              BUFMID - Address of the 255 th location.
4212      ;*
4213      ;* OUTPUTS:     The first 255 locations of the general buffer area contain data
4214      ;*
4215      ;* CALLING SEQUENCE:  JSR      PC,INDATP
4216      ;*
4217      ;* COMMENTS:
4218      ;*
4219      ;* SUBORDINATE ROUTINES CALLED: NONE.
4220      ;* *****
4221
4222 031342 INDATP:: SAVE                      ;SAVE CONTENTS OF GPRS R0 THRU R5.
031342 004537 005474                      JSR      R5,PREG05      ;CALL REGISTER SAVE SUBRT.
4223
4224 031346 012702 004012      MOV      #BUFBAS,R2      ;INITIALIZE THE DATA PATTERN IN THE GENERAL
4225 031352 005003            CLR      R3              ; DATA BUFFER TO A 256 BYTE PATTERN.
4226 031354 110322      2$:      MOVB   R3,(R2)+        ;
4227 031356 005203            INC      R3              ;SELECT THE NEXT CHARACTER.
4228 031360 020227 004412      CMP     R2,#BUFMID      ;CHECK IF WE HAVE 256 DATA PATTERNS.
4229 031364 103773            BLO     2$              ;
4230
4231 031366      60$:      PASS                      ;RESTORE GPRS.
031366 004736                      JSR      PC,@(SP)+    ;RETURN TO PREG05 SUBRT.
4232 031370 000207            RTS     PC

```

GLOBAL SUBROUTINE

- INDTPX -

SEQ 0113

```

4234 .SBTTL GLOBAL SUBROUTINE - INDTPX -
4235 ;* *****
4236 ;* - INITIALISE DATA PATTERN WITHOUT XON OR XOFF -
4237 ;* This subroutine is used to initialise an incremental byte data pattern
4238 ;* in the general buffer area.
4239 ;* The data pattern will be from 0 to 255, but will exclude the following
4240 ;* two characters; (ASCII DC1, DC3) XON AND XOFF. This will cause the
4241 ;* last two data characters to be the same as the first two.
4242 ;*
4243 ;* INPUTS:      BUFBAS - Address of the start of the general buffer area.
4244 ;*             BUF MID - Address of the 255 th location.
4245 ;*
4246 ;* OUTPUTS:     The first 255 locations of the general buffer area contain data
4247 ;*
4248 ;* CALLING SEQUENCE: JSR PC,INDTPX
4249 ;*
4250 ;* COMMENTS:
4251 ;*
4252 ;* SUBORDINATE ROUTINES CALLED: NONE.
4253 ;* *****
4254
4255 031372 004537 005474 INDTPX:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4256 ;* JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4257 ;*
4258 ;* Initialize the 256 byte data pattern.
4259 ;* Ensure the data pattern is free from XON's or XOFF's to prevent errors.
4260 ;* Note: the first two characters and the last two characters will be the same.
4261 ;*
4261 031376 012702 004012 MOV #BUFBAS,R2 ;INITIALIZE THE DATA PATTERN IN THE GENERAL
4262 031402 005003 CLR R3 ; DATA BUFFER TO A 256 BYTE PATTERN.
4263 031404 110322 2$: MOV R3,(R2)+ ;
4264 031406 105203 INCB R3 ;SELECT THE NEXT CHARACTER.
4265 031410 122703 000021 CMPB #21,R3 ;CHECK FOR AN XON CHARACTER.
4266 031414 001001 BNE 4$ ;BRANCH IF CHAR NOT AN XON.
4267 031416 105203 INCB R3 ;FORCE THE NEXT CHARACTER.
4268 031420 122703 000023 4$: CMPB #23,R3 ;CHECK FOR AN XOFF CHARACTER.
4269 031424 001001 BNE 6$ ;BRANCH IF NOT AN XOFF CHARACTER.
4270 031426 105203 INCB R3 ;FORCE THE NEXT CHARACTER.
4271 031430 020227 004412 6$: CMP R2,#BUFMID ;CHECK IF WE HAVE 256 DATA PATTERNS.
4272 031434 103763 BLO 2$ ;
4273
4274 031436 004736 60$: PASS ;RESTORE GPRS.
4275 031440 000207 RTS PC JSR PC,#(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- INICHR -

SEQ 0114

```

4277 .SBTTL GLOBAL SUBROUTINE INICHR -
4278 ;* *****
4279 ;* - Send Initial Characters Routine -
4280 ;* This routine is used to initiate single character transmission.
4281 ;* This routine sends the initial characters to each active line to
4282 ;* cause future TX interrupts which will continue the transmission if
4283 ;* more than one character is to be sent to each active line.
4284 ;*
4285 ;* INPUTS: ACTLNS - Bit map of active DUT lines.
4286 ;* BITTBL - Label of table of words each with a bit set.
4287 ;* CSRA - Contains the address of the DUT CSR.
4288 ;* DPENDB - Base of the data pattern end table (entry per line).
4289 ;* DPLENB - Base of the data pattern length table.
4290 ;* IBM - Bit mask of inactive TX/RX bits.
4291 ;* IESTAT - States of DUT int enable bits (Other bits clear).
4292 ;* NUMLNS - Equated to the number of lines on the DUT.
4293 ;* TXCHRA - Contains the address of the DUT TXCHAR register.
4294 ;* TXCNTB - Label at base of the TX character counter table.
4295 ;* TXPTRB - Label at base of the TX data pattern pointers table.
4296 ;*
4297 ;* OUTPUTS: CSR - DUT CSR IND.ADR.REG field is destroyed.
4298 ;* TXCHAR - DUT TXCHAR has word written to it.
4299 ;* TXCNTx - Counters incremented for lines on which chars sent.
4300 ;* TXPTRB - Each pointer in table points to next TX char for line.
4301 ;*
4302 ;* CALLING SEQUENCE: JSR PC,INICHR
4303 ;*
4304 ;* COMMENTS: This routine assumes that at least one character should be
4305 ;* transmitted on each active line.
4306 ;* Interrupts must be disabled when calling this routine.
4307 ;*
4308 ;* SUBORDINATE ROUTINES CALLED: None.
4309 ;* - *****
4310 INICHR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4311 031442 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4312 031446 013701 002374 MOV IESTAT,R1 ;GET STATE OF TX.IE, RX.IE FOR USE IN SETTING
4313 031452 005002 CLR R2 ;SET LINE NUMBER OFFSET TO LINE 0.
4314 031454 036237 002534 002272 2$: BIT BITTBL(R2),ACTLNS ;TEST THE ACTIVE LINES BIT FOR THIS LINE.
4315 031462 001424 BEQ 6$ ;DON'T TX ON THIS LINE IF IT IS NOT ACTIVE.
4316 031464 010177 150610 MOV R1,@CSRA ;SET UP THE IND.ADR.REG FIELD OF THE CSR.
4317 031470 016205 003512 MOV TXPTRB(R2),R5 ;GET THE TX DATA PATTERN POINTER FOR THIS LINE.
4318 031474 112504 MOV R5,R4 ;GET THE CHAR TO TX ON THIS LINE, INC POINTER.
4319 031476 020562 003312 CMVB R5,DPENDB(R2) ;COMPARE POINTER WITH DATA PATTERN END ADR.
4320 031502 103402 BLO 4$ ;SKIP POINTER WRAPAROUND IF NOT AT PATTERN END.
4321 031504 166205 003352 SUB DPLENB(R2),R5 ;WRAP TX POINTER AROUND TO BEGINNING OF PAT'N.
4322 031510 010562 003512 4$: MOV R5,TPTRB(R2) ;UPDATE THE TX POINTER STORAGE TABLE FOR LINE.
4323 031514 043704 002366 BIC IBM,R4 ;CLEAR INACTIVE BITS OF TX CHARACTER WORD.
4324 031520 052704 100000 BIS #BIT15,R4 ;SET THE TX.DATA.VALID BIT IN THE WORD.
4325 031524 010477 150552 MOV R4,@TXCHA ;TX THE FIRST CHARACTER FOR THIS LINE.
4326 031530 005262 003652 INC TXCNTB(R2) ;INCREMENT TX CHARACTER COUNTER FOR THIS LINE.
4327 031534 005201 6$: INC R1 ;INCREMENT WORD FOR IND.ADR.REG FIELD SET UP.
4328 031536 062702 000002 ADD #2,R2 ;SET LINE NUMBER OFFSET TO NEXT LINE.
4329 031542 020227 000020 CMP R2,#NUMLNS*2 ;COMPARE LINE OFFSET WITH TWICE THE # OF LINES.
4330 031546 002742 BLT 2$ ;LOOP TO SEND CHAR TO ANOTHER LINE IF NOT DONE.
4331
4332 031550 60$: PASS ;RESTORE GPRS.

```

GLOBAL SUBROUTINE

INICHR -

SEQ 0115

031550 004736
4333 031552 000207

RTS PC JSR PC,8(SP)+

;RETURN TO PREG05 SUBRT.

- INIDMA -

4335			
4336			
4337			
4338			
4339			
4340			
4341			
4342			
4343			
4344			
4345			
4346			
4347			
4348			
4349			
4350			
4351			
4352			
4353			
4354			
4355			
4356			
4357			
4358			
4359			
4360			
4361			
4362			
4363			
4364			
4365			
4366	031554		
	031554	004537	005474
4367	031560	013705	002272
4368			
4369			
4370			
4371	031564	005001	
4372			
4373			
4374			
4375			
4376	031566	010104	
4377	031570	006304	
4378	031572	016402	002534
4379	031576	030205	
4380	031600	001414	
4381			
4382			
4383			
4384			
4385			
4386	031602	016403	003352
4387	031606	016402	003512
4388			
4389			
4390			

[illegible]

GLOBAL SUBROUTINE

- INIDMA -

```

4391 031612 004737 030624      JSR    PC,DODMA
4392 031616 103403      BCS    6$      ;SKIP ERROR IF DODMA WAS SUCCESSFUL.
4393                          ;+
4394                          ; Set the proper bit of the TX interrupt flags to indicate the line error.
4395                          ;-
4396 031620 050537 002420      BIS    R5,TXINTF      ;INDICATE THE ERROR.
4397 031624 000402      BR      10$      ;SKIP UPDATING POINTERS AND COUNTERS.
4398                          ;+
4399                          ; Update the TX character count for this line.
4400                          ;-
4401 031626 060364 003652      6$:    ADD    R3,TXCNTB(R4)  ;ADD THE DATA PATTERN LENGTH TO TX CHAR COUNT.
4402                          ;+
4403                          ; Increment line counter,goto next line if not done.
4404                          ;-
4405 031632 005201      10$:    INC    R1      ;INCREMENT THE LINE COUNTER.
4406 031634 020127 000010      CMP    R1,#NUMLNS      ;COMPARE THE LINE COUNTER WITH NUMBER OF LINES.
4407 031640 002752      BLT      2$      ;LOOP TO SEND CHAR TO ANOTHER LINE IF NOT DONE.
4408                          ;-
4409 031642      60$:    PASS      ;RESTORE GPRS.
      031642 004736      JSR    PC,@(SP)+      ;RETURN TO PREG05 SUBRT.
4410 031644 000207      RTS     PC

```


GLOBAL SUBROUTINE

LINBIT

```

4412 .SBTTL GLOBAL SUBROUTINE LINBIT -
4413 ;... *****
4414 ;* Line Number to Bit Map conversion subroutine -
4415 ;* This subroutine is used to generate a bit map (one bit of 16 set)
4416 ;* based on a line number (range: 1 to 16). Only the LS 4 bits of the
4417 ;* line number word are used, the others are masked out (so unmasked
4418 ;* MSBytes of DUT CSRs can be passed to this routine without error).
4419 ;*
4420 ;* INPUTS: R1 - Line number (only LS 4 bits used, others disregarded).
4421 ;* BITTBL - Base label of a 16 word bit table.
4422 ;*
4423 ;* OUTPUTS: R0 - Bit map, bit corresponding to line number is set:
4424 ;* If line number is 3, then bit3 is set, etc.
4425 ;*
4426 ;* CALLING SEQUENCE: JSR PC,LINBIT
4427 ;*
4428 ;* COMMENTS: No checking is performed to verify that the line number is
4429 ;* a legal line number for the DUT (i.e. - less than NUMLNS).
4430 ;* NOTE: The line number is not destroyed or altered, so this
4431 ;* routine can be used easily in loops.
4432 ;*
4433 ;* SUBORDINATE ROUTINES CALLED: None.
4434 ;* *****
4435
4436 031646 LINBIT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4437 031646 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4438 031652 042701 177760 BIC #177760,R1 ;MASK OUT ALL BUT 4 LSBITS OF THE LINE #.
4439 031656 006301 ASL R1 ;MULTIPLY LINE # BY 2 TO GET WORD TABLE OFFSET.
4440 031660 016100 002534 MOV BITTBL(R1),R0 ;GET THE SINGLE BIT BIT MAP.
4441 031664 010066 000002 60$: PASS R0 ;RESTORE GPRS, EXCEPT THE FOLLOWING,
4442 031670 004736 MOV R0,ROSLOT(SP) ;PUT R0 IN STACK SLOT.
4443 031672 000207 JSR PC,@(SP) ;RETURN TO PREG05 SUBRT.
4444 ;R0 - BIT MAP WITH LINE # BIT SET.

```

GLOBAL SUBROUTINE

MAPCNT -

```

4443 .SBTTL GLOBAL SUBROUTINE MAPCNT -
4444 ;* *****
4445 ;* - Count Bits in Bit Map Routine
4446 ;* This subroutine counts the number of bits which are set in a bit map.
4447 ;*
4448 ;* INPUTS: R2 - The bit map for which to count the bits.
4449 ;*
4450 ;* OUTPUTS: R2 - Count of the number of b'its that were set.
4451 ;*
4452 ;* CALLING SEQUENCE: JSR PC,MAPCNT
4453 ;*
4454 ;* COMMENTS:
4455 ;*
4456 ;* SUBORDINATE ROUTINES CALLED: None.
4457 ;* *****
4458
4459 031674 004537 005474 MAPCNT:: SAVE
031674 010201 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
4460 031700 010201 R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4461 031702 001405 MOV R2,R1
4462 BEQ 60$ ;EXIT WITH ZERO IF NO BITS ARE SET IN MAP.
4463 031704 005002 CLR R2
4464 031706 000261 SEC ;CLEAR THE BIT COUNT.
4465 ;COUNT THE LAST BIT TO BE SHIFTED OUT.
4466 031710 005502 2$: ADC R2
4467 031712 006301 ASL R1 ;COUNT THE BIT IF IT WAS SET.
4468 031714 001375 BNE 2$ ;SHIFT ANOTHER BIT OUT OF THE MAP.
4469 ;LOOP IF ALL BITS NOT SHIFTED OUT OF MAP.
4470 031716 010266 000006 60$: PASS R2
031716 004736 MOV ;RESTORE GPRS, EXCEPT THE FOLLOWING:
031722 004736 JSR R2,R2SLOT(SP) ;PUT R2 IN STACK SLOT.
4471 031724 000207 RTS PC PC @ (SP) ;RETURN TO PREG05 SUBRT.
; '2' COUNT OF BITS SET IN BIT MAP.

```

GLOBAL SUBROUTINE

MSLGET -

```

4473 .SBTTL GLOBAL SUBROUTINE MSLGET
4474 *****
4475 ;* - Milli Seconds Loop which returns read word and remaining time -
4476 ;* This subroutine is a general purpose test loop subroutine. It is used
4477 ;* to verify that a certain action occurs before a time-out period. The
4478 ;* calling routine passes in which bits should be set and cleared for the
4479 ;* desired condition and the time-out value in milli seconds.
4480 ;* This routine checks for the desired condition upon entrance into the
4481 ;* routine and then once each milli-second thereafter.
4482 ;* Upon return, the last word which was read to check for the condition
4483 ;* is returned by this subroutine.
4484 ;*
4485 ;* INPUTS: R1 - Time-out value in milli-seconds (up to 64K ms).
4486 ;* R2 - Bit map of bits to test (1 indicates to test the bit).
4487 ;* R3 - Desired states of the indicated fields in R2.
4488 ;* R4 - Address of the word to test.
4489 ;* MSLCNT - Milli second software loop count.
4490 ;*
4491 ;* OUTPUTS: R0 - The last word which was read to check for the condition.
4492 ;* R1 - Remaining number of ms in time out time.
4493 ;* CARRY - Success flag (set if condition is met before time-out).
4494 ;*
4495 ;* CALLING SEQUENCE: JSR PC,MSLGET
4496 ;*
4497 ;* COMMENTS: This routine works with or without a hardware clock, but the
4498 ;* calibration is only guaranteed when a line clock is available
4499 ;* on the system.
4500 ;* This routine can be used as a delay routine, by specifying the
4501 ;* desired delay as the time out and specifying a condition to
4502 ;* look for which will not be met during the delay.
4503 ;* If a time-out value of 0 is specified, this routine checks for
4504 ;* the desired condition before returning. It indicates success
4505 ;* if the condition is met, failure otherwise.
4506 ;*
4507 ;*
4508 ;* SUBORDINATE ROUTINES CALLED: None.
4509 ;* *****
4510
4511 031726 MSLGET:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4512 031726 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4513 ;*
4514 ; Set up mask for removing unused bits in the test word, and clear unused
4515 ; bits in the desired state word to allow direct comparison.
4516 ;
4517 031732 005102 COM R2 ;GET MASK OF UNUSED BITS.
4518 031734 040203 BIC R2,R3 ;MASK OUT UNUSED BITS IN DESIRED STATE WORD.
4519 ;*
4520 ; Handle the test and exit if we have a 0 time-out value.
4521 ;
4522 031736 005701 TST R1 ;TEST THE TIME OUT VALUE FOR ZERO.
4523 031740 001011 BNE 2$ ;IF NON ZERO TIME-OUT, GO LOOP AND TEST.
4524 031742 011400 MOV (R4),R0 ;GET THE WORD TO TEST BEFORE EXITING.
4525 031744 010037 032040 MOV R0,62$ ;SAVE VALUE SO WE CAN RETURN IT.
4526 031750 040200 BIC R2,R0 ;MASK OUT UNTESTED BITS OF WORD.
4527 031752 020003 CMP R0,R3 ;COMPARE AGAINST DESIRED STATE WORD.
4528 031754 000261 SEC ;INDICATE SUCCESS IN CASE WORDS ARE EQUAL.
4529 031756 001420 BEQ 6$ ;EXIT WITH SUCCESS IF WORDS ARE EQUAL.

```

GLOBAL SUBROUTINE

- MSLGET

```

4529 031760 000241          CLC          ;INDICATE FAILURE (TIME-OUT).
4530 031762 000416          BR          6$ ;EXIT WITH FAILURE, WORDS AREN'T EQUAL.
4531                          ;*
4532                          ; Non-zero t me out value. Loop, waiting for condition or time-out.
4533                          ;-
4534 031764 013705 002462    2$: MOV      MSLCNT,R5      ;LOAD MS LOOP COUNT.
4535 031770 011400          4$: MOV      (R4),R0        ;GET THE WORD TO TEST.
4536 031772 010037 032040    MOV      R0,62$          ;SAVE WORD IN CASE THIS IS THE LAST.
4537 031776 040200          BIC      R2,R0             ;MASK OUT UNTESTED BITS OF WORD.
4538 032000 020003          CMP      R0,R3             ;COMPARE AGAINST DESIRED STATE WORD.
4539 032002 000261          SEC                      ;SET CARRY IN CASE OF SUCCESS.
4540 032004 001405          BEQ      6$                ;EXIT WITH SUCCESS IF WORDS ARE EQUAL.
4541 032006 005305          DEC      R5                ;COUNT DOWN THE INSIDE MS LOOP COUNT.
4542 032010 001367          BNE      4$                ;LOOP IF MS NOT UP.
4543 032012 005301          DEC      R1                ;DECREMENT THE MS TIME COUNT.
4544 032014 001363          BNE      2$                ;IF TIME NOT UP, LOOP TO COUNT ANOTHER MS.
4545 032016 000241          CLC                      ;CLEAR CARRY, WE TIMED-OUT.
4546                          ;*
4547                          ; Have either found condition, or timed out (possibly from 0 time-out value).
4548                          ; Restore the last contents read from the test word. Exit routine.
4549                          ;-
4550 032020 013700 032040    6$: MOV      62$,R0        ;PASS OUT THE LAST READ WORD.
4551 032024          60$: PASS      R0,R1              ;RESTORE GPRS, EXCEPT THE FOLLOWING:
                                                MOV      R0,ROSLOT(SP) ;PUT R0 IN STACK SLOT.
                                                MOV      R1,R1SLOT(SP)  ;PUT R1 IN STACK SLOT.
                                                JSR      PC,@(SP)    ;RETURN TO PREG05 SUBRT.
                                                ;R0 - LAST READ WORD CHECKED FOR CONDITION.
                                                ;R1 - REMAINING TIME (0 IF TIME OUT OCCURED).
                                                ;CARRY - SET IF SUCCESS, CLEAR IF TIME OUT.
4552                          ;
4553                          ; Local storage.
4554 032036 000207          RTS      PC
4555                          ;
4556                          ;
4557                          ;
4558 032040 000000          62$: .WORD 0              ;STORAGE FOR THE LAST READ WORD.

```

GLOBAL SUBROUTINE

- MSLOOP -

```

4560 .SBTTL GLOBAL SUBROUTINE - MSLOOP -
4561 ;*****
4562 ;* - Test Loop subroutine -
4563 ;* This subroutine is a general purpose test loop subroutine. It is used
4564 ;* to verify that a certain action occurs before a time out period. The
4565 ;* calling routine passes in which bits should be set and cleared for the
4566 ;* desired condition and the time-out value in milli-seconds.
4567 ;* This routine checks for the desired condition upon entrance into the
4568 ;* routine and then once each milli-second thereafter.
4569 ;*
4570 ;* INPUTS: R1 - Time-out value in milli-seconds (up to 64K ms).
4571 ;* R2 - Bit map of bits to test (1 indicates to test the bit).
4572 ;* R3 - Desired states of the indicated fields in R2.
4573 ;* R4 - Address of the word to test.
4574 ;* MSLCNT - Milli second software loop count.
4575 ;*
4576 ;* OUTPUTS: CARRY - Success flag (set if condition is met before time-out).
4577 ;*
4578 ;* CALLING SEQUENCE: JSR PC,MSLOOP
4579 ;*
4580 ;* COMMENTS: This routine works with or without a hardware clock, but the
4581 ;* calibration is only guaranteed when a line clock is available
4582 ;* on the system.
4583 ;* This routine can be used as a delay routine, by specifying the
4584 ;* desired delay as the time-out and specifying a condition to
4585 ;* look for which will not be met during the delay.
4586 ;* If a time-out value of 0 is specified, this routine checks for
4587 ;* the desired condition before returning. It indicates success
4588 ;* if the condition is met, failure otherwise.
4589 ;*
4590 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
4591 ;*****
4592
4593 032042 MSLOOP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
032042 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4594
4595 ;*
4596 ; Calling the MSLGET routine from the MSLOOP routine isolates the caller of
4597 ; MSLOOP from the returned test word and remaining time-out values.
4598 ; -
4599 032046 004737 031726 JSR PC,MSLGET ;CALL THE MULTI-PURPOSE MS LOOP AND SEARCH RTN.
4600
4601 032052 60$: PASS ;RESTORE GPRS,
032052 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4602 032054 000207 RTS PC ;CARRY - SET IF SUCCESS, CLEAR IF TIME OUT.

```

GLOBAL SUBROUTINE

MUL16U

```

4604 .SBTTL GLOBAL SUBROUTINE - MUL16U -
4605 ;* *****
4606 ;* - 16 Bit Unsigned Multiply Routine -
4607 ;* This routine multiplies 2 16 bit unsigned numbers and returns a 16 bit
4608 ;* unsigned result. The multiplication is performed by iterative
4609 ;* addition of one number to a sum while decrementing the other number
4610 ;* to zero. If overflow occurs (177777 to 0) the product is invalid.
4611 ;*
4612 ;* INPUTS: R1 Multiplicand (16 bit unsigned).
4613 ;* R2 Multiplier (16 bit unsigned).
4614 ;*
4615 ;* OUTPUTS: R1 - Product (16 bit unsigned), -1 if overflow.
4616 ;* CARRY - Set if success (no overflow), clear otherwise.
4617 ;*
4618 ;* CALLING SEQUENCE: JSR PC,MUL16U
4619 ;*
4620 ;* COMMENTS: Note: For minimum execution time R2 should contain the
4621 ;* smaller of the 2 arguments.
4622 ;*
4623 ;* SUBORDINATE ROUTINES CALLED: None.
4624 ;* *****
4625
4626 032056 MUL16U:: SAVE JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
032056 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4627 032062 005003 CLR R3 ;CLEAR THE PRODUCT.
4628 032064 005702 TST R2 ;CHECK THE MULTIPLIER.
4629 032066 001003 BNE 2$ ;GO TO DO MULTIPLICATION IF NOT ZERO.
4630 032070 005001 CLR R1 ;RETURN A PRODUCT OF ZERO.
4631 032072 000261 SEC ;INDICATE SUCCESS.
4632 032074 000412 BR 60$ ;EXIT THE ROUTINE.
4633
4634 032076 060103 2$: ADD R1,R3 ;ADD THE MULTIPLICAND TO THE PRODUCT.
4635 032100 103405 BCS 50$ ;EXIT WITH OVERFLOW IF ONE OCCURRED.
4636 032102 005302 DEC R2 ;DECREMENT THE MULTIPLIER.
4637 032104 001374 BNE 2$ ;LOOP IF MULTIPLIER NOT ZERO.
4638 032106 010301 MOV R3,R1 ;PREPARE TO PASS OUT THE PRODUCT.
4639 032110 000261 SEC ;INDICATE SUCCESS.
4640 032112 000403 BR 60$ ;EXIT WITH SUCCESS.
4641
4642 032114 012701 177777 50$: MOV #-1,R1 ;FORCE PRODUCT TO MAX VALUE, WE OVERFLOWED.
4643 032120 000241 CLC ;INDICATE FAILURE.
4644
4645 032122 032122 010166 000004 60$: PASS R1 ;RESTORE GPRS, EXCEPT THE FOLLOWING:
032126 004736 MOV R1,R1SLOT(SP) ;PUT R1 IN STACK SLOT.
JSR PC,0(SP) ;RETURN TO PREG05 SUBRT.
4646 ; R1 - PRODUCT (16 BIT UNSIGNED).
4647 032130 000207 RTS PC ; CARRY - SET IF SUCCESS (NO OVERFLOW).

```


GLOBAL SUBROUTINE

- NEWCHR -

```

4649 .SBTTL GLOBAL SUBROUTINE NEWCHR -
4650 ;* *****
4651 ;* - New Character Handling Routine -
4652 ;* This subroutine handles a new character which has been read from
4653 ;* the DUT. The counters and pointers which are involved with the
4654 ;* character are updated. The character is checked for errors and
4655 ;* any errors which are found are reported.
4656 ;*
4657 ;* INPUTS: R2 - The read character including error flags and line number.
4658 ;* R3 - Mask of the inactive bits in a TX or RX char byte.
4659 ;* ACTLNS - Bit map of active DUT lines.
4660 ;* DPRSQB - Label at data pattern resync queues table base.
4661 ;* TXRXLB - Base of TX/RX line number association table.
4662 ;* BITTBL - Table of words with bits set for use in forming maps.
4663 ;* ERSRFB - "Print error summary for line" flags.
4664 ;* ERRRTBL - Error information (ERRNBR, ERRMSG, ERRRTYP).
4665 ;* ERCNTB - Base of the RX character error counters table.
4666 ;* NDERPT - Contains number of char errors to report on a line.
4667 ;* INPUTS TO SUBROUTINES: CHCNTB, DPENDB, DPLEN, DPRSQE, EXCNTB, RXCNTB,
4668 ;* RXPTRB, ERRNBR, ERRMSG, ERRRTYP.
4669 ;*
4670 ;* OUTPUTS: ERRBLK - Contents destroyed.
4671 ;* Following variables updated for line on which char was received:
4672 ;* DPRSQ - Data pattern resync que of received characters.
4673 ;* ERCNT - Count of the number of character errors on line.
4674 ;* ERSRFB - Updated "print error summary for line" flags.
4675 ;* EXCNT - Count of the number of extra chars received on line.
4676 ;* RXCNT - Count of the number of characters received on line.
4677 ;* RXPTR - Updated to point to the next expected char on line.
4678 ;*
4679 ;* CALLING SEQUENCE: JSR PC,NEWCHR
4680 ;*
4681 ;* COMMENTS: This routine can report errors with numbers Initial ERRNBR
4682 ;* and Initial ERRNBR + 1. ERRNBR is restored to its initial
4683 ;* value before this routine returns.
4684 ;*
4685 ;* SUBROUTINES CALLED: CKCHR,CKINAC,TXROFF,TXRON.
4686 ;* INDIRECT SUBROUTINES: CHKEXT,CHKLOS,ER9002,ER9003,UPDCHR.
4687 ;* *****
4688
4689 032132 NEWCHR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4690 032132 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4691 032136 010305 ;MOV R3,R5 ;GET THE BIT MAP OF INACTIVE DATA BYTE BITS.
4692 032140 052705 177400 ;BIS #177400,R5 ;ALL UPPER BITS OF EXPECTED DATA ARE INACTIVE.
4693 032144 005037 032410 ;CLR 70$ ;CLEAR THE "ERROR FOUND" FLAG.
4694 ;*
4695 ;* If the new character is valid on an inactive line, go report error.
4696 ;* Routine used also extracts line number from the new character.
4697 032150 004737 030004 ;JSR PC,CKINAC ;CHECK FOR CHAR ON INACTIVE LINE.
4698 032154 103043 ;BCC 4$ ;GO REPORT ERROR IF ON INACTIVE LINE.
4699 ;*
4700 ;* Push the new character on the resync que for this line.
4701 ;*
4702 032156 010304 ;MOV R3,R4 ;CALCULATE BASE ADDRESS OF THE
4703 032160 006304 ;ASL R4 ; DATA PATTERN RESYNCH QUEUE
4704 032162 006304 ;ASL R4 ; (QUEUE IS 4 WORDS LONG) FOR

```

GLOBAL SUBROUTINE

- NEWCHR -

```

4705 032164 062704 005052      ADD    #DPRSQB,R4      ; THIS LINE.
4706 032170 010401      MOV     R4,R1          ;GET THE BASE OF THE QUEUE.
4707 032172 016121 000002      MOV     2(R1),(R1)+      ;MOVE FROM CHR1 SLOT TO CHR0 SLOT.
4708 032176 016121 000002      MOV     2(R1),(R1)+      ;MOVE FROM CHR2 SLOT TO CHR1 SLOT.
4709 032202 010211      MOV     R2,(R1)          ;PUT NEW CHAR INTO CHR2 SLOT.
4710
4711      ;+
4712      ; Check the DATA.VALID for the character at the botton of the queue.
4713      ; If DATA.VALID is clear, exit the routine--nothing to analyze.
4714 032204 011402      MOV     (R4),R2          ;GET CHR0 VALUE, SET FLAGS.
4715 032206 100076      BPL      60$              ;EXIT ROUTINE IF DATA.VALID IS CLEAR.
4716
4717      ;+
4718      ; Test for any of the error bits set in CHR0.
4719 032210 032702 070000      BIT     #70000,R2      ;TEST FOR ANY CHR0 ERROR BITS SET.
4720 032214 001420      BEQ      2$              ;SKIP THIS ERROR IF NO ERROR BITS SET.
4721
4722      ;+
4723      ; We have at least one error flag set on the received char.
4724      ; Report data error flag error if not in summary mode.
4725 032216 005337 032410      DEC     70$          ;SET THE "ERROR FOUND" FLAG.
4726 032222 016300 005364      MOV     TXRXLB(R3),R0    ;GET THE TX LINE OFFSET FOR THIS RX LINE.
4727 032226 036037 002534 002650      BIT     BITTBL(R0),ERSMRF ;CHECK THE ERROR SUMMARY FLAG FOR TX LINE.
4728 032234 001010      BNE      2$              ;IF ERROR SUMMARY FLAG SET, SKIP NEXT REPORT.
4729 032236 012737 025566 005472      MOV     #ER9003,ERRBLK ;SELECT THE ER9003 ERROR REPORT ROUTINE.
4730 032244 004737 037074      JSR     PC,TXROFF      ;TURN OFF TX AND RX DURING ERROR REPORTING.
4731 032250      ERROR      ;>>>>> ERROR <<<<<.
4732 032252 004737 037130      JSR     PC,TXRON      ;TURN TX AND RX BACK ON.      TRAP    C$ERROR
4733
4734      ;+
4735      ; Check the character at the bottom of the resync que for data errors.
4736 032256 004737 027566      JSR     PC,CKCHR      ;CHECK THE CHR0 CHAR FOR ERRORS.
4737 032262 103424      BCS      6$              ;SKIP ERROR REPORT IF CHR0 IS CORRECT.
4738
4739      ;+
4740      ; We have some sort of data error so report it (unless in summary report mode).
4741 032264 005337 032410      DEC     70$          ;SET THE "ERROR FOUND" FLAG.
4742 032270 016300 005364      MOV     TXRXLB(R3),R0    ;GET THE TX LINE OFFSET FOR THIS RX LINE.
4743 032274 036037 002534 002650      BIT     BITTBL(R0),ERSMRF ;CHECK THE ERROR SUMMARY FLAG FOR THIS LINE.
4744 032302 001014      BNE      6$              ;SKIP ERROR REPORT IF ERROR SUMMARY FLAG SET.
4745 032304 012737 025420 005472      MOV     #ER9002,ERRBLK ;SELECT THE ER9002 ERROR REPORT ROUTINE.
4746 032312 005237 005466      INC     ERNBR          ;SELECT INITIAL ERNBR + 1.
4747 032316 004737 037074      JSR     PC,TXROFF      ;TURN OFF TX AND RX DURING ERROR REPORTING.
4748 032322      ERROR      ;>>>>> ERROR <<<<<.
4749 032324 004737 037130      JSR     PC,TXRON      ;TURN TX AND RX BACK ON.      TRAP    C$ERROR
4750 032330 005337 005466      DEC     ERNBR          ;RESTORE INITIAL ERNBR.
4751
4752      ;+
4753      ; Count a character error if one occurred.
4754      ; Update the "report error summary" flag for l'ne based on error count.
4755 032334 005737 032410      TST     70$          ;CHECK THE "ERROR FOUND" FLAG.
4756 032340 001421      BEQ      60$              ;SKIP COUNTING AN ERROR IF FLAG IS CLEAR.
4757 032342 005263 003452      INC     ERCNTB(R3)      ;INCREMENT THE ERROR COUNTER FOR THIS LINE.
4758 032346 001002      BNE      8$              ;SKIP SETTING COUNTER TO MAX IF NO OVERFLOW.
4759 032350 005363 003452      DEC     ERCNTB(R3)      ;RESET THE ERROR COUNTER TO -1 (MAX VALUE).

```


GLOBAL SUBROUTINE

- NEWCHR -

```
4760 032354 005737 002264      8$:   TST   NDERPT      ;DISABLE ERROR SUMMARY FUNCTION IF
4761 032360 001411              BEQ    60$           ; NUMBER OF DATA ERRORS TO REPORT IS 0.
4762 032362 026337 003452 002264      CMP    ERCNTB(R3),NDERPT ;COMPARE ERROR COUNT WITH # OF ERR'S TO RPT.
4763 032370 103405              BLO    60$           ;SKIP SETTING OF SUMMARY FLAG IF NOT TOO MANY.
4764 032372 016300 005364      MOV    TXRXLB(R3),R0      ;GET THE TX LINE OFFSET FOR THIS RX LINE.
4765 032376 056037 002534 002650      BIS    BITTBL(R0),ERSMRF ;SET "PRINT ERROR SUMMARY" FLAG FOR LINE.
4766
4767 032404              60$:   PASS                    ;RESTORE GPRS.
      032404 004736              JSR    PC,8(SP)+        ;RETURN TO PREG05 SUBRT.
4768 032406 000207      RTS    PC
4769
4770 032410 000000      70$:   .WORD 0                  ;LOCAL STORAGE FOR ERROR OCCURRED FLAG.
```

GLOBAL SUBROUTINE

OOPS -

```

4772 .SBTTL GLOBAL SUBROUTINE                                OOPS -
4773 ;* *****
4774 ;* - Program abort subroutine -
4775 ;* This subroutine is used to abort the program when a fatal error is
4776 ;* detected in the program or the host system hardware. An error message
4777 ;* is printed giving some information about the nature of the abort.
4778 ;*
4779 ;* INPUTS:      R1 - Error code giving reason for abort.
4780 ;*
4781 ;* OUTPUTS:     An error message is printed.
4782 ;*             A list of return PC values for all subroutine calls is printed.
4783 ;*
4784 ;* CALLING SEQUENCE: JSR      PC,OOPS
4785 ;*
4786 ;* COMMENTS:
4787 ;*
4788 ;* SUBORDINATE ROUTINES CALLED: None.
4789 ;* *****
4790
4791 032412 OOPS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4792 032412 004537 005474 ;CALL REGISTER SAVE SUBRT.
4793 032416 ; REPORT "HOST COMPUTER HARDWARE OR SOFTWARE BUG ENCOUNTERED." ERROR.
4794 032416 104454 ERRSF 101,EM0101
4795 032420 000145 TRAP C$ERSF
4796 032422 032456 .WORD 101
4797 032424 000000 .WORD EM0101
4798 .WORD 0
4799 032426 ; REPORT "PROGRAM HUNG, WAITING FOR A CONTROL-C."
4800 032426 012746 032542 PRINTF #EM0102
4801 032432 012746 000001 MOV #EM0102,-(SP)
4802 032436 010600 MOV #1,-(SP)
4803 032440 104417 MOV SP,R0
4804 032442 062706 000004 TRAP C$PNTF
4805 2$: BREAK ;LOOK FOR OPERATOR CONTROL-C INPUT.
4806 60$: BR 2$ ;INFINITE LOOP.
4807 60$: PASS ;DON'T NEED THIS, BUT SOMEBODY MAY CHANGE THIS
4808 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
4809 RTS PC ; ROUTINE IN THE FUTURE, SO BE CONSISTANT.
4810
4811 EM0101:: .ASCIZ /HOST COMPUTER HARDWARE OR SOFTWARE BUG ENCOUNTERED./
4812 032456 110 117 123
4813 032461 124 040 103
4814 032464 117 115 120
4815 032467 125 124 105
4816 032472 122 040 110
4817 032475 101 122 104
4818 032500 127 101 122
4819 032503 105 040 117
4820 032506 122 040 123
4821 032511 117 106 124
4822 032514 127 101 122
4823 032517 105 040 102
4824 032522 125 107 040
4825 032525 105 116 103
4826 032530 117 125 116
4827 032533 124 105 122

```

GLOBAL SUBROUTINE

- OOPS -

	032536	105	104	056	
	032541	000			
4802	032542	045	116	045	EM0102:: .ASCIZ /N*APROGRAM HUNG, WAITING FOR A CONTRCL-C. <*****N*/
	032545	101	120	122	
	032550	117	107	122	
	032553	101	115	040	
	032556	110	125	116	
	032561	107	054	040	
	032564	127	101	111	
	032567	124	111	116	
	032572	107	040	106	
	032575	117	122	040	
	032600	101	040	103	
	032603	117	116	124	
	032606	122	117	114	
	032611	055	103	056	
	032614	040	074	052	
	032617	052	052	052	
	032622	052	052	052	
	032625	052	052	052	
	032630	052	052	052	
	032633	045	116	045	
4803	032636	116	000		

.EVEN

GLOBAL SUBROUTINE

- PRTLPR -

SEQ 0129

```

4805 .SBTTL GLOBAL SUBROUTINE - PRTLPR -
4806 ;* *****
4807 ;* -Print the contents of the LPR.
4808 ;* This routine is used to print out extended information on the
4809 ;* contents of the Line Parameter Register (LPR).
4810 ;*
4811 ;* INPUTS: R3 - Contains the number of the line you wish to examine.
4812 ;* CSRA - Contains the address of the DUT's CSR.
4813 ;* IESTAT - Contains the current status of the TX and RX interrupt
4814 ;* enable bits in the DUT's CSR.
4815 ;* LPRA - Contains the address of the DUT's LPR register.
4816 ;*
4817 ;* OUTPUTS: An extended information message is printed on the operators
4818 ;* console.
4819 ;*
4820 ;* CALLING SEQUENCE: JSR PC,PRTLPR
4821 ;*
4822 ;* COMMENTS: This routine changes the indirect address field of the device
4823 ;* under test's CSR.
4824 ;*
4825 ;* SUBORDINATE ROUTINES CALLED: NONE.
4826 ;* -- *****
4827
4828 032640 PRTLPR::SAVE
4829 032640 004537 005474 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
4830 032644 013701 002300 MOV CSRA,R1 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4831 032654 042703 177760 MOV LPRA,R2 ;GET THE CSR ADDRESS.
4832 032660 053703 002374 BIC #177760,R3 ;GET THE LPR ADDRESS.
4833 032664 010311 BIS IESTAT,R3 ;CLEAR ANY UNWANTED BITS.
4834 032666 011204 MOV R3,(R1) ;SET STATE OF TX AND RX INTERRUPT ENABLE BITS.
4835 ;PRINT MESSAGE "CONTENTS OF THE LPR:nnnnnn"
4836 032670 PRINTX #EF9019,#EM9026,R4 ;SELECT LINE.
4837 032672 010446 ;GET CONTENTS OF THE LPR.
4838 032676 012746 023407 MOV R4,-(SP)
4839 032702 012746 010310 MOV #EM9026,-(SP)
4840 032706 010600 MOV #EF9019,-(SP)
4841 032710 104415 MOV #3,-(SP)
4842 032712 062706 000010 MOV SP,R0
4843 032716 004736 TRAP C:PNTX
4844 032720 000207 ADD #10,SP
4845 604: PASS ;RESTORE GPRS.
4846 RTS PC JSR PC,#(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- PUFIFO -

```

4840 .SBTTL GLOBAL SUBROUTINE - PUFIFO -
4841 ;*****
4842 ;* - PURGE THE FIFO
4843 ;* This routine tries to remove all the characters from the FIFO.
4844 ;* Any BMP codes that are found are saved on the BMP code queue.
4845 ;*
4846 ;* INPUTS: RBUFA- Contains the address of the Receiver.
4847 ;*
4848 ;*
4849 ;* OUTPUTS: Carry bit - Indicates the state of the fifo, set:- purged.
4850 ;* BMPCQ - The contents of the BMP code queue may be updated.
4851 ;*
4852 ;* CALLING SEQUENCE: JSR PC,PUFIFO
4853 ;*
4854 ;* COMMENTS:
4855 ;*
4856 ;* SUBORDINATE ROUTINES CALLED: SAVBMP.
4857 ;*****
4858
4859 032722 PUFIFO::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
      032722 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4860 032726 012701 001000 MOV #512,,R1 ;SET MAXIMUM TRY COUNT OF 512.
4861 032732 013704 002302 MOV RBUFA,R4 ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
4862
4863 032736 011402 2$: MOV (R4),R2 ;GET THE CONTENTS OF THE RECEIVER BUFFER REG.
4864 032740 100016 BPL 6$ ;EXIT IF THE FIFO IS EMPTY, DATA_VALID CLR.
4865
4866 ;+
4867 ; Check if the read character is actually a BMP code.
4868 ; If it is, then save it on the BMP code queue to be reported later.
4869 032742 012700 070000 ;- MOV #70000,R0 ;GENERATE A BIT MAP OF CHAR ERROR BITS
4870 032746 040200 BIC R2,R0 ; WHICH ARE NOT SET FOR CHAR.
4871 032750 001006 BNE 4$ ;THROW CHAR AWAY IF NOT BMP OR SELFTEST CODE.
4872
4873 ;+
4874 ; Check if the read data is modem status, BMP or Selftest?.
4875 032752 012700 000300 ;- MOV #300,R0 ; CHECK IF BMP OR SELFTEST?.
4876 032756 040200 BIC R2,R0 ;TRY TO CLEAR BMP FLAGS IN THE READ DATA.
4877 032760 001002 BNE 4$ ;IF IT IS MODEM OR SELFTEST CODE THROW IT AWAY.
4878 032762 004737 035624 JSR PC,SAVBMP ;SAVE BMP CODE ON THE QUEUE.
4879
4880 032766 005301 4$: DEC R1 ;DECREMENT THE TRY COUNT.
4881 032770 001362 BNE 2$ ;LOOP TO TRY AGAIN.
4882 032772 000241 CLC ;CLEAR CARRY, TO INDICATE FIFO NOT PURGED.
4883 032774 000401 BR 60$ ;EXIT WITH CARRY CLEAR.
4884 032776 000261 6$: SEC ;SET CARRY, TO INDICATE FIFO PURGED.
4885
4886 033000 60$: PASS ;RESTORE GPRS,
      033000 004736 JSR PC,@(SP), ;RETURN TO PREG05 SUBRT.
4887
4888 033002 000207 RTS PC ;CARRY BIT, SET INDICATES FIFO PURGED.

```

GLOBAL SUBROUTINE

- PUFIFR -

```

4890 .SBTTL GLOBAL SUBROUTINE PUFIFR -
4891 ;*****
4892 ;* - Purge FIFO report any errors found.
4893 ;* This routine removes all data from the FIFO. Any BMP codes that are
4894 ;* and are save on the queue to be reported later in the BMP report test.
4895 ;* unexpected data (ie any non-status information) that are found,
4896 ;* reported as an error.
4897 ;* 1. the FIFO will not purge after 512 attempts, then the current test
4898 ;* that called this routine receives a failure flag that should be used
4899 ;* to abort the test.
4900 ;*
4901 ;* INPUTS: ERRIBL - ERRTYPE, ERRMSG, ERRNBR are set up correctly.
4902 ;* RBUFA- Contains the address of the Receiver.
4903 ;*
4904 ;* OUTPUTS: Carry bit - Abort test flag, C1r = ABORT TEST, Set = OK.
4905 ;* ERRBLK - Value will be destroyed.
4906 ;* BMPCQP - The BMP code queue pointer may be updated.
4907 ;* The contents of the BMP code queue may be updated.
4908 ;*
4909 ;* CALLING SEQUENCE: JSR PC,PUFIFR
4910 ;*
4911 ;* COMMENTS: This routine reports errors with numbers initial ERRNBR
4912 ;* thru to ERRNBR+2.
4913 ;* The ERRNBR is restored to its INITIAL value before returning.
4914 ;*
4915 ;* SUBORDINATE ROUTINES CALLED: ER1603,ER9001,ER9002,SAVBMP.
4916 ;*****
4917
4918 033004 PUFIFR::SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
4919 033004 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
4919 033010 013746 005466 MOV ERRNBR, (SP) ;SAVE THE CONTENTS OF THE ERROR NUMBER.
4920 033014 012705 001000 MOV #512.,R5 ;SET MAXIMUM READ COUNTER TO 2*FIFO SIZE.
4921
4922 ;*
4923 ;* Read data from the FIFO until DATA VALID is clear of read counter is zero.
4924 ;* Report any BMP or Unexpected data as errors.
4925 033020 017702 147256 2$: MOV RBUFA,R2 ;GET THE CONTENTS OF THE RECEIVER BUFFER REG.
4926 033024 100057 BPL 8$ ;EXIT IF DATA VALID CLEAR, ie. FIFO PURGED.
4927
4928 ;*
4929 ;* Check if read data is status or unexpected character.
4930 033026 012700 070000 MOV #70000,R0 ;GENERATE A BIT MAP OF CHAR ERROR BITS
4931 033032 040200 BIC R2,R0 ; WHICH ARE NOT SET FOR CHAR.
4932 033034 001012 BNE 4$ ;SKIP BMP CHECK IF IT IS UNEXPECTED DATA.
4933
4934 ;*
4935 ;* Check if the read data is modem status, BMP or Selftest?.
4936 ;* If it is a BMP code then save it on the queue.
4937 033036 012737 025330 005472 MOV #ER9001,ERRBLK ;SET UP THE CORRECT ERROR REPORTING ROUTINE.
4938 033044 012700 000300 MOV #300,R0 ; CHECK IF BMP OR SELFTEST?.
4939 033050 040200 BIC R2,R0 ;TRY TO CLEAR BMP FLAGS IN THE READ DATA.
4940 033052 001003 BNE 4$ ;SKIP BMP ERROR REPORT IF MODEM OR SELFTEST?.
4941 033054 004737 035624 JSR PC,SAVBMP ;SAVE THE BMP CODE ON THE QUEUE.
4942 033060 000424 BR 6$ ;BRANCH TO CHECK READ COUNT.
4943
4944 ;*
4945 ;* Check if the read data is Modem, Selftest or Unexpected data.

```


GLOBAL SUBROUTINE

- PUFIFR -

SEQ 0132

Address	Hex	Dec	Hex	Dec	Hex	Dec	Label	Instruction	Comment
4946	033062	032702	000001				4\$:	BIT	#BIT0,R2 ;TEST THE MODFM STATUS INDICATION BIT.
4947	033066	001421						BEQ	6\$;DO NOT REPORT ANY ERROR IF MODEM STATUS.
4948	033070	012701	023772					MOV	#EM9104,R1 ;PASS THE CORRECT ERROR MESSAGE TO REPORT.
4949	033074	010203						MOV	R2,R3 ;EXTRACT THE LINE NUMBER FROM
4950	033076	000303						SWAB	R3 ; THE READ DATA.
4951	033100	042703	177760					BIC	#177760,R3 ;
4952	033104	006303						ASL	R3 ;FORM LINE NUMBER TIMES 2 FOR ER9002 ROUTINE.
4953	033106	052704	100000					BIS	#BIT15,R4 ;SET THE "NONE" EXPECTED MESSAGE FLAG.
4954	033112	005237	005466					INC	ERRNBR ;SET ERROR NUMBER TO INITIAL ERRBR+1.
4955	033116	012737	025420	005472				MOV	#ER9002,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
4956									;REPORT ERROR "UNEXPECTED DATA FOUND IN FIFO".
4957	033124							ERROR	; >>>> ERROR <<<<.
4958	033124	104460							TRAP C\$ERROR
4958	033126	005337	005466					DEC	ERRNBR ;RESTORE ERROR NUMBER TO INITIAL ERRNBR.
4959									
4960	033132	005305					6\$:	DEC	R5 ;DECREMENT READ COUNTER.
4961	033134	001331						BNE	2\$;LOOP TO READ NEXT CHAR FROM FIFO IF COUNT > 0.
4962									
4963									; The FIFO will not clear, report the error and indicate that the test is to
4964									; be ABORTED.
4965									;
4966	033136	062737	000002	005466				ADD	#2,ERRNBR ;SET ERROR NUMBER TO INITIAL ERRNBR+2.
4967	033144	012737	025034	005472				MOV	#ER1603,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
4968	033152	012701	023052					MOV	#EM9017,R1 ;PASS THE MESSAGE TO BE REPORTED.
4969									;REPORT THE ERROR "FIFO WILL NOT PURGE, (DATA VALID STUCK SET)"
4970									; "?????? TEST ABORTED".
4971	033156							ERROR	; >>>> ERROR <<<<.
4972	033156	104460							TRAP C\$ERROR
4972	033160	000241						CLC	;INDICATE THE TEST IS TO BE ABORTED.
4973	033162	000401						BR	10\$;EXIT THIS ROUTINE AND ABORT THE CURRENT TEST.
4974									
4975	033164	000261					8\$:	SEC	;SET THE CARRY, DO NOT ABORT THE TEST.
4976									
4977	033166	012637	005466				10\$:	MOV	(SP)+,ERRNBR ;RESTORE INITIAL ERROR NUMBER.
4978	033172						60\$:	PASS	;RESTORE GPRS,
4979	033172	004736							JSR PC,#(SP)+ ;RETURN TO PREG05 SUBRT.
4980									;CARRY BIT, SET INDICATES FIFO PURGED, DO NOT
4981	033174	000207						RTS	PC ; ABORT THE TEST.

GLOBAL SUBROUTINE

PURRXB

```

4983 .SBTTL GLOBAL SUBROUTINE PURRXB -
4984 ;* *****
4985 ;* - Purge the RX Buffer in Memory Routine -
4986 ;* This subroutine is used before the beginning of a TX/RX of data
4987 ;* patterns to clear out the RX buffer and to initialize the various
4988 ;* counters and pointers related to that buffer.
4989 ;*
4990 ;* INPUTS: RXBSTA - Label at the beginning of the RX buffer.
4991 ;*
4992 ;* OUTPUTS: RXBCNT - Count of # of chars in RX buffer (Cleared).
4993 ;*           RXBIPT - Input pointer to RX buffer (Initialized).
4994 ;*           RXBOPT - Output pointer to RX buffer (Initialized).
4995 ;*           The contents of the RX BUFFER are cleared.
4996 ;*
4997 ;* CALLING SEQUENCE: JSR PC,PURRXB
4998 ;*
4999 ;* COMMENTS:
5000 ;*
5001 ;* SUBORDINATE ROUTINES CALLED: None.
5002 ;* *****
5003
5004 033176 PURRXB:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5005 033176 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5006 033202 012701 003062 MOV #RXBOPT,R1 ;GET THE ADDRESS OF THE RX OUTPUT POINTER.
5007 033206 012721 003070 MOV #RXBSTA,(R1)+ ;INITIALIZE THE RX BUFFER OUTPUT POINTER.
5008 033212 012721 003070 MOV #RXBSTA,(R1)+ ;INITIALIZE THE RX BUFFER INPUT POINTER.
5009 033216 005021 2$: CLR (R1)+ ;CLEAR CHAR COUNT AND THE BUFFER AREA.
5010 033220 020127 003270 CMP R1,#RXBEND ;CHECK IF LAST LOCATION HAS BEEN CLEARED.
5011 033224 101774 BLOS 2$ ;LOOP IF NOT DONE.
5012
5013 033226 60$: PASS ;RESTORE GPRS.
5014 033230 004736 000207 JSR PC,#(SP)+ ;RETURN TO PREG05 SUBRT.
5014 033230 000207 RTS PC

```


GLOBAL SUBROUTINE

- RDCHRS -

```

5016 .SBTTL GLOBAL SUBROUTINE - RDCHRS -
5017 ;* *****
5018 ;* - Read and Compare Input Characters Routine -
5019 ;* This subroutine reads the characters from the RX buffer in memory.
5020 ;* If characters stop appearing in the buffer with DATA.VALID set
5021 ;* or if more than the allowable number of characters has been read from
5022 ;* the buffer this routine exits with an RX complete indication.
5023 ;* Each read char is analyzed and any necessary errors are reported.
5024 ;*
5025 ;* INPUTS: ACTLNS - Bit map of the active DUT lines.
5026 ;*          ERRNBR - Set to error number of first error in this routine.
5027 ;*          IBM - Mask of the inactive bits in a TX or RX char byte.
5028 ;*          OSTEND - Address of the end of the output storage fifo buffer.
5029 ;*          OSTPTR - Pointer to the next byte to read from OSTORE.
5030 ;*          RXBOPT - Pointer into the RX char buffer in memory.
5031 ;*          RXTOUT - Time-out value for RX of last char.
5032 ;*
5033 ;* OUTPUTS: Error messages may be printed at the operator's console.
5034 ;*          TXDBLF - TX/RX disabled flag (Cleared).
5035 ;*          TXENBM - TX.ENABLE state mask (Destroyed).
5036 ;*          SAVPRI - Storage for processor priority (Destroyed).
5037 ;*          SAVTEN - Storage for TX.ENABLE states (Destroyed).
5038 ;*
5039 ;* CALLING SEQUENCE: JSR PC,RDCHRS
5040 ;*
5041 ;* COMMENTS: This routine reports errors with numbers Initial ERRNBR
5042 ;*            thru Initial ERRNBR + 4.
5043 ;*            ERRNBR is restored before this routine returns.
5044 ;*
5045 ;* SUBROUTINES CALLED: CKCHR,NEWCHR,REPCOD,RXIE0,RXIE1,TXENBL,TXIE0,TXIE1,
5046 ;*                    WAIBIS.
5047 ;* *****
5048 ;*
5049 033232 RDCHRS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5050 033232 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5051 033236 013704 005466 ;MOV ERRNBR,R4 ;PRESERVE THE INITIAL ERROR NUMBER.
5052 033242 013703 002366 ;MOV IBM,R3 ;GET THE INACTIVE BIT MASK.
5053 033246 005037 002656 ;CLR TXDBLF ;CLEAR THE TX DISABLED FLAG.
5054 033252 004737 035600 ;JSR PC,RXIE1 ;TURN ON DUT RECEPTION INTERRUPTS.
5055 033256 004737 036574 ;JSR PC,TXIE1 ;TURN ON DUT TRANSMISSION INTERRUPTS.
5056 ;*
5057 ;* Clear all resync queues for all lines.
5058 ;*
5059 033262 012701 005052 ;MOV #DPRSQB,R1 ;GET BASE ADDRESS OF RESYNC QUEUES TABLE.
5060 033266 012702 005252 ;MOV #DPRSQE,R2 ;GET END ADDRESS OF RESYNC QUEUES TABLE.
5061 033272 005021 2$: ;CLR (R1)+ ;CLEAR A WORD OF THE TABLE.
5062 033274 020102 ;CMP R1,R2 ;CHECK IF POINTER AT END OF TABLE.
5063 033276 103775 ;BLO 2$ ;LOOP UNTIL TABLE IS CLEAR.
5064 ;*
5065 ;* Wait for a character to appear in the FIFO.
5066 ;* If no character appears within time-out period: exit routine, we're done.
5067 ;*
5068 033300 013701 002402 4$: ;MOV RXTOUT,R1 ;GET TIME OUT FOR SLOWEST BAUD RATE IN USE.
5069 033304 023737 002652 002272 ;CMP TXDONF,ACTLNS ;CHECK FOR TRANSMISSION DONE ON ACTIVE LINES.
5070 033312 001402 ;BEQ 6$ ;SKIP ADDING 50 MS DELAY IF TX DONE ALL LINES.
5071 033314 062701 000062 ;ADD #50.,R1 ;ADD 50 MILLI SEC TO DELAY IF NOT LAST CHAR.

```

GLOBAL SUBROUTINE

- RDCHRS -

```

5072 033320 052701 170000      6$:   BIS    #170000,R1      ;INDICATE TO TEST DATA.VALID BIT.
5073 033324 013702 003062      MOV    RXBOPT,R2      ;INDICATE TO CHECK MEMORY RECEIVE BUFFER.
5074 033330 004737 037740      JSR     PC,WAIBIS      ;WAIT FOR RECEIVED CHAR OR TIME-OUT.
5075 033334 103073              BCC     10$          ;EXIT ROUTINE IF TIME-OUT, WE'RE DONE.
5076
5077 033336 004737 031052      JSR     PC,GETCHR      ;READ A CHARACTER FROM THE MEMORY BUFFER.
5078
5079      ;+
5080      ; Check if the TX ISR is disabled.
5081      ; Re enable RX ISR if the space for new chars is low enough.
5082      ; If the buffer can accomodate more chars then re-enable transmiss'ion.
5083 033342 005737 002656      8$:   TST     TXDBLF      ;CHECK IF TX IS DISABLED.
5084 033346 100024              BPL     10$          ;SKIP RX/TX CHECK IF TX NOT DISABLED.
5085 033350 023727 003066 000020  CMP    RXBCNT,#RXBETX ;COMPARE BUFFER COUNT WITH LEVEL TO ENABLE RX.
5086 033356 101020              BHI     10$          ;SKIP ENABLE RX IF BUFFER TOO FULL.
5087 033360 004737 035600      JSR     PC,RXIE1      ;ENABLE RECEPTION INTERRUPTS.
5088 033364 013705 002410      MOV    TXENBM,R5      ;GET THE PRESERVED TX.ENABLE STATES.
5089 033370 023727 003066 000020  CMP    RXBCNT,#RXBETX ;COMPARE BUFFER COUNT WITH LEVEL TO ENABLE TX.
5090 033376 101010              BHI     10$          ;SKIP ENABLING TX IF BUFFER TOO FULL.
5091 033400 106701              MFPS    R1           ;SAVE THE CURRENT PROCESSOR PRIORITY.
5092 033402 106427 000340      MTPS    #PRI07        ;DISABLE INTERRUPTS.
5093 033406 004737 036446      JSR     PC,TXENBL     ;ENABLE TRANSMISSION.
5094 033412 005037 002656      CLR     TXDBLF      ;CLEAR THE TX DISABLE FLAG.
5095 033416 106401              MTPS    R1           ;RE-ENABLE INTERUPTS.
5096 033420
5097      10$:
5098 033420 005337 002646      DEC     CHRTOT      ;DECREMENT THE TOTAL CHAR COUNTER.
5099 033424 001011              BNE     12$          ;SKIP ERROR IF NOT TOO MANY RECEIVED.
5100 033426 010437 005466      MOV     R4,ERRNBR     ;SET ERROR NUMBER TO INITIAL ERRNBR.
5101 033432 012701 023313      MOV     #EM9025,R1      ;SELECT THE PROPER ERROR MESSAGE.
5102 033436 012737 024644 005472  MOV     #ER0503,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
5103
5104      ;+
5105      ; Report error at Initial ERRNBR.
5106      ; "MORE THAN TWICE THE EXPECTED NUMBER OF CHARACTERS RECEIVED."
5107
5108 033444 104460              ERROR
5109 033446 000452              BR      60$          ;EXIT THE ROUTINE, WE'RE GIVING UP.
5110
5111      ;+
5112      ; Determine if the character is data or a status code.
5113
5114 033450 012700 070000      12$:  MOV     #70000,R0      ;GENERATE A BIT MAP OF CHARACTER ERROR BITS
5115 033454 040200              BIC     R2,R0          ; WHICH ARE NOT SET FOR THE CHARACTER.
5116 033456 001007              BNE     14$          ;SKIP REPORTING OF ERROR CODE IF WE HAVE CHAR.
5117
5118      ;+
5119      ; The data is either a BMP code or a Modem Status code.
5120      ; Report that the code was found.
5121      ; Errors reported with error numbers >>>> ERRNBR+1 and ERRNBR+2 <<<<.
5122
5123 033460 010437 005466      MOV     R4,ERRNBR     ;GET THE ERROR NUMBER PASSED INTO THIS ROUTINE.
5124 033464 005237 005466      INC     ERRNBR      ;SET ERROR NUMBER TO INITIAL ERRNBR+1.
5125 033470 004737 034350      JSR     PC,REPCOD     ;REPORT THE BMP OR MODEM STATUS CHANGE CODE.
5126 033474 000407              BR      16$          ;BRANCH TO GET THE NEXT CHARACTER.
5127
5128      ;+
5129      ; The data is a valid character:
5130      ; Compare the read data with the expected data.
5131      ; Update expected data pointer.

```

GLOBAL SUBROUTINE

- RDCHRS -

```

5128      ; Errors reported with error numbers >>>> ERRNBR+3 and ERRNBR+4 <<<<.
5129      ;-
5130 033476 010437 005466      14$:  MOV    R4,ERRNBR      ;CALCULATE THE STARTING ERROR NUMBER FOR THE
5131 033502 062737 000003 005466  ADD    #3,ERRNBR      ; NEXT ROUTINE CALL (INITIAL ERRNBR+3).
5132 033510 004737 032132      JSR     PC,NEWCHR      ;HANDLE THE NEW DATA CHARACTER.
5133      ;+
5134      ; Done processing this character.
5135      ; Read another char from the DUT FIFO.
5136      ; If DATA.VALID is set, loop to check the received character.
5137      ; If DATA.VALID is clear loop to wait for it set or time-out.
5138      ;-
5139 033514 004737 031052      16$:  JSR     PC,GETCHR      ;READ A CHARACTER FROM THE RX BUFFER.
5140 033520 103710              BCS     8$              ;IF DATA.VALID SET, GO TO CHECK THE RX CHAR.
5141 033522 000666              BR      4$              ;LOOP TO WAIT CHAR OR TIME-OUT IF BUFFER EMPTY.
5142      ;+
5143      ; Use dummy characters to force analysis of characters in resync queues.
5144      ;-
5145 033524 004737 035546      18$:  JSR     PC,RXIE0      ;TURN OFF DUT RX INTERRUPTS.
5146 033530 004737 036242      JSR     PC,TXDONE      ;CHECK IF TX DONE, TURN OFF DUT TX INTERRUPTS.
5147 033534 005002              CLR     R2              ;CLEAR THE DUMMY CHARACTER.
5148 033536 005001              CLR     R1              ;CLEAR THE LOOP COUNTER.
5149 033540 004737 032132      20$:  JSR     PC,NEWCHR      ;FORCE ONE RESYNC QUE CHAR TO BE ANALYZED.
5150 033544 062702 000400      ADD     #400,R2          ;INCREMENT THE LINE NUMBER IN THE DUMMY CHAR.
5151 033550 005201              INC     R1              ;INCREMENT THE LOOP COUNTER.
5152 033552 120127 000010      CMPB    R1,#NUMLNS      ;TEST FOR LOOP COUNTER EQUAL TO # OF DUT LINES.
5153 033556 002770              BLT     20$              ;LOOP IF LOOP COUNT IS NOT ALL LINES DONE.
5154 033560 005701              TST     R1              ;CHECK FOR SECOND TIME AROUND OUTER LOOP.
5155 033562 100404              BMI     60$              ;EXIT IF OUTER LOOP DONE TWICE.
5156 033564 005002              CLR     R2              ;CLEAR THE DUMMY CHAR FOR 2ND TIME AROUND LOOP.
5157 033566 012701 100000      MOV     #100000,R1        ;CLEAR LOOP COUNT, SET OUTER LOOP FLAG.
5158 033572 000762              BR      20$              ;LOOP THE SECOND TIME AROUND OUTER LOOP.
5159      ;-
5160 033574 010437 005466      60$:  MOV     R4,ERRNBR      ;RESTORE THE ERROR NUMBER TO ITS INITIAL VALUE.
5161 033600              PASS              ;RESTORE GPRS.
5162 033602 004736              JSR     PC,0(SP)+        ;RETURN TO PREG05 SUBRT.
5162 033602 000207              RTS     PC

```

GLOBAL SUBROUTINE

- RDMAST -

```

5164 .SBTTL GLOBAL SUBROUTINE RDMAST -
5165 ;** *****
5166 ;* - Report DMA_START Bit Errors Routine -
5167 ;* This subroutine checks for lines which have DMA_START bit errors
5168 ;* during the just completed DMA transmission. If any are found,
5169 ;* they are reported.
5170 ;*
5171 ;* INPUTS: ERRMSG - Address of primary error message for this routine.
5172 ;*          ERRNBR - Error number of error reported in this routine.
5173 ;*          TXINTF - Contains bit map of lines with DMA_START bit errors.
5174 ;*
5175 ;* OUTPUTS: ERRBLK - Address of the error reporting routine (Destroyed).
5176 ;*            Messages may be printed at the operator console.
5177 ;*
5178 ;* CALLING SEQUENCE: JSR PC,RDMAST
5179 ;*
5180 ;* COMMENTS: If no lines have DMA_START bit errors, no messages are printed.
5181 ;*
5182 ;* SUBORDINATE ROUTINES CALLED: ER9102.
5183 ; *****
5184
5185 033604 RDMAST:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5186 033604 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5187 033610 013702 002420 MOV TXINTF,R2 ;GET COPY OF THE DMA_START ERRORS BIT MAP.
5188 033614 001406 BEQ 60$ ;EXIT IF NO DMA_START ERROR BITS ARE SET.
5189 ;*
5190 ; We have some DMA_START bit errors to report.
5191 033616 012737 026466 005472 MOV #ER9102,ERRBLK ;SELECT THE ERROR REPORTING ROUTINE.
5192 033624 012701 023703 MOV #EM9102,R1 ;INDICATE THAT WE HAVE DMA_START BIT ERROR.
5193 ;*
5194 ; Report "DMA_START BIT SET AFTER RESET OR TX.ACTION ... ON LINES(S):"
5195 ;
5196 033630 ERROR ;
5197 033630 104460 ;>>>> ERROR <<<<.
5198 033632 004736 TRAP C$ERROR
5199 033634 000207 60$: PASS ;RESTORE GPRS.
RTS PC JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- RDPDR -

```

5201      .SBTTL GLOBAL SUBROUTINE                                RDPDR -
5202      ;*****
5203      ;* - Read and verify Data Pattern from Device Registers Routine -
5204      ;* This routine reads and verifies the rotated data pattern which has
5205      ;* been written by the WDPDR subroutine.
5206      ;* Each active line's register's contents is read and compared with the
5207      ;* written data.
5208      ;* After the unused and Read Only (RO) bits are masked out, any errors are
5209      ;* reported from this routine.
5210      ;* This routine will take into account the type of write operation which
5211      ;* was performed by the WDPDR subroutine.
5212      ;*
5213      ;* INPUTS:      R2 - Used to pass in the data pattern to be rotated & verified.
5214      ;*              R3 - Byte indicator (- => lo byte, + => hi byte, 0 => both).
5215      ;*              R4 - Operation type indicator (- => BIC, + => BIS, 0 => MOV).
5216      ;*              ACTLNS - Bit map of active lines on the device under test.
5217      ;*              CSRA - Contains the CSR address of the Device under test.
5218      ;*              DRADRT - Base address of device register address table.
5219      ;*              ERCNTB - Label at base of error counters table for lines.
5220      ;*              ERRMSG - Set up with the proper error message for this test.
5221      ;*              ERRNBR - Set up with the proper error number.
5222      ;*              LPRO - Equated to LPR reg offset from device CSR address.
5223      ;*              NUMLNS - Number of lines on the device under test.
5224      ;*              NDERPT - Number of individual data errors to report on a line.
5225      ;*              TXBFCO - Equated to TBUFFCT reg offset from device CSR address.
5226      ;*              UNBTTB - Base address of the unused bit table.
5227      ;*
5228      ;* OUTPUTS:     Error messages may be printed at the operator's console.
5229      ;*              ERCNT - Error counters table is updated for line under test.
5230      ;*              ERRBLK - Contents destroyed.
5231      ;*              ERSMRF - Error summary flags bit set if line in summary mode.
5232      ;*              UUT CSR - All bits cleared, except IND.ADR.REG field destroyed.
5233      ;*
5234      ;* CALLING SEQUENCE: JSR PC,RDPDR
5235      ;*
5236      ;* COMMENTS:     For byte accesses, only the specified byte is verified.
5237      ;*
5238      ;* SUBORDINATE ROUTINES CALLED: ER1601,ROLDAP.
5239      ;-- *****
5240
5241      033636      RDPDR:: SAVE                                ;SAVE CONTENTS OF GPRS R0 THRU R5.
5242      033636      004537      005474      JSR R5,PREG05      ;CALL REGISTER SAVE SUBRT.
5243      033642      012737      024736      005472      MOV #ER1601,ERRBLK ;SE, UP THE ADDRESS OF THE ERROR REPORT RTN.
5244      ;*
5245      ;* Determine whether register data should be inverted from data pattern.
5246      033650      005704      TST R4                                ;CHECK THE OPERAND TYPE INDICATOR.
5247      033652      100001      BPL 2$                                ;BIC WRITE PERFORMED? NO, USE STANDARD DATA.
5248      033654      005102      COM R2                                ;YES, INVERT THE DATA PATTERN.
5249      ;*
5250      ;* Set up outer loop.
5251      ;
5252      033656      005005      2$: CLR R5                                ;CLEAR LINE COUNTER TO SELECT LINE 0.
5253      ;*
5254      ; The outer loop follows. Each pass through this loop reads and compares data
5255      ; from all of the device registers for a particular line if the line is active.
5256      ;--

```

GLOBAL SUBROUTINE

RDPDR -

```

5257 033660 010237 034054      4$:  MOV    R2,70$      ;SAVE THE OUTER LOOP DATA PATTERN.
5258 033664 010577 146410      MOV    R5,0CSRA    ;SET CSR IND.ADP.REG FIELD TO THIS LINE.
5259 033670 010500              MOV    R5,R0
5260 033672 006300              ASL     R0
5261 033674 036037 002534 002272  BIT    BITTBL(R0),ACTLNS
5262 033702 001452              BEQ     16$      ;IS THE LINE ACTIVE? NO, SKIP THE LINE.
5263 033704 012703 000004      MOV     @LPRO,R3    ;YES, INITIALIZE REGISTER OFFSET FOR LPR.
5264
5265      ;+
5266      ; The inner loop follows. Each pass through this loop reads and compares
5267      ; data from a device register.
5268 033710 010204      6$:  MOV     R2,R4      ;SAVE THE INNER LOOP DATA PATTERN.
5269 033712 046302 002320      BIC     UNBITB(R3),R2    ;REMOVE UNUSED BITS FROM EXPECTED DATA.
5270 033716 016300 002300      MOV     DRADRT(R3),R0
5271 033722 005766 000010      TST     R3SLOT(SP)    ;CHECK THE ACCESS TYPE INDICATOR.
5272 033726 001002      BNE     8$      ;BYTE ACCESS? YES, GO PERFORM BYTE READ.
5273 033730 011001      MOV     (R0),R1    ;NO, PERFORM WORD READ OF DEVICE REGISTER.
5274 033732 000416      BR       12$
5275 033734 100410      8$:  BMI     10$      ;LOW BYTE ACCESS? YES, GO DO LOW BYTE READ.
5276 033736 005200      INC     R0      ;HIGH BYTE ACCESS. FORM HIGH BYTE ADDRESS.
5277 033740 111001      MOV     (R0),R1    ;READ THE HI BYTE OF THE DUT REGISTER.
5278 033742 000301      SWAB     R1      ;PUT HI BYTE BACK INTO THE HI BYTE.
5279 033744 042701 000377      BIC     @377,R1    ;REMOVE THE UNUSED BYTE IN ACTUAL DATA.
5280 033750 042702 000377      BIC     @377,R2    ;REMOVE THE UNUSED BYTE IN EXPECTED DATA.
5281 033754 000405      BR       12$
5282 033756 111001      10$:  MOV     (R0),R1    ;READ THE LOW BYTE OF THE DUT REGISTER.
5283 033760 042701 177400      BIC     @177400,R1    ;REMOVE THE UNUSED BYTE.
5284 033764 042702 177400      BIC     @177400,R2    ;FORM EXPECTED LOW BYTE FOR COMAPARISON.
5285
5286 033770 046301 002320      12$:  BIC     UNBITB(R3),R1    ;REMOVE UNUSED BITS FROM ACTUAL DATA.
5287 033774 020102      CMP     R1,R2      ;COMPARE ACTUAL AND EXPECTED DATA.
5288 033776 001404      BEQ     14$      ;ACTUAL = EXPECTED? YES, SKIP ERROR.
5289 034000 004737 030236      JSR     PC,CTERR    ;NO, COUNT THE ERROR, CHECK FOR ERROR SUMMARY.
5290 034004 103401      BCS     14$      ;USE ERROR SUMMARY? YES, SKIP ERROR.
5291      ;No, report "BAD BIT(S) IN DEVICE xxxxx REGISTER FOR LINE nn (D)."
5292 034006      TRAP     C$ERROR
5293 034006 104460      14$:  MOV     R4,R2      ;RESTORE THE INNER LOOP DATA PATTERN.
5294 034012 004737 034642      JSR     PC,ROLDAP    ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
5295 034016 062703 000002      ADD     @2,R3      ;SET REGISTER OFFSET TO THE NEXT REGISTER.
5296 034022 020327 000016      CMP     R3,@TXBFC0    ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
5297 034026 003730      BLE     6$      ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
5298
5299      ;+
5300      ; Back into the outer loop. Now set up for next line. Loop if not done.
5301 034030 013702 034054      16$:  MOV     70$,R2    ;SET UP TO ROTATE THE DATA PATTERN.
5302 034034 004737 034642      JSR     PC,ROLDAP    ;ROTATE THE DATA PATTERN.
5303 034040 005205              INC     R5      ;COUNT THIS LINE
5304 034042 020527 000010      CMP     R5,@NUMLNS    ;COMPARE LINE COUNT WITH NUMBER OF LINES.
5305 034046 002704              BLT     4$      ;LOOP IF SOME LINES NOT DONE.
5306
5307 034050      60$:  PASS              ;RESTORE GPRS.
5308 034050 004736              RTS     PC      JSR     PC,@(SP)+    ;RETURN TO PREG05 SUBRT.
5309 034052 000207
5310 034054 000000      70$:  .WORD    0      ;STORAGE FOR DATA PATTERN OUTSIDE INNER LOOP.

```


GLOBAL SUBROUTINE

- READBX -

```

5312 .SBTTL GLOBAL SUBROUTINE - READBX -
5313 ;+ *****
5314 ; - READ CHARACTERS FROM THE FIFO AND CHECKS FOR BMPS AND XONS-
5315 ; This subroutine is used in the FIHAVL.TST.
5316 ; It reads the specified number of characters from the fifo and checks
5317 ; for BMP codes and XON characters.
5318 ;
5319 ; INPUTS: R0 - Contains the number of chars to read from the fifo.
5320 ;
5321 ; OUTPUTS: R1 - Contains address of error message to be reported
5322 ; Clear if no error found.
5323 ; Carry used to indicate if fifo was found empty, carry clear.
5324 ;
5325 ; CALLING SEQUENCE: JSR PC,READ
5326 ;
5327 ; COMMENTS:
5328 ;
5329 ; SUBORDINATE ROUTINES CALLED: CHKBMP.
5330 ;- *****
5331
5332 034056 READBX:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5333 034056 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5334 034062 005001 ;CLEAR GPR THAT HOLDS THE ADDRESS OF ERRMSG.
5335 034064 013703 002302 MOV RBUFA,R3 ;GET THE ADDRESS OF THE RECEIVER BUFFER REG.
5336 034070 011302 2$: MOV (R3),R2 ;READ A CHARACTER FROM THE FIFO.
5337 034072 100015 BPL 8$ ;BRANCH IF FIFO IS EMPTY.
5338 ;+
5339 ; Check if the read character is a BMP code.
5340 ; If it is a BMP code save it on the queue to be reported later, and
5341 ; abort the test.
5342 034074 004737 027256 JSR PC,CHKBMP ;CHECK IF CHARACTER IS A BMP CODE.
5343 034100 103410 BCS 6$ ;BRANCH IF A BMP CODE WAS FOUND.
5344 034102 120227 000021 CMPB R2,421 ;CHECK IF IT IS AN XON.
5345 034106 001003 BNE 4$ ;BRANCH IF NOT AN XON.
5346 034110 012701 021617 MOV 4EM5402,R1 ;PASS THE MESSAGE TO BE REPORTED.
5347 034114 000402 BR 6$ ;GO EXIT TEST.
5348 034116 005300 4$: DEC R0 ;DECREMENT THE READ COUNT.
5349 034120 001363 BNE 2$
5350 034122 000261 6$: SEC ;SET CARRY TO INDICATE SUCCESS.
5351 034124 000401 BR 60$ ;EXIT
5352 034126 000241 8$: CLC ;CLEAR CARRY BIT TO INDICATE FAILURE.
5353
5354 034130 60$: PASS R1 ;RESTORE GPRS,
5355 034130 010166 000004 MOV R1,R1SLOT(SP) ;PUT R1 IN STACK SLOT.
5356 034134 004736 JSR PC,4(SP)+ ;RETURN TO PREG05 SUBRT.
5357 034136 000207 RTS PC

```

GLOBAL SUBROUTINE

- REGTST -

```

5357 .SBTTL GLOBAL SUBROUTINE - REGTST -
5358 ;* *****
5359 ;* - Registers Test Subroutine -
5360 ;* Subroutine to test the Device Under Test (DUT) registers. The used
5361 ;* bits of the registers are either all cleared or all set and then the
5362 ;* data pattern is written and verified using either word or byte
5363 ;* accesses in read/write or read/modify/write mode.
5364 ;*
5365 ;* INPUTS: R3 - Byte indicator (- => low, + => high, 0 => both bytes).
5366 ;* R4 - Access mode (-1 => set then BIC, 1 => clear then BIS,
5367 ;* (-2 => set then MOV, +2 clear then MOV).
5368 ;* ERRNBR - Set up with initial error number.
5369 ;*
5370 ;* OUTPUTS: GPRS0 - GPR save area 0 is destroyed.
5371 ;* Device Under Test registers are written.
5372 ;* Error messages may be printed at the operators console.
5373 ;*
5374 ;* CALLING SEQUENCE: JSR PC,REGTST
5375 ;*
5376 ;* COMMENTS: This routine loop 16 times writing the same data pattern
5377 ;* rotated left once each iteration.
5378 ;* This routine can report errors INITIAL ERRNBR thru INITIAL+2.
5379 ;*
5380 ;* SUBORDINATE ROUTINES CALLED: RDPDR,ROLDAP,SWAPO,WDPDR
5381 ;* *****
5382
5383 034140 REGTST:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5384 034140 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5385 ;*
5386 ; Set up the GPRs for the writting of the data pattern.
5387 ;*
5387 034144 012705 000020 MOV #16,R5 ;SET UP LOOP COUNTER TO COUNT 16 ITERATIONS.
5388 034150 012702 167410 MOV #167410,R2 ;INITIALIZE THE DATA PATTERN.
5389 034154 032704 000001 BIT #BIT0,R4 ;TEST FOR R/W ACCESS.
5390 034160 001001 BNE 2$ ;R/M/W ACCESS? YES, R4 IS ALL SET UP.
5391 034162 005004 CLR R4 ;NO, INDICATE R/W ACCESS.
5392 034164
5393 2$:
5394 ;*
5395 ; Set up the GPRs for the clearing or setting of all the used bits.
5396 ;*
5396 034164 010400 MOV R4,R0 ;PASS OPERATION TYPE INDICATOR AROUND SWAPO.
5397 034166 004737 036036 JSR PC,SWAPO ;GET ALTERNATE GPR SET IN R1 THRU R5.
5398 034172 013701 005466 MOV ERRNBR,R1 ;SAVE THE INITIAL ERROR NUMBER.
5399 034176 010004 MOV R0,R4
5400 034200 005404 NEG R4 ;SET UP OP TYPE FOR CLEARING OR SETTING.
5401 034202 005002 CLR R2 ;SET UP CLEAR WRITE PATTERN.
5402 034204 026627 000012 000002 CMP R4,SLOT(SP),#2 ;TEST FOR CLEAR THEN MOV TEST SEQUENCE.
5403 034212 001401 BEQ 4$ ;CLEAR THEN MOV? YES, LEAVE WRITE PAT CLEAR.
5404 034214 005102 COM R2 ;NO, SET ALL BITS OF WRITE PATTERN.
5405 034216 005003 4$: CLR R3 ;INDICATE THAT WORD ACCESSES SHOULD BE USED.
5406 034220 005000 CLR R0 ;SET ALTERNATE BYTE EXPECTED DATA PAT TO CLEAR.
5407 034222 026627 000012 177776 CMP R4,SLOT(SP),#-2 ;TEST FOR SET THEN MOV TEST SEQUENCE.
5408 034230 001001 BNE 6$ ;SET THEN MOV? YES, LEAVE ALT BYTE PAT CLEAR.
5409 034232 005100 COM R0 ;NO, SET ALT BYTE EXPECTED DATA PAT TO ALL 1'S.
5410 034234 004737 036036 6$: JSR PC,SWAPO ;RESTORE SWAPPED GPR VALUES TO R1 THRU R5.
5411 ;*
5412 ; Start of data pattern loop.

```


GLOBAL SUBROUTINE

- REGTST -

```

5413
5414 034240      ;
5415      8$:
5416      ;
5417      ; Set or clear all the used bits of the device registers for all lines.
5418      ; Verify that all the bits were set or cleared correctly.
5419 034240 004737 036036      JSR    PC,SWAPO      ;GET ALTERNATE GPRS FOR SETTING INITIAL STATES.
5420 034244 004737 040054      JSR    PC,WDPDR      ;GO CLEAR ALL USED REGISTER BITS, ALL LINES.
5421 034250 010137 005466      MOV    R1,ERRNBR      ;SET UP ERROR NUMBER TO INITIAL ERRNBR.
5422 034254 004737 033636      JSR    PC,RDPDR      ;VERIFY ALL USED REGISTER BITS, ALL LINES.
5423 034260 004737 036036      JSR    PC,SWAPO      ;RESTORE MAIN GPRS CONTENTS.
5424
5425      ;
5426      ; Write data patterns, all lower byte used bits, all registers, all lines.
5427      ; Verify that the data pattern was written correctly.
5428 034264 004737 040054      JSR    PC,WDPDR      ;WRITE DATA PATTERN TO DEVICE REGISTERS.
5429 034270 005237 005466      INC     ERRNBR      ;SET ERROR NUMBER TO INITIAL+1.
5430 034274 004737 033636      JSR    PC,RDPDR      ;VERIFY DATA PATTERN IN ALTERED BYTE(S).
5431 034300 005703              TST     R3          ;CHECK THE BYTE INDICATOR.
5432 034302 001411              BEQ     10$          ;WORD ACCESS? YES, SKIP SECOND BYTE CHECK.
5433
5434      ;
5435      ; Check that the alternate (unmodified) byte is clear or set as expected.
5436 034304 010201              MOV     R2,R1          ;SAVE THE DATA PATTERN.
5437 034306 010002              MOV     R0,R2          ;GET THE ALTERNATE BYTE EXPECTED DATA.
5438 034310 005403              NEG     R3            ;INDICATE THAT OTHER BYTE IS TO BE CHECKED.
5439 034312 005237 005466      INC     ERRNBR      ;SET ERROR NUMBER TO INITIAL+2.
5440 034316 004737 033636      JSR    PC,RDPDR      ;VERIFY DATA PATS IN OTHER BYTES OF REGISTERS.
5441 034322 005403              NEG     R3            ;RESTORE BYTE INDICATOR.
5442 034324 010102              MOV     R1,R2          ;RESTORE DATA PATTERN.
5443
5444      ;
5445      ; Prepare the next data pattern and loop if not done.
5446 034326 004737 034642      10$: JSR    PC,ROLDAP      ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
5447 034332 005305              DEC     R5            ;COUNT THIS ITERATION OF THE LOOP.
5448 034334 003341              BGT     8$            ;ALL PATTERNS DONE? NO, LOOP.
5449
5450 034336 013737 002634 005466 60$: MOV    GPRS08,ERRNBR      ;YES, RESTORE ERROR NUMBER AND EXIT.
5451 034344              PASS              ;GET THE ERROR NUMBR FROM GPR SWAP STORAGE.
5452 034346 004736              JSR    PC,0(SP)+      ;RESTORE GPRS.
5452 034346 000207              RTS     PC              ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- REPCOD -

SEQ 0143

```

5454 .SBTTL GLOBAL SUBROUTINE - REPCOD -
5455 ;** *****
5456 ;* - Routine to Report Error Code From DUT -
5457 ;* This routine reports an error code which has been read from the DUT
5458 ;* FIFO. The code is checked to determine whether it is a Selftest code
5459 ;* an Modem Status Change code or a BMP code. This routine assumes that
5460 ;* the code indicates an error. If a BMP code is found it is not reported
5461 ;* immediately, but is saved on the BMP code queue to be reported later.
5462 ;*
5463 ;* INPUTS: R2 - Contains the error code complete with flags and line #.
5464 ;* ERRABL - ERRTP,ERRNBR,and ERRMSG set up correctly.
5465 ;*
5466 ;* OUTPUTS: ERRBLK - Value may be destroyed.
5467 ;* BMPCQP - Maybe updated if a BMP code is added to the queue.
5468 ;*
5469 ;* CALLING SEQUENCE: JSR PC,REPCOD
5470 ;*
5471 ;* COMMENTS: ERRNBR is restored to its entering value by this routine.
5472 ;* This routine reports errors with numbers ERRNBR thru ERRNBR+1.
5473 ;*
5474 ;* SUBORDINATE ROUTINES CALLED: ER9001,SAVBMP.
5475 ;-- *****
5476 034350 REPCOD:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5477 034350 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5478 034354 012737 025330 005472 MOV #ER9001,ERRBLK ;SELECT THE ERROR REPORTING ROUTINE.
5479 034362 013703 005466 MOV ERRNBR,R3 ;PRESERVE THE ERROR NUMBER.
5480 034366 010204 MOV R2,R4 ;EXTRACT THE LINE NUMBER FIELD
5481 034370 000304 SWAB R4 ; FROM THE ERROR CODE WHICH WAS
5482 034372 042704 177760 BIC #177760,R4 ; PASSED INTO THIS ROUTINE.
5483 ;*
5484 ; Determine the type of code which is to be reported.
5485 034376 012701 022355 MOV #EM9003,R1 ;SELECT MODEM STATUS CODE MESSAGE.
5486 034402 032702 000001 BIT #BIT0,R2 ;TEST THE MODEM STATUS INDICATION BIT.
5487 034406 001422 BEQ 41 ;GOTO REPORT ERROR IF MODEM STATUS CODE.
5488 034410 005237 005466 INC ERRNBR ;SELECT THE SELFTEST CODE ERROR NUMBER.
5489 034414 012701 022377 MOV #EM9004,R1 ;SELECT SELFTEST CODE MESSAGE.
5490 034420 012700 000300 MOV #300,R0 ;CHECK IF SELF-TEST OR BMP CODE.
5491 034424 040200 BIC R2,R0 ;TRY TO CLEAR BMP BITS.
5492 034426 001003 BNE 21 ;GO CHECK FOR SELFTEST CODE IF NOT BMP.
5493 034430 004737 035624 JSR PC,SAVBMP ;SAVE THE BMP CODE ON THE QUEUE.
5494 034434 000420 BR 601 ;EXIT THIS ROUTINE.
5495 034436 122702 000201 21: CMPB #201,R2 ;CHECK FOR SELF TEST NULL CODE.
5496 034442 001413 BEQ 61 ;EXIT ROUTINE IF NULL CODE FOUND.
5497 034444 122702 000203 CMPB #203,R2 ;CHECK FOR SKIP SELF TEST CODE.
5498 034450 001410 BEQ 61 ;EXIT ROUTINE IF SKIP SELF TEST CODE FOUND.
5499 034452 000400 BR 41 ;GO REPORT SELF TEST ERROR.
5500 ;*
5501 ; Report "UNEXPECTED xxxxx CODE FOUND IN RECEIVE CHAR FIFO."
5502 ;--
5503 034454 042702 177400 41: BIC #177400,R2 ;REMOVE UPPER BYTE OF CODE TO BE REPORTED.
5504 034460 004737 037074 JSR PC,TXROFF ;TURN OFF TX AND RX DURING ERROR REPORTING.
5505 034464 ERROR ;
5506 034466 104460 037130 JSR PC,TXRON ;TURN TX AND RX BACK ON.
5507 ;*
5508 ; Restore the initial error number.

```

GLOBAL SUBROUTINE

- REPC00 -

SEQ 0144

5509

5510 034472 010337 005466

5511

5512 034476

034476 004736

5513 034500 000207

:-

60:

60:

MOV R3,ERRNBR

PASS

RTS

PC

JSR

;RESTORE GPRS.
PC,8(SP).

;RETURN TO PREG05 SUBRT.

GLOBAL SUBROUTINE

REPSMR -

```

5515 .SBTTL GLOBAL SUBROUTINE REPSMR -
5516 ;* *****
5517 ;* - Report Error Summary Routine -
5518 ;* This subroutine reports an error summary for those lines which have
5519 ;* exceeded the number of individual errors to report for a single line
5520 ;* in a single test. This parameter can be specified by the operator if
5521 ;* he/she answers the Software Parameter Questions.
5522 ;*
5523 ;* INPUTS: ERCNTB - Label at base of line error counters table.
5524 ;*          ERRMSG - Address of primary error message.
5525 ;*          ERRNBR - Error number of errors in this routine.
5526 ;*          ERSMRF - "Report error summary for line" flags.
5527 ;*
5528 ;* OUTPUTS: ERRBLK - Address of error reporting routine (Destroyed).
5529 ;*          Summary messages may be printed at the operator console.
5530 ;*
5531 ;* CALLING SEQUENCE: JSR PC,REPSMR
5532 ;*
5533 ;* COMMENTS: If no lines have exceeded the maximum number of individual
5534 ;*            errors to report, no messages are printed by this routine.
5535 ;*            Error summaries in this routine are reported as errors.
5536 ;*            The contents of ERRBLK are destroyed.
5537 ;*
5538 ;* SUBORDINATE ROUTINES CALLED:
5539 ;* *****
5540
5541 034502 REPSMR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5542 034502 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5543 034506 005737 002650 TST ERSMRF ;CHECK THE "PRINT LINE ERROR SUMMARY" FLAGS.
5544 034512 001404 BEQ 60$ ;EXIT WITHOUT ACTION IF NO SUMMARY FLAGS SET.
5545 ;*
5546 ;* We have some error summaries to report.
5547 034514 012737 025750 005472 MOV #ER9004,ERRBLK ;SELECT ERROR REPORTING ROUTINE.
5548 ;*
5549 ;* Report
5550 ;* "ERROR SUMMARY REPORT FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:"
5551 ;*
5552 034522 ERROR
5553 034522 104460 TRAP C$ERROR
5554 034524 60$: PASS ;RESTORE GPRS.
5555 034524 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
5556 034526 000207 RTS PC

```

GLOBAL SUBROUTINE

- RESETT -

```

5557 .SBTTL GLOBAL SUBROUTINE RESETT -
5558 *****
5559 ;* - Reset Device Under Test -
5560 ;* This subroutine is used to reset the DUT to a known state.
5561 ;* If reset does not successfully complete, ie. time-out occurs, then
5562 ;* an abort test error message is reported.
5563 ;*
5564 ;* INPUTS: CSRA Contains the address of the CSR
5565 ;* TXBFCA - Contains address of DUT DMA Buffer Count register.
5566 ;* ERRTBL ERRTP,ERNBR,and ERRMSG set up correctly.
5567 ;*
5568 ;* OUTPUTS: The DUT performs its reset function into a known state.
5569 ;* CARRY - Clear indicates the test is to be aborted.
5570 ;* ERRBLK - value may be destroyed.
5571 ;* IESTAT - TX and RX interrupt flags are cleared.
5572 ;* TX and RX interrupt enable bits in the DUT's CSR are cleared.
5573 ;*
5574 ;* CALLING SEQUENCE: JSR PC,RESETT
5575 ;*
5576 ;* COMMENTS: This subroutine can report errors with numbers initial ERRNBR
5577 ;* This routine does not destroy the value of ERRNBR.
5578 ;*
5579 ;* SUBORDINATE ROUTINES CALLED: DELAY,MSLGET.
5580 ;*****
5581
5582 034530 RESETT:: SAVE ;S4 = CONTENTS OF GPRS R0 THRU R5.
5583 034530 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5584 034534 012702 000040 MOV #BIT05,R2 ;SET BIT MASK OF MASTER RESET BIT.
5585 ;*
5586 ; Test the state of the master reset bit in the CSR.
5587 ; If MR is set then wait for self-test to complete.
5588 ; If time out occurs, report the error and pass-out abort test indicator.
5589 ;-
5589 034540 013704 002300 MOV CSRA,R4 ;GET THE ADDRESS OF THE DUT'S CSR.
5590 034544 030214 BIT R2,(R4) ;CHECK STATE OF MASTER RESET BIT.
5591 034546 001406 BEQ 2$ ;DON'T DELAY IF MR IS ALREADY CLEAR.
5592 034550 005003 CLR R3 ;SET UP DESIRED STATE OF MASTER RESET BIT.
5593 034552 012701 004704 MOV #2500.,R1 ;PASS TIME OUT VALUE OF 2.5 SECONDS.
5594 034556 004737 031726 JSR PC,MSLGET ;WAIT FOR SELF-TEST TO COMPLETE, MR CLEAR.
5595 034562 103012 BCC 4$ ;GO REPORT ERROR IF TIMEOUT OCCURRED.
5596
5597 ;*
5598 ; Set Master Reset bit in CSR. Clear TX and RX enable bits, etc.
5599 ; Skip the selftest.
5600 ; Time-out of 2.5 secs, just in case the self-test executes.
5601 ;-
5602 034564 010277 145510 2$: MOV R2,@CSRA ;SET MASTER RESET BIT, DISABLE TX AND RX INTS.
5603 034570 004737 035740 JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST.
5604 ;*
5605 ; Set Self-test time-out of 2.5 seconds, and wait for M.R to clear.
5606 ; If Time out occurs, then report the fatal error and pass-out the abort
5607 ; test indicator.
5608 ;-
5609 034574 005003 CLR R3 ;SET UP DESIRED STATE OF MASTER RESET BIT.
5610 034576 012701 004704 MOV #2500.,R1 ;PASS TIME-OUT VALUE OF 2.5 SECONDS.
5611 034602 004737 031726 JSR PC,MSLGET ;WAIT FOR SELF-TEST TO COMPLETE, MR CLEAR.
5612 034606 103410 BCS 6$ ;SKIP ERROR REPORT IF MR CLEARED IN TIME

```

GLOBAL SUBROUTINE

- RESETT

```

5613
5614
5615
5616
5617 034610 012701 015226
5618 034614 012737 025034 005472
5619
5620
5621 034622
      034622 104460
5622 034624 000241
5623 034626 000403
5624
5625
5626
5627
5628 034630 005037 002374
5629 034634 000261
5630
5631 034636
      034636 004736
5632
5633 034640 000207
5634

```

```

;+
; Set up error message to report 'fatal error found during reset, test aborted .
; Indicate test is to be aborted by clearing the carry bit.
;-
4$:  MOV    #EM1601,R1      ;PASS ERROR MESSAGE TO REPORT.
      MOV    #ER1603,ERRBLK ;PASS ADDRESS OF ERROR HANDLING ROUTINE.
      ;REPORT ERROR "TIME-OUT OCCURRED WAITING FOR MASTER RESET TO CLEAR"
      ; "TEST ABORTED"
      ERROR
;
      >>>>> ERROR <<<<<
                                TRAP    C$ERROR
      CLC
      BR      60$           ;INDICATE TEST IS TO BE ABORTED.
                                ;EXIT THIS SUBROUTINE, ABORT TEST INDICATOR.
;+
; Clear TX and RX Interrupt enable status flags in IESTAT.
; Exit with continue test indicator set (ie,carry set).
;-
6$:  CLR     IESTAT         ;CLEAR TX AND RX INTERRUPT STATUS FLAGS.
      SEC
                                ;INDICATE SUCCESS, CONTINUE TEST.
;
60$:  PASS
                                ;RESTORE GPRS, PASS THE FOLLOWING INTACT:
                                PC,#(SP)+      ;RETURN TO PREGOS SUBRT.
                                ;CARRY BIT:IF CLEAR,INDICATES ABORT TEST.
      JSR
      RT      PC

```


GLOBAL SUBROUTINE

- ROLDAP -

```

5636 .SBTTL GLOBAL SUBROUTINE - ROLDAP -
5637 ;*****
5638 ;* - Rotate Left Data Pattern
5639 ;* This routine rotates the passed input data pattern left, without going
5640 ;* through the carry. The carry is initially set or cleared depending
5641 ;* upon the state of the MSB of the data pattern, before a ROL instruction
5642 ;* is executed.
5643 ;*
5644 ;* INPUTS: R2 - Contains the data pattern to be rotated
5645 ;*
5646 ;* OUTPUTS: R2 Contains the rotated data pattern
5647 ;*
5648 ;* CALLING SEQUENCE: JSR PC, ROLDAP
5649 ;*
5650 ;* COMMENTS:
5651 ;*
5652 ;* SUBORDINATE ROUTINES CALLED: NONE
5653 ;*****
5654
5655 034642 ROLDAP::SAVE
5656 034642 004537 005474 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
5657 034646 010202 MOV R2,R2 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5658 034650 005702 TST R2 ;SET PROCESSOR STATUS CODES
5659 034652 100402 BMI 2$ ;CHECK MSB
5660 034656 000401 CLC ;BRANCH IF SET
5661 034660 000261 BR 4$ ;CLEAR CARRY BIT IF MSB CLEAR
5662 034662 006102 2$: SEC ;
5663 034664 010266 000006 4$: ROL R2 ;SET CARRY IF MSB SET
5664 034670 004736 60$: PASS R2 ;ROTATE DATA PATTERN LEFT
5665 034672 000207 RTS PC ;RESTORE GPRS, EXCEPT
;PUT R2 IN STACK SLOT.
;RETURN TO PREG05 SUBRT.
;R2 - CONTAINS THE ROTATED DATA PATTERN

```

GLOBAL SUBROUTINE

- RRXNDN -

```

5667 .SBTTL GLOBAL SUBROUTINE RRXNDN -
5668 ;* *****
5669 ;* Report Reception Not Completed Routine -
5670 ;* This subroutine checks for lines which did not receive the complete
5671 ;* data pattern. If any are found, they are reported.
5672 ;*
5673 ;* INPUTS: R5 - Local active lines bit map.
5674 ;* DPLEN8 - Base of table of data pattern lengths.
5675 ;* ERRMSG - Address of primary error message for this routine.
5676 ;* ERRNBR - Error number of error reported in this routine.
5677 ;* RXCNTB - Label at base of the RX character counters table.
5678 ;* RXDNF - Reception done flags.
5679 ;*
5680 ;* OUTPUTS: ERRBLK - Address of the error reporting routine (Destroyed).
5681 ;* Messages may be printed at the operator console.
5682 ;*
5683 ;* CALLING SEQUENCE: JSR PC,RRXNDN
5684 ;*
5685 ;* COMMENTS: If no lines failed to complete their reception, no messages
5686 ;* are printed.
5687 ;*
5688 ;* SUBORDINATE ROUTINES CALLED: ER9005.
5689 ;* *****
5690
5691 034674 RRXNDN:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5692 034674 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5693 034700 010502 MOV R5,R2 ;GET COPY OF THE LOCAL ACTIVE LINES BIT MAP.
5694 034702 043702 002654 BIC RXDNF,R2 ;GET MAP OF ACTIVE LINES WITH RX DONE FLAG CLR.
5695 034706 001410 BEQ 60$ ;EXIT IF NO ACTIVE LINES HAVE RX DONE FLAG CLR.
5696
5697 ;* We have some "RX not completed" errors to report.
5698 034710 012737 026052 005472 MOV #ER9005,ERRBLK ;SELECT THE ERROR REPORTING ROUTINE.
5699 034716 012701 023043 MOV #EM9016,R1 ;INDICATE THAT WE ARE DEALING WITH RECEPTION.
5700 034722 012704 003712 MOV #RXCNTB,R4 ;PASS BASE OF RX CHAR COUNTERS TABLE TO ER9005.
5701
5702 ;* Report "SINGLE CHARACTER MODE TEST ERROR:"
5703 ;* "DATA PATTERN NOT COMPLETELY RECEIVED ON ALL LINES:"
5704 ;* ...
5705 ;*
5706 034726 ERROR
5707 034726 104460 TRAP C$ERROR
5708
5709 034730 60$: PASS ;RESTORE GPRS.
5710 034730 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
5710 034732 000207 RTS PC

```


GLOBAL SUBROUTINE

- RSTRPT -

```

5712 .SBTTL GLOBAL SUBROUTINE - RSTRPT -
5713 ;* *****
5714 ;* - Report any Reset Errors Routine
5715 ;* This routine determines if any error codes are among the diagnostic
5716 ;* codes reported placed in the DUT received character FIFO by the
5717 ;* Self-test. If any non BMP error codes are found, or if other errors
5718 ;* are encountered, appropriate errors are reported. Any BMP codes that
5719 ;* are found, are placed on the BMP code queue to be reported later.
5720 ;* This routine also purges the DUT FIFO looking for any characters
5721 ;* or modem status codes. If any are found, errors are reported.
5722 ;*
5723 ;* INPUTS: ERRMSG - Address of the primary error message.
5724 ;*          ERRNBR - Error number of first error reported by this routine.
5725 ;*          NUMLNS - Equated to the number of line on the DUT.
5726 ;*          RBUFA - Contains address of the DUT receiver FIFO.
5727 ;*
5728 ;* OUTPUTS: CARRY - Success flag (set if FIFO cleared successfully).
5729 ;*          ERRBLK - Address of the error report routine (Destroyed).
5730 ;*          Error messages can be printed at the operators console.
5731 ;*
5732 ;* CALLING SEQUENCE: JSR PC,RSTRPT
5733 ;*
5734 ;* COMMENTS: This subroutine can report errors with numbers Initial ERRNBR
5735 ;*          thru Initial ERRNBR+4.
5736 ;*          This routine does not destroy the value of ERRNBR.
5737 ;*
5738 ;* SUBORDINATE ROUTINES CALLED: ER0503,ER9007,ER9008,SAVBMP.
5739 ;* *****
5740
5741 034734 RSTRPT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5742 034734 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5743 ;*
5744 ; Read correct number (number of line on DUT) of chars from the FIFO.
5745 ; Verify that each char is a selftest success code.
5746 034740 005003 ;*
5747 034742 013705 005466 ; We expect a selftest code, but this FIFO slot is empty.
5748 034746 017702 145330 ;*
5749 034752 100412 ;*
5750 ;*
5751 ;*
5752 ;*
5753 034754 010537 005466 ;*
5754 034760 012701 023163 ;*
5755 034764 012737 026300 005472 ;*
5756 ;*
5757 ;*
5758 ;*
5759 ;*
5760 034772 ;*
5761 034772 104460 ;*
5762 ;*
5763 ;*
5764 034774 000261 ;*
5765 034776 000545 ;*
5766 ;*

```

GLOBAL SUBROUTINE

- RSTRPT -

```

5767      ; Determine if this is not a selftest code.
5768      ;
5769 035000 012700 070001      4$:      MOV      #70001,R0      ;GENERATE BIT MAP OF ANY CLEAR ERROR BITS OR
5770 035004 040200      BIC      R2,R0      ; BIT 0 WHICH ARE CLEAR.
5771 035006 001033      BNE      8$      ;GO TO REPORT ERROR IF THIS IS NOT A TEST CODE.
5772      ;+
5773      ; We have a test code (either BMP or selftest code).
5774      ; Determine what type of code we have.
5775      ;
5776 035010 032702 000200      BIT      #BIT7,R2      ;TEST ROM VERSION CODE INDICATOR BIT.
5777 035014 001443      BEQ      10$      ;SKIP ERRORS IF SELFTEST ROM VERSION CODE.
5778 035016 120227 000203      CMPB    R2,#203      ;CHECK IF SKIP SELF TEST CODE.
5779 035022 001440      BEQ      10$      ;SKIP ERROR REPORT IF SKIP SELF TEST CODE FOUND
5780 035024 120227 000201      CMPB    R2,#201      ;CHECK IF NULL CODE PRESENT.
5781 035030 001435      BEQ      10$      ;SKIP ERROR REPORT IF SELF TEST NULL CODE.
5782 035032 012700 000300      MOV      #300,R0      ;TEST CODE TYPE BITS FOR BOTH CODE
5783 035036 040200      BIC      R2,R0      ; TYPE BITS SET (INDICATING BMP CODE).
5784 035040 001003      BNE      6$      ;IF IT IS NOT A BMP CODE GO REPORT ERROR.
5785 035042 004737 035624      JSR      PC,SAVBMP      ;SAVE THE BMP CODE ON THE QUEUE.
5786 035046 000426      BR       10$      ;GO GET THE NEXT CHARACTER FROM THE FIFO.
5787      ;+
5788      ; We have a selftest error code.
5789      ;-
5790 035050 010537 005466      6$:      MOV      R5,ERRNBR      ;RESTORE ERROR NUMBER TO INITIAL VALUE.
5791 035054 005237 005466      INC      ERRNBR      ;CALCULATE INITIAL ERROR NUMBER PLUS 1.
5792 035060 012701 023210      MOV      #EM9020,R1      ;PASS ERROR MESSAGE INFO TO ER9008 ROUTINE.
5793 035064 012737 026356 005472      MOV      #ER9008,ERRBLK      ;SELECT PROPER ERROR REPORT ROUTINE.
5794      ;+
5795      ; Report error with number Initial ERRNBR + 1.
5796      ; "UNEXPECTED SELFTEST ERROR CODE FOR LINE nn IN FIFO AFTER RESET:"
5797      ;-
5798 035072      ERROR      ;          >>>>> ERROR <<<<<.
5799 035072 104460      TRAP      C$ERROR
5800 035074 000413      BR       10$      ;GO TO END OF LOOP.
5801      ;+
5802      ; We have a non-selftest code (either BMP code or data char).
5803      ;
5804 035076 010537 005466      8$:      MOV      R5,ERRNBR      ;RESTORE ERROR NUMBER TO INITIAL VALUE.
5805 035102 062737 000002 005466      ADD      #2,ERRNBR      ;CALCULATE INITIAL ERROR NUMBER PLUS 2.
5806 035110 012701 023173      MOV      #EM9019,R1      ;PASS ERROR MESSAGE INFO TO ER9007 ROUTINE.
5807 035114 012737 026300 005472      MOV      #ER9007,ERRBLK      ;SELECT PROPER ERROR REPORT ROUTINE.
5808      ;+
5809      ; Report error with number Initial ERRNBR + 2.
5810      ; "NON-SELFTEST CODE IN SELFTEST CODE FIFO SLOT FOR LINE nn AFTER RESET."
5811 035122      ERROR      ;          >>>>> ERROR <<<<<.
5812 035122 104460      TRAP      C$ERROR
5813      ;+
5814      ; End of loop, loop if not all chars have been read from the FIFO.
5815      ;
5816 035124 005203      10$:      INC      R3      ;SET CODE COUNTER FOR NEXT ITERATION OF LOOP.
5817 035126 020327 000010      CMP      R3,#NUMLNS      ;TEST FOR ALL CODES READ.
5818 035132 002705      BLT      2$      ;LOOP IF NOT CHARS READ FROM FIFO.
5819      ;+
5820      ; Purge the FIFO until DATA.VALID is clear or until too many chars are read.
5821      ;-
5821 035134 012704 000022      MOV      #18.,R4      ;INITIALIZE THE CHARACTER COUNTER.

```

GLOBAL SUBROUTINE

- RSTRPT -

```

5822 035140 010537 005466      MOV    R5,ERRNBR      ;GET INITIAL VALUE OF THE ERROR NUMBER.
5823 035144 062737 000003 005466  ADD    #3,ERRNBR      ;CALCULATE ERROR NUMBER OF NEXT ERROR.
5824 035152 012737 026356 005472  MOV    #ER9008,ERRBLK  ;SELECT PROPER ERROR REPORT ROUTINE.
5825 035160 017702 145116      MOV    @RBUFA,R2      ;READ A CHARACTER FROM THE DUT FIFO.
5826 035164 000261      SEC                      ;INDICATE SUCCESS IN CASE DATA.VALID IS CLEAR.
5827 035166 100051      BPL     60$          ;EXIT ROUTINE WITH SUCCESS IF DATA.VALID CLEAR.
5828
5829      ;+
5830      ; We have a character.
5831      ; Determine if character is a data character.
5832 035170 012700 070000      MOV    #70000,R0      ;TEST BITS 12 THRU 14 OF THE
5833 035174 040200      BIC    R2,R0          ; CODE READ FROM THE DUT FIFO.
5834 035176 001403      BEQ    14$          ;SKIP THIS ERROR IF CODE IS NOT A DATA CHAR.
5835
5836      ;+
5837      ; We have an unexpected data character: set up and go to report error.
5838 035200 012701 023234      MOV    #EM9022,R1      ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
5839 035204 000423      BR     22$          ;GO TO REPORT THIS ERROR.
5840
5841      ;+
5842      ; We have an unexpected code.
5843      ; Determine if the code is a modem status code.
5844 035206 032702 000001      BIT    #BIT0,R2      ;TEST MODEM STATUS INDICATOR BIT OF CODE.
5845 035212 001003      BNE    16$          ;SKIP THIS ERROR IF NOT MODEM STATUS CODE.
5846
5847      ;+
5848      ; We have a modem status code: set up and go to report error.
5849 035214 012701 023253      MOV    #EM9023,R1      ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
5850 035220 000415      BR     22$          ;GO TO REPORT THIS ERROR.
5851
5852      ;+
5853      ; We have an onboard test code.
5854      ; Determine if this code is a BMP code.
5855 035222 032702 000200      BIT    #BIT7,R2      ;TEST THE ROM VERSION BIT OF THE CODE.
5856 035226 001404      BEQ    18$          ;GOTO SET UP FOR SELFTEST CODE IF ROM VERSION.
5857 035230 012700 000300      MOV    #300,R0          ;
5858 035234 040200      BIC    R2,R0          ;TEST THE ERROR TYPE BITS OF THE CODE.
5859 035236 001403      BEQ    20$          ;SKIP THIS ERROR IF BMP CODE.
5860
5861      ;+
5862      ; We have a selftest code: set up and go to report error.
5863 035240 012701 023275      MOV    #EM9024,R1      ;SELECT ERROR MSG INFO FOR ER0808 ROUTINE.
5864 035244 000403      BR     22$          ;GO TO REPORT THIS ERROR.
5865
5866      ;+
5867      ; We have a BMP code: save it on the queue.
5868 035246 004737 035624      JSR    PC,SAVBMP      ;SAVE THE BMP CODE ON THE QUEUE.
5869 035252 000401      BR     24$          ;
5870
5871      ;+
5872      ; Report the error with error number of Initial ERRNBR + 3.
5873      ; "UNEXPECTED xxx xxxx FOR LINE nn IN FIFO AFTER RESET:"
5874 035254      22$: ERROR      ;          >>>>> ERROR <<<<<.
5875      ;          TRAP    C$ERROR
5876      ;+
5877      ; End of loop.
      ; Count the character we just received, and check for too many received.

```

GLOBAL SUBROUTINE

- RSTRPT -

```

5878
5879 035256 005304      24$: DEC R4      ;COUNT THIS CHARACTER.
5880 035260 001337      BNE 12$      ;LOOP IF NOT TOO MANY CHARACTERS PURGED.
5881
5882      ;+
5883      ; We read too many valid characters while trying to purge the FIFO.
5884      ; Report error and exit without success.
5885      ; "FIFO WILL NOT PURGE (DATA.VALID STUCK SET), REMAINDER OF TEST SKIPPED."
5886 035262 012701 023052      MOV #EM9017,R1      ;SELECT PROPER ERROR MESSAGE.
5887 035266 010537 005466      MOV R5,ERRNBR      ;GET INITIAL ERROR NUMBER.
5888 035272 062737 000004 005466      ADD #4,ERRNBR      ;CALCULATE INITIAL ERRNBR + 4.
5889 035300 012737 024644 005472      MOV #ER0503,ERRBLK      ;SELECT PROPER ERROR REPORT ROUTINE.
5890      ;PRINT ERROR REPORT.
5891 035306      ERROR      ;
5892 035306 104460      ;>>>> ERROR <<<<.
5893 035310 000241      CLC      ;CLEAR THE SUCCESS FLAG.      TRAP C$ERROR
5894 035312      60$: PASS      ;RESTORE GPRS,
5895 035314 000207      RTS PC      JSR PC,0(SP)+      ;RETURN TO PREG05 SUBRT.
      ; CARRY - SUCCESS FLAG (SET IF FIFO IS PURGED).

```

GLOBAL SUBROUTINE

RTXNDN -

```

5897 .SBTTL GLOBAL SUBROUTINE RTXNDN -
5898 ;* *****
5899 ;* - Report Transmission Not Completed Routine -
5900 ;* This subroutine checks for lines which did not transmit the complete
5901 ;* data pattern. If any are found, they are reported.
5902 ;*
5903 ;* INPUTS: R5 - Local active lines bit map.
5904 ;* DPLENB - Label at base of data pattern length table.
5905 ;* ERRMSG - Address of primary error message for this routine.
5906 ;* ERRNBR - Error number of error reported in this routine.
5907 ;* TXCNTB - Label at base of the TX character counters table.
5908 ;* TXDNF - Transmission done flags.
5909 ;*
5910 ;* OUTPUTS: ERRBLK - Address of the error reporting routine (Destroyed).
5911 ;* Messages may be printed at the operator console.
5912 ;*
5913 ;* CALLING SEQUENCE: JSR PC,RTXNDN
5914 ;*
5915 ;* COMMENTS: If no lines failed to complete their transmission, no messages
5916 ;* are printed.
5917 ;*
5918 ;* SUBORDINATE ROUTINES CALLED: ER9005.
5919 ;* - *****
5920
5921 035316 RTXNDN:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5922 035316 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5923 035322 010502 MOV R5,R2 ;GET COPY OF THE LOCAL ACTIVE LINES BIT MAP.
5924 035324 043702 002652 BIC TXDNF,R2 ;GET MAP OF ACTIVE LINES WITH TX DONE FLAG CLR.
5925 035330 001410 BEQ 60$ ;EXIT IF NO ACTIVE LINES HAVE TX DONE FLAG CLR.
5926 ;*
5927 ; We have some "TX not completed" errors to report.
5928 035332 012737 026052 005472 MOV #ER9005,ERRBLK ;SELECT THE ERROR REPORTING ROUTINE.
5929 035340 012701 023027 MOV #EM9015,R1 ;INDICATE WE ARE DEALING WITH TRANSMISSION.
5930 035344 012704 003652 MOV #TXCNTB,R4 ;PASS BASE OF TX CHAR COUNTERS TO TABLE ER0805.
5931 ;*
5932 ; Report "SINGLE CHARACTER MODE TEST ERROR:"
5933 ; "DATA PATTERN NOT COMPLETELY TRANSMITTED ON ALL LINES:"
5934 ; ...
5935 ; -
5936 ;
5937 035350 ERROR ;
5938 035350 104460 >>>> ERROR <<<<. TRAP C$ERROR
5939
5940 035352 60$: PASS ;RESTORE GPRS.
5941 035354 004736 000207 RTS PC JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- RXDSBL -

```

5943 .SBTTL GLOBAL SUBROUTINE - RXDSBL -
5944 ;+ *****
5945 ;* - Disable Receivers -
5946 ;* This subroutine is used to disable reception on selected lines by,
5947 ;* clearing the associated RX_ENABLE bit on the DUT.
5948 ;*
5949 ;* INPUTS: R5 - Bit's set correspond to lines on which to clear RX_ENABLE.
5950 ;* CSRA - Contains the address of the DUT CSR.
5951 ;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
5952 ;* NUMLNS - Equated to be the maximum number of lines available.
5953 ;* LNCTRA - Contains the address of the LNCTRL register.
5954 ;*
5955 ;* OUTPUTS: R5 - Bit's set indicate initial states of all RX_ENABLE bits.
5956 ;* LNCTRA - The state of the RX_ENABLE bit may be altered.
5957 ;* The contents of the IND_ADD_REG field in the CSR are destroyed.
5958 ;*
5959 ;* CALLING SEQUENCE: JSR PC,RXDSBL
5960 ;*
5961 ;* COMMENTS:
5962 ;*
5963 ;* SUBORDINATE ROUTINES CALLED: NONE.
5964 ;-- *****
5965
5966 035356 RXDSBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
5967 035356 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
5968 035362 010500 MOV R5,R0 ;COPY BIT MAP OF LINES TO DISABLE RECEPTION.
5969 035370 012701 000001 MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
5970 035374 012702 002310 MOV LNCTRA,R2 ;GET THE ADDRESS OF THE LNCTRL REGISTER.
5971 035400 013703 000010 MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER PLUS ONE.
5972 035404 005005 MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
5973 CLR R5 ;LOG POSSIBLE RX DISABLED ON ALL LINES.
5974
5975 ; Select every line in turn, and log the state of each RX_ENABLE bit.
5976 035406 010477 144666 2$: MOV R4,@CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
5977 035412 032712 000004 BIT #BIT2,(R2) ;CHECK STATE OF RX_ENABLE BIT ON SELECTED LINE.
5978 035416 001401 BEQ 4$ ;SKIP NEXT INSTRUCTION IF RX_ENABLE CLEAR.
5979 035420 050105 BIS R1,R5 ;LOG RX_ENABLE BIT SET FOR SELECTED LINE.
5980
5981 ; Clear RX_ENABLE on lines that have a corresponding bit set in the rx disable
5982 ; line bit map.
5983
5984 035422 030100 4$: BIT R1,R0 ;CHECK STATE OF DISABLE LINE BIT MAP.
5985 035424 001402 BEQ 6$ ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
5986 035426 042712 000004 BIC #BIT2,(R2) ;CLEAR RX_ENABLE BIT ON SELECTED LINE.
5987 035432 005204 6$: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
5988 035434 006301 ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
5989 035436 005303 DEC R3 ;DECREMENT LINE NUMBER.
5990 035440 001362 BNE 2$ ;LOOP TO CHECK NEXT LINE.
5991
5992 035442 60$: PASS R5 ;RESTORE GPRS, EXCEPT
5993 035442 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
5994 035446 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
5995 ;R5 - PREVIOUS STATES OF ALL RX_ENABLE BITS.
5996 RTS PC

```


GLOBAL SUBROUTINE

- RXENBL -

```

5996 .SBTTL GLOBAL SUBROUTINE - RXENBL -
5997 ;* *****
5998 ;* - Enable Receiver -
5999 ;* This subroutine is used to enable reception on selected lines by
6000 ;* setting the associated RX.ENABLE bit on the DUT.
6001 ;*
6002 ;* INPUTS: R5 - Bit's set correspond to lines on which to set RX.ENABLE.
6003 ;* CSRA - Contains the address of the DUT CSR.
6004 ;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
6005 ;* NUMLNS - Equated to be the maximum number of lines available.
6006 ;* LNCTRA - Contains the address of the LNCTRL register.
6007 ;*
6008 ;* OUTPUTS: R5 - Bit's set indicate previously disabled lines.
6009 ;* LNCTRA - The state of the RX.ENABLE bit may be altered.
6010 ;* The contents of the IND.ADD.REG field in the CSR are destroyed.
6011 ;*
6012 ;* CALLING SEQUENCE: JSR PC,RXENBL
6013 ;*
6014 ;* COMMENTS:
6015 ;*
6016 ;* SUBORDINATE ROUTINES CALLED: NONE.
6017 ;*
6018 ;* *****
6019 035452 RXENBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6020 035452 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6021 035456 010500 MOV R5,R0 ;COPY BIT MAP OF LINES TO ENABLE.
6022 035460 012701 000001 MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
6023 035470 012703 000010 MOV LNCTRA,R2 ;GET THE ADDRESS OF THE LNCTRL REGISTER.
6024 035474 013704 002374 MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER.
6025 035500 005005 MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
6026 CLR R5 ;CLEAR RX.ENABLE BIT LOG OF DISABLED LINES.
6027 ;*
6028 ; Select every line in turn, and log any RX.ENABLE bit that is clear.
6029 035502 010477 144572 2$: MOV R4,0CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
6030 035506 032712 000004 BIT #BIT2,(R2) ;CHECK STATE OF RX.ENABLE BIT ON SELECTED LINE.
6031 035512 001001 BNE 4$ ;SKIP NEXT INSTRUCTION IF RX.ENABLE SET.
6032 035514 050105 BIS R1,R5 ;LOG RX ENABLE BIT CLEAR FOR SELECTED LINE.
6033 ;*
6034 ; Set RX.ENABLE on lines that have a corresponding bit set in the rx enable
6035 ; line bit map.
6036 ;*
6037 035516 030100 4$: BIT R1,R0 ;CHECK STATE OF RX.ENABLE LINE BIT MAP.
6038 035520 001402 BEQ 6$ ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
6039 035522 052712 000004 BIS #BIT2,(R2) ;ENABLE RECEPTION ON SELECTED LINE.
6040 035526 005204 6$: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
6041 035530 006301 ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
6042 035532 005303 DEC R3 ;DECREMENT LINE NUMBER.
6043 035534 001362 BNE 2$ ;LOOP TO CHECK NEXT LINE.
6044 ;*
6045 035536 60$: PASS R5 ;RESTORE GPRS, EXCEPT
6046 035536 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
6047 035542 004736 JSR PC,0(SP) ;RETURN TO PREG05 SUBRT.
6048 035544 000207 RTS PC ;R5 - LINE BIT MAP CORRESPONDING TO THE
; PREVIOUS LINES THAT WERE DISABLED.

```

GLOBAL SUBROUTINE

RXIEO

```

6050          .SBTTL GLOBAL SUBROUTINE          RXIEO
6051          ;* *****
6052          ;*          - RECEIVER INTERRUPT DISABLE -
6053          ;*          This routine is used to disable receiver interrupts in the DHV11.
6054          ;*
6055          ;* INPUTS:      NONE.
6056          ;*
6057          ;* OUTPUTS:     The RX.INT.ENBL bit is cleared in the DUT CSR.
6058          ;*               IESTST -contains the updated status of the TX and RX interrupt
6059          ;*               enable bits.
6060          ;*
6061          ;* CALLING SEQUENCE:  JSR      PC,RXIEO
6062          ;*
6063          ;* COMMENTS:     The contents of the indirect address register field in
6064          ;*               the DUT CSR are destroyed.
6065          ;*
6066          ;* SUBORDINATE ROUTINES CALLED: NONE.
6067          ;* *****
6068 035546 010046  RXIEO::  MOV      RO,-(SP)          ;SAVE CONTENTS OF RO ON THE STACK.
6069 035550 106746      MFPS    -(SP)          ;SAVE PROCESSOR PRIORITY ON STACK.
6070 035552 106427 000340      MTPS   @PRI07      ;IGNORE ANY INTERRUPT THAT MAY BE GENERATED.
6071 035556 042737 137777 002374      BIC    @137777,IESTAT ;CLEAR RX.INT.ENBL BIT IN IESTAT.
6072 035564 013777 002374 144506      MOV     IESTAT,@CSRA ;DISABLE RX INTERRUPTS.
6073 035572 106426      MIPPS  (SP)+          ;ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
6074 035574 012600      MOV     (SP)+,RO      ;RESTORE RO.
6075 035576 000207      RTS      PC

```


GLOBAL SUBROUTINE

- RXIE1 -

```

6077      .SBTTL GLOBAL SUBROUTINE                      - RXIE1 -
6078      ;.. *****
6079      ;*                                     RECEIVER INTERRUPT ENABLE -
6080      ;*      This routine is used to enable receiver interrupts in the DHV11.
6081      ;*
6082      ;* INPUTS:      NONE.
6083      ;*
6084      ;* OUTPUTS:     The RX.INT.ENBL bit is set in the DUT CSR.
6085      ;*               IESTST -contains the updated status of the TX and RX interrupt
6086      ;*               enable bits.
6087      ;*
6088      ;* CALLING SEQUENCE:  JSR      PC,RXIE1
6089      ;*
6090      ;* COMMENTS:      The contents of the indirect address register field in
6091      ;*                  the DUT CSR are destroyed.
6092      ;*
6093      ;* SUBORDINATE ROUTINES CALLED: NONE.
6094      ;-- *****
6095
6096 035600 052737 000100 002374 RXIE1:: BIS    @BIT06, IESTAT ;SET RX.INT.ENBL BIT IN IESTAT.
6097 035606 042737 137677 002374      BIC    @137677, IESTAT ;CLEAR ALL OTHER BITS, EXCEPT TX AND RX I.E.
6098 035614 013777 002374 144456      MOV    IESTAT, @CSRA ;ENABLE RX INTERRUPTS.
6099 035622 000207                      RTS     PC

```

GLOBAL SUBROUTINE

- SAVBMP

```

6101 .SBTTL GLOBAL SUBROUTINE - SAVBMP -
6102 ;* *****
6103 ;* - Save BMP codes Routine -
6104 ;* This routine saves the parameter passed in, onto the BMP code queue
6105 ;* together with the number of the currently executing test.
6106 ;*
6107 ;* INPUTS: R2 - Contains the BMP code that is to be placed on the queue.
6108 ;* BMPCQP - Contains address of next location in the bmp queue.
6109 ;* BMPCQB - Label at base of the BMP code queue.
6110 ;* BMPCQE - Label of next location after the end of the BMP queue.
6111 ;* TSTNUM - Contains the number of the current test.
6112 ;*
6113 ;* OUTPUTS: BMPCQP - Incremented by 4.
6114 ;* The contents of the BMP code queue are updated.
6115 ;*
6116 ;* CALLING SEQUENCE: JSR PC,SAVBMP
6117 ;*
6118 ;* COMMENTS: If the overflow occurs then the last location will be
6119 ;* overwritten by any subsequent attempts to update the queue.
6120 ;*
6121 ;* SUBORDINATE ROUTINES CALLED: None.
6122 ;* - *****
6123
6124 035624 SAVBMP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6125 035624 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6126 035630 013704 002660 MOV BMPCQP,R4 ;GET THE POINTER TO THE NEXT LOCATION IN QUEUE.
6127 035634 113724 002364 MOV TSTNUM,(R4)+ ;SAVE THE CURRENT TEST NUMBER ON THE QUEUE.
6128 035640 005204 INC R4 ;INCREMENT THE POINTER TO GIVE AN EVEN ADDRESS.
6129 035642 042702 177400 BIC #177400,R2 ;CLEAR THE UNWANTED BITS FROM THE BMP CODE.
6130 035646 010224 MOV R2,(R4)+ ;SAVE THE BMP CODE ON THE QUEUE.
6131 035650 020427 003062 CMP R4,BMPCQE ;CHECK IF OVERFLOW WILL OCCUR THE NEXT TIME.
6132 035654 103402 BLO 2$ ;GO SAVE THE POINTER IF WE WILL NOT OVERFLOW.
6133 035656 162704 000004 SUB #4,R4 ;RESET THE POINTER TO THE LAST LOCATION IN QUE.
6134 035662 010437 002660 2$: MOV R4,BMPCQP ;SAVE THE POINTER.
6135 035666 004736 60$: PASS ;RESTORE GPRS.
6136 035670 000207 RTS PC JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

SETPAR -

```

6138 .SBTTL GLOBAL SUBROUTINE - SETPAR -
6139 ;* *****
6140 ;* SET TX AND CONTROL PARAMETERS -
6141 ;* This subroutine is used in the FIHAVL.TST.
6142 ;* It initialises the selected line to the following state:
6143 ;* Internal loopback, IAUTO enabled, LPR:38.4k, 8 bits/char, 2 stop,
6144 ;* odd parity.
6145 ;*
6146 ;* _NPUTS: R1 - Contains number of the line to be initialised.
6147 ;*
6148 ;* OUTPUTS: LNCTRL and LPR registers for the selected line are destroyed.
6149 ;*
6150 ;* CALLING SEQUENCE: JSR PC,SETPAR
6151 ;*
6152 ;* COMMENTS:
6153 ;*
6154 ;* SUBORDINATE ROUTINES CALLED: DELAY,WTWLNC,WTWLPR.
6155 ;* *****
6156
6157 035672 SETPAR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6158 035672 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6159 035676 004737 031646 JSR PC,LINBIT ;GET A BIT MAP FOR THIS LINE.
6160 035704 012700 000206 MOV R0,R5 ;COPY THE LINE BIT MAP.
6161 035710 004737 040250 MOV #206,R0 ;PASS INTERNAL LOPBCK, ENABLE RX AND IAUTO.
6162 035714 012700 177670 JSR PC,WTWLNC ;INITIALISE THE LINE CONTROL REGISTER.
6163 035720 004737 040324 MOV #177670,R0 ;PASS THE LPR CONTENTS.
6164 035724 012704 000012 JSR PC,WTWLPR ;SET THE LPR CONTENTS TO 38.4K BAUD.
6165 035730 004737 030370 MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI SECONDS.
6166 JSR PC,DELAY ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
6167 035734 60$: PASS ;RESTORE GPRS.
6168 035734 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
6168 035736 000207 RTS PC

```

GLOBAL SUBROUTINE

- SKPSTS -

```

6170 .SBTTL GLOBAL SUBROUTINE - SKPSTS -
6171 ;** *****
6172 ;* - Skip Selftest Routine -
6173 ;* This subroutine is used to skip the selftest after a DUT reset has been
6174 ;* initiated. It must be entered immediately after setting the DUT Master
6175 ;* Reset routine or after the execution of a bus reset (because of timing
6176 ;* considerations).
6177 ;*
6178 ;* INPUTS: CSRA - Contains address of the DUT CSR.
6179 ;* TXBFCA - Contains address of DUT DMA Buffer Count register.
6180 ;*
6181 ;* OUTPUTS: Skip selftest codes are written to the DUT registers.
6182 ;*
6183 ;* CALLING SEQUENCE: JSR PC,SKPSTS
6184 ;*
6185 ;* COMMENTS:
6186 ;*
6187 ;* SUBORDINATE ROUTINES CALLED: DELAY.
6188 ; *****
6189
6190 035740 SKPSTS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6191 035740 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6192 035744 012704 000012 MOV #10.,R4 ;PASS DELAY VALUE OF 10 MILLI-SECONDS.
6193 035750 004737 030370 JSR PC,DELAY ;DELAY FOR 10 MILLI-SECONDS.
6194 ;
6195 ; Write skip self-test code (52525) to all the indexed DUT Registers.
6196 035754 012701 000050 ;
6197 MOV #NUMLNS!BIT05,R1 ;FORM IND.ADR.REG FIELD (PLUS M.R. BIT) WORD.
6198 ;THE ABOVE INCLUSION OF THE M.R. BIT IS NECESSARY BECAUSE OF THE
6199 ; LACK OF A M.R. BIT WRITE LOCK-OUT ON THE DHV11-M.
6200 035760 012703 052525 MOV #52525,R3 ;INITIALISE THE SKIP SELF-TEST CODE.
6201 035764 005301 4$: DEC R1 ;SELECT THE NEXT SET OF DEVICE REGISTERS.
6202 035766 013704 002300 MOV CSRA,R4 ;GET THE ADDRESS OF THE CSR OF THE DUT.
6203 035772 010124 MOV R1,(R4)+ ;SELECT A BANK OF DUT REGISTERS.
6204 035774 010324 6$: MOV R3,(R4)+ ;WRITE THE CODE TO A DUT REGISTER.
6205 035776 020437 002316 CMP R4,TXBFCA ;COMPARE POINTER WITH LAST REGISTER ADDRESS.
6206 036002 103774 BLO 6$ ;LOOP IF NOT ALL REGS DONE IN THIS BANK.
6207 036004 032701 000017 BIT #17,R1 ;TEST FOR IND.ADR.REG FIELD DECREMENTED TO 0.
6208 036010 001365 BNE 4$ ;LOOP UNTIL ALL REGISTERS CONTAIN THE CODE.
6209 036012 036012 004736 60$: PASS ;RESTORE GPRS.
6210 036014 000207 RTS PC JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE - STPSW -

6212
6213
6214
6215
6216
6217
6218
6219
6220
6221
6222
6223
6224
6225
6226
6227
6228
6229 036016
036016 004537 005474
6230 036022 010146
6231 036024 012746 036032
6232 036030 000002
6233 036032
036032 004736
6234 036034 000207
6235
6236
6237
6238

```
.SBTTL GLOBAL SUBROUTINE      - STPSW -
;+ *****
;*                               - SET PROCESSOR STATUS WORD -
;*                               THIS ROUTINE SETS THE PSW TO THE CONTENTS OF R1.
;*
;* INPUTS:          R1 - CONTAINS THE NEW PSW SETTINGS
;*
;* OUTPUTS:         PSW - SET TO THE CONTENTS OF R1
;*
;* CALLING SEQUENCE:      JSR      PC,STPSW
;*
;* COMMENTS:          USED IN THE DMA ADDRESS TEST TO SET THE PROCESSOR
;*                    PRIORITY WITHOUT MAKING A CALL TO THE DRS.
;*
;* SUBROUTINES CALLED:    NONE.
;+ *****

STPSW:: SAVE
        MOV      R1,-(SP)      JSR      R5,PREG05      ;CALL REGISTER SAVE SUBRT.
        MOV      @ADDR,(SP)    ;PUSH THE NEW PSW CONTENTS ONTO THE STACK
        RTI      ;PUSH THE NEW PC VALUE ONTO THE STACK
        PASS     ;LOAD THE NEW PC AND PSW
ADDR:
        RTS      PC           JSR      PC,@(SP)+      ;RETURN TO PREG05 SUBRT.
        ;RETURN
```

GLOBAL SUBROUTINE - STPSW

SEQ 0163

```

6240
6241 .SBTTL GLOBAL SUBROUTINE - SWAPO -
6242 ;+ *****
6243 ;* - Swap GPRs With GPR Set 0 Routine -
6244 ;* This subroutine swaps the present contents of GPRs R1 thru R5 with
6245 ;* the contents of the number zero GPR save area. The contents of R0
6246 ;* are not altered by this subroutine.
6247 ;*
6248 ;* INPUTS: GPR contents R1 thru R5.
6249 ;* GPRS0B - Label at base of GPR save area number zero.
6250 ;*
6251 ;* OUTPUTS: R1 thru R5 contain the previous contents of GPR save area
6252 ;* zero words 1 thru 5 respectively.
6253 ;* GPRS0 - GPR save area 0 words 1 thru 5, contain previous
6254 ;* contents of GPRs R1 thru R5 respectively.
6255 ;*
6256 ;* CALLING SEQUENCE: JSR PC,SWAPO
6257 ;*
6258 ;* COMMENTS: The state of the CARRY flag is not altered by this routine.
6259 ;*
6260 ;* SUBORDINATE ROUTINES CALLED: None.
6261 ;-- *****
6262
6263 036036 010046 SWAPO:: MOV R0,-(SP) ;SAVE THE CONTENTS OF R0.
6264 ;+
6265 ; Load the stack into the GPRs.
6266 ;--
6267 036040 010146 MOV R1,-(SP) ;SAVE THE CONTENTS OF R1.
6268 036042 010246 MOV R2,-(SP) ;SAVE THE CONTENTS OF R2.
6269 036044 010346 MOV R3,-(SP) ;SAVE THE CONTENTS OF R3.
6270 036046 010446 MOV R4,-(SP) ;SAVE THE CONTENTS OF R4.
6271 036050 010546 MOV R5,-(SP) ;SAVE THE CONTENTS OF R5.
6272 ;+
6273 ; Load the GPRs from the GPR save area 0.
6274 ;--
6275 036052 012700 002634 MOV #GPRS0B,R0 ;GET THE BASE ADDRESS OF GPR SAVE AREA 0.
6276 036056 012001 MOV (R0)+,R1 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 1.
6277 036060 012002 MOV (R0)+,R2 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 2.
6278 036062 012003 MOV (R0)+,R3 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 3.
6279 036064 012004 MOV (R0)+,R4 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 4.
6280 036066 012005 MOV (R0)+,R5 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 5.
6281 ;+
6282 ; Load the GPR save area 0 from the stack.
6283 ;--
6284 036070 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 5 WITH SAVED R5.
6285 036072 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 4 WITH SAVED R4.
6286 036074 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 3 WITH SAVED R3.
6287 036076 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 2 WITH SAVED R2.
6288 036100 012640 MOV (SP)+,-(R0) ;LOAD GPR SAVE AREA 0 WORD 1 WITH SAVED R1.
6289
6290 036102 012600 MOV (SP)+,R0 ;RESTORE THE INITIAL VALUE OF R0.
6291
6292 036104 000207 RTS PC

```


GLOBAL SUBROUTINE

- TSABRT -

```

6294 .SBTTL GLOBAL SUBROUTINE TSABRT -
6295 ;* *****
6296 ;* - TEST ABORT ROUTINE -
6297 ;* This subroutine is used when a non-test related error has been found
6298 ;* during the execution of the current test.
6299 ;* It is used to inform the operator that the current test has been
6300 ;* aborted.
6301 ;*
6302 ;* INPUTS: ERRMSG - Contains the name of the current test.
6303 ;*          ERRNBR - Contains the correct error number.
6304 ;*          The remainder of the ERRTBL is correctly initialised.
6305 ;*
6306 ;* OUTPUTS: Messages are reported to the operator.
6307 ;*
6308 ;* CALLING SEQUENCE: JSR PC,TSABRT
6309 ;*
6310 ;* COMMENTS:
6311 ;*
6312 ;* SUBORDINATE ROUTINES CALLED: ER1603.
6313 ;* - *****
6314
6315 036106 TSABRT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
        036106 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6316 036112 012701 036130 MOV #2$,R1 ;PASS ADDRESS OF FIRST MESSAGE TO BE REPORTED.
6317 036116 012737 025034 005472 MOV #ER1603,ERRBLK ;SET-UP THE ERROR REPORTING ROUTINE.
6318 036124 ERROR ;
        036124 104460 ;>>>> ERROR <<<<.
6319 036126 000432 TRAP C$ERROR
6320 036130 040 116 117 2$: BR 60$ ;
        036133 116 055 122 .ASCIZ / NON-RELATED TEST ERROR FOUND DURING TEST EXECUTION/
        036136 105 114 101
        036141 124 105 104
        036144 040 124 105
        036147 123 124 040
        036152 105 122 122
        036155 117 122 040
        036160 106 117 125
        036163 116 104 040
        036166 104 125 122
        036171 111 116 107
        036174 040 124 105
        036177 123 124 040
        036202 105 130 105
        036205 103 125 124
        036210 111 117 116
        036213 000
6321 .EVEN
6322 036214 60$: PASS ;RESTORE GPRS.
        036214 004736 JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
6323 036216 000207 RTS PC

```

GLOBAL SUBROUTINE

- TXDATP -

```

6325 .SBTTL GLOBAL SUBROUTINE - TXDATP -
6326 ;++ *****
6327 ;* - TRANSMIT DATA PATTERN -
6328 ;* This subroutine is used in the FIHAVL.TST.
6329 ;* It transmits a specified number of data bytes on the specified line.
6330 ;*
6331 ;* INPUTS: R0 - Contains the number of data bytes to TX.
6332 ;* R1 - Contains line numb on which transmission is to take place.
6333 ;* BUFBAS to BUFMID contains a 256 byte data pattern.
6334 ;*
6335 ;* OUTPUTS: Data is sent out on the specified line.
6336 ;* Carry set = TX successful.
6337 ;*
6338 ;* CALLING SEQUENCE: TXDATP
6339 ;*
6340 ;* COMMENTS:
6341 ;*
6342 ;* SUBORDINATE ROUTINES CALLED: DODMA.
6343 ;-- *****
6344
6345 036220 TXDATP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
        036220 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6346 036224 010003 MOV R0,R3 ;PASS THE NUMBER OF CHARS TO TX.
6347 036226 012702 004012 MOV #BUFBAS,R2 ;PASS THE START OF THE DATA PATTERN TO TX.
6348 036232 004737 030624 JSR PC,DODMA ;TRANSMIT THE DATA PATTERN.
6349 036236 60$: PASS ;RESTORE GPRS.
        036236 004736 JSR PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
6350 036240 000207 RTS PC

```


GLOBAL SUBROUTINE

- TXDONE -

```

6352 .SBTTL GLOBAL SUBROUTINE - TXDONE -
6353 ;* *****
6354 ;* - TRANSMISSION DONE -
6355 ;* This subroutine is used in the transmission/reception tests to allow
6356 ;* time for transmission to complete on outstanding lines.
6357 ;*
6358 ;* INPUTS: ACTLNS - Contains bit map of all active lines.
6359 ;* TXDNF - TX done flags, set for lines that have sent all chars.
6360 ;* CHCNT - Table containing the number of chars to be TX'd.
6361 ;*
6362 ;*
6363 ;* OUTPUTS: Transmission interrupts are disabled.
6364 ;*
6365 ;* CALLING SEQUENCE: JSR PC,TXDONE
6366 ;*
6367 ;* COMMENTS:
6368 ;*
6369 ;* SUBORDINATE ROUTINES CALLED: MSLOOP,MUL16U.
6370 ;* *****
6371
6372 036242 TXDONE:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
036242 004537 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
005474
6373 ;*
6374 ; Check if all active lines have completed transmission.
6375 ; If any have not yet completed, determine the tx char count for a
6376 ; line that has outstanding characters to transmit. Using this value,
6377 ; calculate the time-out value needed at the currently selected baud rate.
6378 ;*
6379 036246 013703 002272 MOV ACTLNS,R3 ;GET THE ACTIVE LINE BIT MAP.
6380 036252 013702 002652 MOV TXDNF,R2 ;GET THE BIT MAP OF LINES THAT HAVE COMPLETED.
6381 036256 040203 BIC R2,R3 ;GENERATE A BIT MAP OF LINES THAT ARE STILL TX.
6382 036260 005703 TST R3 ;CHECK IF ALL LINES HAVE COMPLETED TX.
6383 036262 001427 BEQ 6$ ;GO DISABLE TX INTERRUPTS IF ALL DONE.
6384 ;*
6385 ; Find a line that has not completed transmission.
6386 ; Obtain the expected character count for that line (which is the same for
6387 ; all other lines with outstanding tx's).
6388 ; Calculate time-out value.
6389 ;*
6390 036264 005004 CLR R4 ;CLEAR LINE NUMBER COUNTER.
6391 036266 012702 000001 MOV #1,R2 ;SELECT BIT MAP FOR THE FIRST LINE.
6392 036272 030203 2$: BIT R2,R3 ;SEE IF THIS LINE HAS COMPLETED.
6393 036274 001003 BNE 4$ ;BRANCH IF THIS LINE HAS NOT COMPLETED TX.
6394 036276 006102 ROL R2 ;SHIFT THE LINE BIT MAP FOR THE NEXT LINE.
6395 036300 005204 INC R4 ;INCREMENT THE LINE NUMBER COUNTER.
6396 036302 000773 BR 2$ ;LOOP TO CHECK THE NEXT LINE.
6397 036304 006304 4$: ASL R4 ;LINE NUMBER X 2 TO OBTAIN OFFSET INTO TABLE.
6398 036306 016401 003612 MOV CHCNTB(R4),R1 ;GET THE EXPECTED NUMBER OF CHARS TO BE TX'D.
6399 036312 013702 002402 MOV RXTOUT,R2 ;GET THE CURRENT TIME-OUT VALUE FOR ONE CHAR.
6400 036316 004737 032056 JSR PC,MUL16U ;(NUMBER OF CHARS TO TX) X (TIME-OUT OF 1 CHAR)
6401 036322 006301 ASL R1 ;MULTIPLY DELAY TIME BY 2 TO GIVE A SAFE VALUE.
6402 ;*
6403 ; Wait for all outstanding transmissions to complete or time out.
6404 ; Disable all transmission interrupts.
6405 ;*
6406 036324 013702 002272 MOV ACTLNS,R2 ;PASS A BIT MAP OF THE BITS TO TEST.
6407 036330 010203 MOV R2,R3 ;PASS THE EXPECTED STATE OF THE TXDNF.

```

GLOBAL SUBROUTINE

- TXDONE -

6408	036332	012704	002652		MOV	#TXDONF,R4		;PASS THE ADDRESS OF THE WORD TO TEST.
6409	036336	004737	032042		JSR	PC,MSLOOP		;WAIT FOR TIME-OUT OF TX COMPLETION.
6410	036342	004737	036542	6\$:	JSR	PC,TXIE0		;DISABLE ALL TX INTERRUPTS.
6411								
6412	036346			60\$:	PASS			;RESTORE GPRS.
	036346	004736						
6413	036350	000207			RTS	PC	JSR	PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

GLOBAL SUBROUTINE

- TXDONE -

```

6415
6416 .SBTTL GLOBAL SUBROUTINE - TXDSBL -
6417 ;* *****
6418 ;* - Transmitter Disable -
6419 ;* This subroutine is used to disable transmission on selected lines by,
6420 ;* clearing the associated TX.ENABLE bit on the DUT.
6421 ;*
6422 ;* INPUTS: R5 - Bit's set correspond to lines on which to clear TX.ENABLE.
6423 ;* CSRA - Contains the address of the DUT CSR.
6424 ;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
6425 ;* NUMLNS - Equated to be the maximum number of lines available.
6426 ;* TXAD2A - Contains the address of the TBUFFAD2 register.
6427 ;*
6428 ;* OUTPUTS: R5 - Bit's set indicate the initial states of all TX.ENABLE bits.
6429 ;* TBUFFAD2 - The state of the TX.ENABLE bit may be altered.
6430 ;* The contents of the IND.ADD.REG field in the CSR are destroyed.
6431 ;*
6432 ;* CALLING SEQUENCE: JSR PC,TXDSBL
6433 ;*
6434 ;* COMMENTS:
6435 ;*
6436 ;* SUBORDINATE ROUTINES CALLED: NONE.
6437 ;*
6438 ;* *****
6439 TXDSBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6440 036352 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6441 036356 010500 MOV R5,R0 ;COPY BIT MAP OF LINES TO DISABLE TRANSMISSION.
6442 036360 012701 000001 MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
6443 036370 005202 MOV TXAD2A,R2 ;GET THE ADDRESS OF THE TBUFFAD2 REGISTER.
6444 036372 012703 000010 INC R2 ;GET THE ADDRESS OF THE MSBYTE OF TBUFFAD2 REG.
6445 036376 013704 002374 MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER PLUS ONE.
6446 036402 005005 MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
6447 CLR R5 ;LOG POSSIBLE TX DISABLED ON ALL LINES.
6448 ;*
6449 ; Select every line in turn, and log the state of each TX.ENABLE bit.
6450 036404 010477 143670 2$: MOV R4,#CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
6451 036410 105712 TSTB (R2) ;CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.
6452 036412 100001 BPL 4$ ;SKIP NEXT INSTRUCTION IF TX.ENABLE CLEAR.
6453 036414 050105 BIS R1,R5 ;LOG TX ENABLE BIT SET FOR SELECTED LINE.
6454 ;*
6455 ; Clear TX.ENABLE on lines that have a corresponding bit set in the tx disable
6456 ; line bit map.
6457 ;*
6458 036416 030100 4$: BIT R1,R0 ;CHECK STATE OF DISABLE LINE BIT MAP.
6459 036420 001402 BEQ 6$ ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
6460 036422 142712 000200 BICB #BIT7,(R2) ;CLEAR TX.ENABLE BIT ON SELECTED LINE.
6461 036426 005204 6$: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
6462 036430 006301 ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
6463 036432 005303 DEC R3 ;DECREMENT LINE NUMBER.
6464 036434 001363 BNE 2$ ;LOOP TO CHECK NEXT LINE.
6465 ;*
6466 036436 60$: PASS R5 ;RESTORE GPRS,EXCEPT
6467 036436 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK
6468 036442 004736 JSR PC,#(SP) ;RETURN TO PREGC
;R5 - PREVIOUS STATES OF ALL TX.ENABLE BITS.
RTS PC

```

GLOBAL SUBROUTINE

- TXENBL -

```

6470 .SBTTL GLOBAL SUBROUTINE - TXENBL -
6471 ;* *****
6472 ;* - Transmitter Enable -
6473 ;* This subroutine is used to enable transmission on selected lines by
6474 ;* setting the associated TX.ENABLE bit on the DUT.
6475 ;*
6476 ;* INPUTS: R5 - Bit's set correspond to lines on which to set TX.ENABLE.
6477 ;* CSRA - Contains the address of the DUT CSR.
6478 ;* IESTAT - Contains the state of TXIE and RXIE bits in the CSR.
6479 ;* NUMLNS - Equated to be the maximum number of lines available.
6480 ;* TXAD2A - Contains the address of the TBUFFAD2 register.
6481 ;*
6482 ;* OUTPUTS: R5 - Bit's set indicate previously disabled lines.
6483 ;* TBUFFAD2 - The state of the TX.ENABLE bit may be altered.
6484 ;* The contents of the IND.ADD.REG field in the CSR are destroyed.
6485 ;*
6486 ;* CALLING SEQUENCE: JSR PC,TXENBL
6487 ;*
6488 ;* COMMENTS:
6489 ;*
6490 ;* SUBORDINATE ROUTINES CALLED: NONE.
6491 ;* *****
6492
6493 TXENBL:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6494 036446 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6495 036452 010500 MOV R5,R0 ;COPY BIT MAP OF LINES TO ENABLE.
6496 036454 012701 000001 MOV #BIT0,R1 ;INITIALIZE THE SELECTED LINE BIT MASK.
6497 036460 013702 002314 MOV TXAD2A,R2 ;GET THE ADDRESS OF THE TBUFFAD2 REGISTER.
6498 036466 012703 000010 INC R2 ;GET THE ADDRESS OF THE MSBYTE OF TBUFFAD2 REG.
6499 036472 013704 002374 MOV #NUMLNS,R3 ;GET MAXIMUM LINE NUMBER.
6500 036476 005005 MOV IESTAT,R4 ;GET THE STATES OF THE INT ENABLE BITS.
6501 CLR R5 ;CLEAR TX.ENABLE BIT LOG OF DISABLED LINES.
6502
6503 ; Select every line in turn, and log any TX.ENABLE bit that is clear.
6504 036500 010477 143574 2: MOV R4,0CSRA ;WRITE TO DUT CSR TO SELECT LINE REGISTERS.
6505 036504 105712 TSTB (R2) ;CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.
6506 036506 100401 BMI 4: ;SKIP NEXT INSTRUCTION IF TX.ENABLE SET.
6507 036510 050105 BIS R1,R5 ;LOG TX ENABLE BIT CLEAR FOR SELECTED LINE.
6508
6509 ; Set TX.ENABLE on lines that have a corresponding bit set in the tx enable
6510 ; line bit map.
6511
6512 036512 030100 4: BIT R1,R0 ;CHECK STATE OF TX.ENABLE LINE BIT MAP.
6513 036514 001402 BEQ 6: ;BRANCH IF THIS LINE TO REMAIN UNALTERED.
6514 036516 152712 000200 BISB #BIT7,(R2) ;ENABLE TRANSMISSION ON SELECTED LINE.
6515 036522 005204 6: INC R4 ;PREPARE TO SELECT REGISTERS FOR NEXT LINE.
6516 036524 006301 ASL R1 ;SHIFT BIT MAP FOR NEXT LINE.
6517 036526 005303 DEC R3 ;DECREMENT LINE NUMBER.
6518 036530 001363 BNE 2: ;LOOP TO CHECK NEXT LINE.
6519
6520 036532 60: PASS R5 ;RESTORE GPRS, EXCEPT
6521 036532 010566 000014 MOV R5,R5SLOT(SP) ;PUT R5 IN STACK SLOT.
6522 036536 004736 JSR PC,0(SP) ;RETURN TO PREG05 SUBRT.
6523 036540 000207 RTS PC ;R5 - LINE BIT MAP CORRESPONDING TO THE
; PREVIOUS LINES THAT WERE DISABLED.

```

GLOBAL SUBROUTINE

- TXIEO -

```

6525      .SBTTL GLOBAL SUBROUTINE          - TXIEO -
6526      ;* *****
6527      ;* - TRANSMITTER INTERRUPT DISABLE -
6528      ;* This routine is used to disable transmitter interrupts in the DHV11.
6529      ;*
6530      ;* INPUTS:      NONE.
6531      ;*
6532      ;* OUTPUTS:     The TX.INT.ENBL bit is cleared in the DUT CSR.
6533      ;*               IESTST -contains the updated status of the TX and RX interrupt
6534      ;*               enable bits.
6535      ;*
6536      ;* CALLING SEQUENCE:  JSR      PC,TXIEO
6537      ;*
6538      ;* COMMENTS:      The contents of the indirect address register field in
6539      ;*                  the DUT CSR are destroyed.
6540      ;*
6541      ;* SUBORDINATE ROUTINES CALLED: NONE.
6542      ;* *****
6543 036542 010046      TXIEO::  MOV      RO,-(SP)      ;SAVE CONTENTS OF RO ON THE STACK.
6544 036544 106746      MFPS      -(SP)      ;SAVE CURRENT PROCESSOR PRIORITY ON THE STACK.
6545 036546 106427 000340      MTPS      @PRI07      ;IGNORE ANY INTERRUPTS THAT MAY BE GENERATED.
6546 036552 042737 177677 002374      BIC      @177677,IESTAT      ;CLEAR TX.INT.ENBL BIT IN IESTAT.
6547 036560 013777 002374 143512      MOV      IESTAT,@CSRA      ;DISABLE TX INTERRUPTS.
6548 036566 106426      MTPS      (SP)+      ;ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
6549 036570 012600      MOV      (SP)+,RO      ;RESTORE RO.
6550 036572 000207      RTS      PC

```

GLOBAL SUBROUTINE

- TXIE1 -

```

6552      .SBTTL GLOBAL SUBROUTINE          - TXIE1 -
6553      ;* *****
6554      ;*          - TRANSMITTER INTERRUPT ENABLE -
6555      ;*          This routine is used to enable transmitter interrupts in the DHV11.
6556      ;*
6557      ;* INPUTS:      NONE.
6558      ;*
6559      ;* OUTPUTS:     The TX.INT.ENBL bit is set in the DUT CSR.
6560      ;*               IESTST -contains the updated status of the TX and RX interrupt
6561      ;*               enable bits.
6562      ;*
6563      ;* CALLING SEQUENCE:  JSR      PC,TXIE1
6564      ;*
6565      ;* COMMENTS:     The contents of the indirect address register field in
6566      ;*               the DUT CSR are destroyed.
6567      ;*
6568      ;* SUBORDINATE ROUTINES CALLED: NONE.
6569      ;* -- *****
6570
6571 036574 052737 040000 002374 TXIE1:: BIS    #BIT14,IESTAT ;SET TX.INT.ENBL BIT IN IESTAT.
6572 036602 042737 137677 002374      BIC    #137677,IESTAT ;CLEAR ALL BITS EXCEPT TX RX I.E BITS.
6573 036610 013777 002374 143462      MOV    IESTAT,@CSRA ;DISAB_L TX INTERRUPTS.
6574 036616 000207          RTS     PC

```


GLOBAL SUBROUTINE

- TXRINI -

SEQ 0172

```

6576 .SBTTL GLOBAL SUBROUTINE - TXRINI -
6577 ;... *****
6578 ;*
6579 ;* - Transmit and Receive Initialization Routine -
6580 ;* This subroutine performs the initialization of the various pointers,
6581 ;* counters, and flags which are used during the transmission and
6582 ;* reception portion of a test. This initialization is performed on
6583 ;* the specified lines only, other line variables remain unchanged.
6584 ;*
6585 ;* INPUTS:
6586 ;* CHCNTB - Label at base of line character count table.
6587 ;* CHRTOT - Max # of chars to RX on lines already initialized.
6588 ;* DPENDB - Label at base of line data pattern end table.
6589 ;* DPLENB - Label at base of line data pattern length table.
6590 ;* EXCNTB - Label at base address of extra char counters table.
6591 ;* IESTAT - Present state of the RX.IE and TX.IE bits.
6592 ;* NUMLNS - Equated to number of lines on the DUT.
6593 ;* RXCNTB - Label at base address of RX character counters table.
6594 ;* RXPTRB - Label at base adr of "next RX char" pointers table.
6595 ;* TXCNTB - Label at base address of TX character counters table.
6596 ;* TXPTRB - Label at base adr of "next TX char" pointers table.
6597 ;* CBB - Label at base of TX/RX control block.
6598 ;* CB Contents TX/RX control block contains the following:
6599 ;* CBLPRA - DUT LPR contents.
6600 ;* CBLNCA - DUT LNCTRL contents.
6601 ;* CBDPAA - Address of beginning of data pattern.
6602 ;* CBDPLA - Length in bytes of data pattern.
6603 ;* CBDPNA - Number of data patterns to transmit.
6604 ;* CBMAPA - Bit map of lines to be initialized.
6605 ;* CBLPBA - Type of loopback to be used for test.
6606 ;* CBOFSA - Amount to offset each TX start in the data pat.
6607 ;* TXRXLB - Label at base of TX/RX line association table.
6608 ;*
6609 ;* OUTPUTS:
6610 ;* CHCNT - Table of number of line TX characters (Initialized).
6611 ;* CHRTOT - Maximum number of chars to receive (2 * pat length).
6612 ;* DPEND - Table of data pattern ends (Initialized).
6613 ;* DPLEN - Table of data pattern lengths (Initialized).
6614 ;* DUT LNCTRL - Line control registers (Initialized).
6615 ;* DUT LPR - Line parameter registers (Initialized).
6616 ;* EXCNT - Table of extra RX char counts (Clred, selected lines).
6617 ;* RXCNT - Table of RX character counts (Clred, selected lines).
6618 ;* RXDONF - "Reception Done" flags (Cleared for selected lines).
6619 ;* RXPTR - Table of receive pointers (Initialized).
6620 ;* TXCNT - Table of TX character counters (Clred, selected lines).
6621 ;* TXDONF - "Transmission Done" flags (Clred for selected lines).
6622 ;* TXPTR - Table of transmit pointers (Initialized).
6623 ;* TXRXL - TX/RX line association table (Initialized).
6624 ;*
6625 ;* CALLING SEQUENCE: JSR PC,TXRINI
6626 ;*
6627 ;* COMMENTS: If the calculation of the CHRTOT value (2 times the data
6628 ;* pattern length) results in a number greater than 64K then
6629 ;* CHRTOT is initialized to 64K - 1.
6630 ;* This routine will not force internal loopback based on the
6631 ;* loopback type in CBLPBA. The user must set up CBLNCA correctly
6632 ;* to get internal loopback.
6633 ;*
6634 ;* SUBORDINATE ROUTINES CALLED: WTWLNC,WTWLPR,
6635 ;... *****

```

GLOBAL SUBROUTINE

- TXRINI -

```

6633 036620      TXPINI:: SAVE
      036620 004537 005474      JSR      R5,PREG05      ;SAVE CONTENTS OF GPRS R0 THRU R5.
6634      ;+
6635      ; Set up the LPR and LNCTRL registers as specified in the TX/RX Control Block.
6636      ;+
6637 036624 013705 003304      MOV      CBMAPA,R5      ;GET THE BIT MAP OF SELECTED LINES.
6638 036630 013700 003274      MOV      CBLNCA,R0      ;GET THE NEW LNCTRL CONTENTS.
6639 036634 023727 003306 000001      CMP      CBLPBA,#1      ;CHECK IF INTERNAL LOOPBACK HAS BEEN SELECTED.
6640 036642 001002      BNE      2$      ;SKIP SETTING INT. LOPBCK IN MAINTENANCE FIELD.
6641 036644 052700 000200      BIS      #200,R0      ;SET INTERNAL LOOPBACK IN MAINTENANCE FIELD.
6642 036650 004737 040250 2$:      JSR      PC,WTWLNC      ;SET UP THE LNCTRL REGS FOR SELECTED LINES.
6643 036654 013700 003272      MOV      CBLPRA,R0      ;GET THE NEW LPR CONTENTS.
6644 036660 004737 040324      JSR      PC,WTWLPR      ;SET UP THE LPR REGISTERS FOR SELECTED LINES.
6645 036664 004737 036446      JSR      PC,TXENBL      ;ENABLE TX FOR ALL SELECTED LINES.
6646      ;+
6647      ; Set up and begin loop which handles one line per iteration.
6648      ;+
6649 036670 005004      CLR      R4      ;CLEAR THE LINE OFFSET.
6650 036672 013705 003276      MOV      CBDPAA,R5      ;INITIALIZE THE TX START ADDRESS VALUE.
6651 036676 013703 003300      MOV      CBDPLA,R3      ;GET THE LENGTH OF THE DATA PATTERN.
6652 036702 060503      ADD      R5,R3      ;CALCULATE END ADDRESS OF THE DATA PATTERN.
6653 036704 036437 002534 003304 4$:      BIT      BITTBL(R4),CBMAPA ;CHECK IF THIS LINE IS SELECTED FOR INIT.
6654 036712 001452      BEQ      12$      ;SKIP SET UP IF LINE IS NOT SELECTED.
6655      ;+
6656      ; This line is selected for initialization.
6657      ; Set up proper entry in number of chars to TX and RX table.
6658      ; Include char count on this line in max allowable char total for all lines.
6659      ;+
6660 036714 013701 003300      MOV      CBDPLA,R1      ;GET THE LENGTH OF THIS LINE'S DATA PATTERN.
6661 036720 013702 003302      MOV      CBDPNA,R2      ;GET THE NUMBER OF PATTERNS TO TX AND RX.
6662 036724 004737 032056      JSR      PC,MUL16U      ;CALCULATE THE TOTAL NUMBER OF CHARS TO TX/RX.
6663 036730 010164 003612      MOV      R1,CHCNTB(R4) ;SET UP THE NUMBER OF TX/RX CHARS FOR LINE.
6664 036734 060137 002646      ADD      R1,CHRTOT      ;ADD TWICE THE NUMBER OF CHARACTERS TO TX/RX
6665 036740 103403      BCS      6$      ; ON THIS LINE TO THE TOTAL NUMBER OF CHARS
6666 036742 060137 002646      ADD      R1,CHRTOT      ; WHICH WE WILL ALLOW TO BE RECEIVED ON
6667 036746 103003      BCC      8$      ; ALL LINES.
6668 036750 012737 177777 002646 6$:      MOV      #-1,CHRTOT      ; SET MAX CHAR TOTAL TO -1 IF OVERFLOW.
6669 036756      8$:
6670      ;+
6671      ; Set up the data pattern end and length for this line.
6672      ;+
6673 036756 013764 003300 003352      MOV      CBDPLA,DLENB(R4) ;SET UP TX DATA PATTERN LENGTH FOR THIS LINE.
6674 036764 010364 003312      MOV      R3,DPENDB(R4) ;SET UP TX DATA PAT END ADDRESS FOR THIS LINE.
6675      ;+
6676      ; Set up the TX counter and character pointer for this line.
6677      ;+
6678 036770 005064 003652      CLR      TXCNTB(R4) ;CLEAR THE TX COUNTER FOR THIS LINE.
6679 036774 010564 003512      MOV      R5,TXPTRB(R4) ;SET UP THE TX CHAR POINTER FOR THIS LINE.
6680      ;+
6681      ; Set up the TX/RX line association offset table entry for this line.
6682      ;+
6683 037000 010402      MOV      R4,R2      ;SELECT LINE OFFSET FOR NON-STAGGERED LPBK.
6684 037002 023727 003306 000002      CMP      CBLPBA,#2      ;TEST FOR STAGGERED LOOPBACK.
6685 037010 001003      BNE      10$      ;SKIP SETTING STAGGERED LPBK IF NOT.
6686 037012 006202      ASR      R2      ;FORM BYTE OFFSET INTO TABLE FROM TX LINE #.
6687 037014 116202 005444      MOV      STGTRB(R2),R2 ;GET THE RX LINE CORRESPONDING WITH TX LINE.
6688 037020 010264 005364 10$:      MOV      R2,TXRXLB(R4) ;LOAD TX TABLE ENTRY WITH RX LINE OFFSET.

```


GLOBAL SUBROUTINE

- TXRINI -

```

6689
6690
6691
6692
6693 037024 005062 003712
6694 037030 005062 003412
6695 037034 010562 003552
6696
6697
6698
6699 037040 063705 003310
6700 037044 020503
6701 037046 103403
6702 037050 163705 003300
6703 037054 000773
6704
6705
6706
6707 037056 005204
6708 037060 005204
6709
6710
6711
6712 037062 020427 000020
6713 037066 002706
6714
6715 037070
        037070 004736
6716 037072 000207

```

```

;+
; Set up the RX counters and character pointer for the RX line which
; is associated with this TX line.
;-
        CLR    RXCNTB(R2)    ;CLEAR THE RX COUNTER FOR THIS RX LINE.
        CLR    EXCNTB(R2)    ;CLEAR THE EXTRA CHAR COUNTER FOR THIS RX LINE.
        MOV     R5,RXPTRB(R2) ;SET UP THE RX CHAR POINTER FOR THIS RX LINE.
;+
; Update the TX start pointer in preparation for the next line.
;-
12$:    ADD     CBOFSA,R5      ;ADD THE TX OFFSET TO THE TX START POINTER.
14$:    CMP     R5,R3          ;COMPARE TX START WITH END OF DATA PATTERN.
        BLO     16$           ;SKIP WRAPAROUND IF START IS BEFORE PAT END.
        SUB     CBDPLA,R5     ;SUBTRACT DATA PATTERN LENGTH FROM START.
        BR      14$          ;LOOP UNTIL START IS WITHIN DATA PATTERN.
;+
; Update the TX line number offset to the next line.
;-
16$:    INC     R4
        INC     R4
;+
; Test for done handling all possible lines on the device.
;-
        CMP     R4,#NUMLNS*2  ;COMPARE OFFSET WITH 2 TIMES MAX # OF LINES.
        BLT     4$           ;LOOP IF NOT ALL LINES DONE.
60$:    PASS
        JSR     PC            ;RESTORE GPRS.
        RTS     PC            ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- TXROFF -

```

6718 .SBTTL GLOBAL SUBROUTINE - TXROFF -
6719 ;** *****
6720 ;* - Turn TX and RX off Routine -
6721 ;* This subroutine is used to turn off DUT transmission and reception.
6722 ;* This routine achieves this by boosting processor priority to 5 to
6723 ;* avoid RX interrupts and by clearing all the DUT TX.ENABLE bits to
6724 ;* halt TX (either DMA or single character TX). The states of the
6725 ;* TX.ENABLE bits and the processor priority are saved for restoration
6726 ;* when TX and RX are re-enabled.
6727 ;*
6728 ;* INPUTS: MAPLNS - Bit map of all possible lines on the DUT.
6729 ;*
6730 ;* OUTPUTS: SAVPRI - Saved processor priority.
6731 ;* SAVTEN - Bit map of TX.ENABLE bits (Bit set if TX.ENABLE was set).
6732 ;*
6733 ;* CALLING SEQUENCE: JSR PC,TXROFF
6734 ;*
6735 ;* COMMENTS:
6736 ;*
6737 ;* SUBORDINATE ROUTINES CALLED: TXDSBL.
6738 ;-- *****
6739
6740 037074 TXROFF:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
        037074 004537 005474 R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6741 037100 MFPS SAVPRI JSR ;GET THE PRESENT PROCESSOR PRIORITY.
        037100 106737 002406 MTPS #PRI05 ;DISABLE DUT INTERRUPTS.
6742 037104 MOV #MAPLNS,R5 ;PREPARE TO DISABLE TX ON ALL DUT LINES.
        037104 106427 000240 JSR PC,TXDSBL ;CLEAR ALL DUT TX.ENABLE BITS.
6743 037110 MOV R5,SAVTEN ;PRESERVE THE PREVIOUS TX.ENABLE BIT STATES.
        037110 012705 000377 JSR PC,TXDSBL ;RESTORE GPRS.
6744 037114 004737 036352 60$: PASS ;RETURN TO PREG05 SUBRT.
        037114 010537 002404 RTS PC JSR PC,@(SP)+
6745 037120 037124 004736
6746 037124 000207
6747 037126 000207

```

GLOBAL SUBROUTINE

- TXRON -

```

6749 .SBTTL GLOBAL SUBROUTINE - TXRON -
6750 ;* *****
6751 ;* - Turn TX and RX on Routine -
6752 ;* This subroutine is used to turn on DUT transmission and reception.
6753 ;* This routine restores the DUT TX.ENABLE bits and the processor priority
6754 ;* to the states saved by the TXROFF routine.
6755 ;*
6756 ;* INPUTS: SAVPRI - Saved processor priority.
6757 ;* SAVTEN - Bit map of TX.ENBL bits (Bit set if TX.ENBL was set).
6758 ;*
6759 ;* OUTPUTS: DUT TX.ENABLE bits - Set to specified states.
6760 ;* PROCESSOR PRIORITY - Set to specified priority.
6761 ;*
6762 ;* CALLING SEQUENCE: JSR PC,TXRON
6763 ;*
6764 ;* COMMENTS:
6765 ;*
6766 ;* SUBORDINATE ROUTINES CALLED: TXENBL.
6767 ;* *****
6768
6769 037130 TXRON:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
        037130 004537 005474 R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6770 037134 013705 002404 MOV SAVTEN,R5 ;GET THE SAVED STATES OF THE TX.ENABLE BITS.
6771 037140 004737 036446 JSR PC,TXENBL ;SET THE SPECIFIED TX.ENABLE BITS.
6772 037144 106437 002406 MTPS SAVPRI ;RESTORE THE PROCESSOR PRIORITY.
6773 037150 037150 004736 60: PASS ;RESTORE GPRS.
        037152 000207 RTS PC JSR PC,0(SP) ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- TXRREP -

```

6776 .SRTTL GLOBAL SUBROUTINE - TXRREP -
6777 ;** *****
6778 ;* - Report Final TX/RX Errors Routine -
6779 ;* This subroutine reports errors which are found after the completion
6780 ;* of the TX, RX, and verification of data patterns. It reports errors
6781 ;* dealing with incomplete TX or RX and with DMA_START bits.
6782 ;*
6783 ;* INPUTS: ACTLNS - Bit map of active DUT lines.
6784 ;* DPLENB - Label at base of the data pattern lengths table.
6785 ;* ERRMSG - Address of primary error message for this routine.
6786 ;* ERRNBR - Error number of error reported in this routine.
6787 ;* RXCNTB - Label at base of the RX character counters table.
6788 ;* RXDNF - Reception done flags.
6789 ;* TXCNTB - Label at base of the TX character counters table.
6790 ;* TXDNF - Transmission done flags.
6791 ;* TXINTF - Contains bit map of lines with DMA_START bit errors.
6792 ;*
6793 ;* OUTPUTS: CARRY FLAG - Restored to its entering value.
6794 ;* ERRBLK - Address of the error reporting routine (Destroyed).
6795 ;* Messages may be printed at the operator console.
6796 ;*
6797 ;* CALLING SEQUENCE: JSR PC,TXRREP
6798 ;*
6799 ;* COMMENTS: This routine reports errors at Initial ERRNBR thru
6800 ;* Initial ERRNBR+2.
6801 ;* If no lines failed to complete their reception or failed to
6802 ;* complete their transmission or had DMA_START bit errors
6803 ;* then no messages are printed.
6804 ;*
6805 ;* SUBORDINATE ROUTINES CALLED: CONMAP,ER9005,ER9102,RDMAST,RRXNDN,RTXNDN.
6806 ;-- *****
6807
6808 037154 004537 005474 TXRREP:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6809 037154 006003 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6810 037162 013704 ROR R3 ;ROTATE CARRY INTO GPR TO SAVE CARRY STATE.
6811 037166 013705 MOV ERRNBR,R4 ;SAVE THE INITIAL ERROR NUMBER VALUE.
6812 037172 004737 MOV ACTLNS,R5 ;GET THE ACTIVE LINES BIT MAP.
6813 037176 005237 JSR PC,RDMAST ;REPORT ANY DMA_START BIT ERRORS.
6814 037202 004737 INC ERRNBR ;SELECT INITIAL ERROR NUMBER + 1.
6815 037206 005237 JSR PC,RTXNDN ;REPORT TX NOT COMPLETE IF NECESSARY.
6816 037212 004737 INC ERRNBR ;SELECT INITIAL ERROR NUMBER + 2.
6817 037216 004737 JSR PC,CONMAP ;GENERATE AN ASSOCIATED LINE BIT MAP.
6818 037222 010437 JSR PC,RRXNDN ;REPORT RX NOT COMPLETE IF NECESSARY.
6819 037222 005466 MOV R4,ERRNBR ;RESTORE THE INITIAL ERROR NUMBER VALUE.
6820 037226 006103 ROL R3 ;ROTATE SAVED CARRY STATE BACK INTO CARRY.
6821 037230 604: PASS ;RESTORE GPRS, THIS ROUTINE PRESERVES THE
6822 037232 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
6822 037232 000207 RTS PC ; INITIAL CARRY STATE.

```

GLOBAL SUBROUTINE

- UNSDIV -

```

6824 .SBTTL GLOBAL SUBROUTINE - UNSDIV -
6825 ;* *****
6826 ;* - Unsigned Divide Routine -
6827 ;* This subroutine is used to divide a 32 bit unsigned dividend by a
6828 ;* 16 bit unsigned divisor giving a 16 bit quotient. All numbers are
6829 ;* considered to be unsigned. A success flag is not set on return if
6830 ;* the quotient was too big to be contained in 16 bits.
6831 ;*
6832 ;* INPUTS: R1 - The divisor, unsigned, 16 bits.
6833 ;* R2 - Most significant word of the dividend, unsigned, 16 bits.
6834 ;* R3 - Least significant word of the dividend, unsigned, 16 bits.
6835 ;*
6836 ;* OUTPUTS: R1 - Quotient, unsigned, 16 bits (177777 if overflow).
6837 ;* CARRY - Success flag, set if complete quotient fits in 16 bits.
6838 ;*
6839 ;* CALLING SEQUENCE: JSR PC,UNSDIV
6840 ;*
6841 ;* COMMENTS: If the divisor is 0 the quotient is returned as all ones
6842 ;* (177777) and the carry is clear regardless of the dividend.
6843 ;*
6844 ;* SUBORDINATE ROUTINES CALLED: None.
6845 ;* -- *****
6846
6847 037234 UNSDIV:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
037234 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6848
6849 ;*
6850 ; Check for quotient greater than 16 bits condition.
6851 ;*
6851 037240 010204 MOV R2,R4 ;GET MSW OF DIVIDEND FOR SUBTRACT.
6852 037242 160104 SUB R1,R4 ;SUBTRACT DIVISOR FROM MSW OF DIVIDEND.
6853 037244 103403 BCS 2$ ;IF IT DIDN'T GO, WE HAVE QUOTIENT < 16 BITS.
6854 037246 012701 177777 MOV #1,R1 ;SET QUOTIENT TO ALL ONES (177777).
6855 037252 000442 BR 60$ ;EXIT WITH CARRY CLEAR.
6856
6857 ;*
6858 ; Set up counters and various working GPRs.
6859 037254 005004 2$: CLR R4 ;CLEAR THE LSW OF THE DIVISOR.
6860 037256 000241 CLC ;CLEAR CARRY FOR THE SHIFT OF THE DIVISOR.
6861 037260 006001 ROR R1 ; DIVISOR BY
6862 037262 006004 ROR R4 ; 2(UNSIGNED)
6863 037264 012700 000020 MOV #16,R0 ;SET UP INITIAL SHIFT COUNT TO 16.
6864
6865 ;*
6866 ; The subtract and shift loop.
6867 037270 010246 4$: MOV R2,-(SP) ;SAVE MSWORD OF DIVIDEND.
6868 037272 010346 MOV R3,-(SP) ;SAVE LSWORD OF DIVIDEND.
6869 037274 160403 SUB R4,R3 ;LSWORD DIVIDEND - LSWORD OF DIVISOR.
6870 037276 005602 SBC R2 ;MSWORD DIVIDEND - BORROW
6871 037300 103402 BCS 6$ ;IF BORROW FROM BORROW SUBTRACT, IT DIDN'T GO.
6872 037302 160102 SUB R1,R2 ;MSWORD DIVIDEND - MSWORD OF DIVISOR.
6873 037304 130003 BCC 8$ ;IF NO BORROW, IT WENT, CARRY IS CLEAR.
6874
6875 ;*
6876 ; It didn't go, so we shift a 1 into the quotient (complemented later).
6877 ; Carry is set.
6878 037306 012603 6$: MOV (SP)+,R3 ;RESTORE LSWORD OF DIVIDEND.
6879 037310 012602 MOV (SP)+,R2 ;RESTORE MSWORD OF DIVIDEND.

```

GLOBAL SUBROUTINE

- UNSDIV -

```

6880 037312 000401      BR      10$      ;GOTO SHIFT 1 INTO THE QUOTIENT.
6881                    ;+
6882                    ; It went, so we restore the stack and shift a 0 into quotient (will be
6883                    ; complemented later).  Carry is clear.
6884                    ;+
6885 037314 012626      8$:  MOV      (SP)+,(SP)+      ;POP THE SAVED DIVIDEND OFF OF THE STACK.
6886                    ;+
6887                    ; Shift the result of the subtract attempt into the quotient shift reg.
6888                    ;+
6889 037316 006105      10$:  ROL      R5              ;SHIFT NEXT BIT INTO THE INVERTED QUOTIENT.
6890 037320 000241      CLC                      ;DIVIDE THE
6891 037322 006001      ROR      R1              ; DEVISOR BY
6892 037324 006004      ROR      R4              ; 2 (UNSIGNED).
6893 037326 005300      DEC      R0              ;COUNT THIS SHIFT AND SUBTRACT.
6894 037330 001357      BNE      4$              ;LOOP FOR ANOTHER SHIFT & SUB IF NOT DONE.
6895 037332 005105      COM      R5              ;GET QUOTIENT FROM INVERTED QUOTIENT.
6896                    ;+
6897                    ; Now we either round up or leave quotient alone.
6898                    ;+
6899 037334 000241      CLC                      ;CLEAR THE CARRY FOR THE SHIFT OF THE DIVIDEND.
6900 037336 006103      ROL      R3              ;MULTIPLY LSWORD OF DIVIDEND BY 2, MSWORD IS 0.
6901 037340 103402      BCS      12$             ;IF CARRY FROM SHIFT, ROUND UP.
6902 037342 160403      SUB      R4,R3           ;SUBTRACT DIVISOR FROM DIVIDEND.
6903 037344 103403      BCS      14$             ;IF BORROW, DON'T ROUND UP.
6904                    ;+
6905                    ; Round up, extra subtract went.
6906                    ;+
6907 037346 005205      12$:  INC      R5              ;INCREMENT THE QUOTIENT BY ONE.
6908 037350 001001      BNE      14$             ;IF NO OVERFLOW, WE LEAVE THE ROUND UP.
6909 037352 005305      DEC      R5              ;DON'T LET ROUNDING CAUSE OVERFLOW.
6910                    ;+
6911                    ; All done, pass quotient and exit.
6912                    ;+
6913 037354 010501      14$:  MOV      R5,R1          ;PASS QUOTIENT BACK IN R1.
6914 037356 000261      SEC                      ;INDICATE NO OVERFLOW.
6915                    ;+
6916 037360                60$:  PASS      R1          ;RESTORE GPRS, LEAVE THE FOLLOWING INTACT:
6917                    ;R1,R1SLOT(SP)          ;PUT R1 IN STACK SLOT.
6918 037366 000207      MOV      JSR      PC,8(SP)+    ;RETURN TO PREG05 SUBRT.
6919                    ;R1 - 16 BIT, UNSIGNED QUOTIENT.
6920                    ;CARRY - SET INDICATES NO OVERFLOW (SUCCESS).

```


GLOBAL SUBROUTINE

- UFDCHR -

SEQ 0180

```

6920 .SBTTL GLOBAL SUBROUTINE - UPDCHR -
6921 ;** *****
6922 ;*
6923 ;* - Update Character Pointers and Counters Routine -
6924 ;* This subroutine updates the pointers and counters associated with
6925 ;* the reception of a character on a specified line. The receive char
6926 ;* pointer is set to the next expected character, the receive char count
6927 ;* is incremented, and the count is checked to determine if the reception
6928 ;* is complete. If the reception is complete the Reception Done Flag
6929 ;* is set for the specified line.
6930 ;* INPUTS: R3 - Line number times 2 of line on which char was received.
6931 ;* BITTBL - Label of table of words used to form single bit maps.
6932 ;* CHCNTB - Base of number of chars to TX on each line table.
6933 ;* DPENDB - Base of data pattern end addresses table.
6934 ;* DPLENB - Base of data pattern lengths table.
6935 ;* RXCNTB - Base of the RX character counters table.
6936 ;* RXPTRB - Base of the RX character pointers table.
6937 ;* TXRXLB - Base of TX/RX line number association table.
6938 ;*
6939 ;* OUTPUTS: Following variables updated for line on which char was received:
6940 ;* RXCNT - Count of the number of characters received on line.
6941 ;* RXDNF - RX done flags with BIT0 for line 0 ... (Updated).
6942 ;* RXPTR - Updated to point to the next expected char on line.
6943 ;*
6944 ;* CALLING SEQUENCE: JSR PC,UPDCHR
6945 ;*
6946 ;* COMMENTS:
6947 ;*
6948 ;* SUBORDINATE ROUTINES CALLED: None.
6949 ;* - *****
6950 037370 UPDCHR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
6951 037370 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
6952 037374 016302 005364 MOV TXRXLB(R3),R2 ;GET TX LINE NUMBER OFFSET FOR THIS RX LINE.
6953 ;*
6954 ; Update the RX data pointer with wraparound at the end of the data pattern.
6955 037400 016301 003552 MOV RXPTRB(R3),R1 ;GET THE RX DATA POINTER FROM THE RX PTR TABLE.
6956 037400 005201 INC R1 ;INCREMENT THE RX POINTER VALUE BY 1.
6957 037400 020162 003312 CMP R1,DPENDB(R2) ;CMP RX PTR VALUE WITH ADR OF END OF DATA PAT.
6958 037412 103402 BLO 2$ ;SKIP WRAPPING RX PTR AROUND IF NOT AT END.
6959 037414 166201 003352 SUB DPLENB(R2),R1 ;WRAP RX PTR AROUND TO START OF DATA PATTERN.
6960 037420 010163 003552 2$: MOV R1,RXPTRB(R3) ;UPDATE THE RX POINTER WITH THE NEW VALUE.
6961 ;*
6962 ; Update the RX character count with overflow detection.
6963 ;*
6964 037424 016301 003712 MOV RXCNTB(R3),R1 ;GET THE RX CHARACTER COUNT.
6965 037430 005201 INC R1 ;INCREMENT THE RX CHAR COUNT VALUE BY 1.
6966 037432 001002 BNE 4$ ;SKIP SETTING COUNT TO MAX IF NO OVERFLOW.
6967 037434 012701 177777 MOV #-1,R1 ;SET RX CHAR COUNT VALUE TO MAX VALUE.
6968 037440 010163 003712 4$: MOV R1,RXCNTB(R3) ;UPDATE THE RX CHAR COUNT WITH NEW VALUE.
6969 ;*
6970 ; Check for RX completion on this line.
6971 ; If RX is complete on this line, set the correct RX done flag.
6972 ;
6973 037444 016204 003612 MOV CHCNTB(R2),R4 ;GET THE NUMBER OF TX CHARS IN COMPLETE TX.
6974 037450 020104 CMP R1,R4 ;COMPARE RX CHAR COUNT WITH NUMBER OF TX CHARS.
6975 037452 103403 BLO 60$ ;EXIT ROUTINE IF NOT ALL CHARS RECEIVED.

```

GLOBAL SUBROUTINE

- UPDCHR -

SEQ 0181

```
6976 037454 056337 002534 002654      BIS      BITTBL(R3),RXDNF      ;SET THE RX DONE FLAG FOR THIS LINE.
6977
6978 037462                                604:    PASS      ;RESTORE GPRS.
        037462 004736                                JSR      PC,0(SP)+
6979 037464 000207                        RTS      PC      ;RETURN TO PREG05 SUBRT.
```


GLOBAL SUBROUTINE

- VANSUP -

SEQ 0182

```

6981      .SBTTL GLOBAL SUBROUTINE          - VANSUP -
6982      ;* *****
6983      ;* - TRANSMISSION / RECEPTION SET-UP ROUTINE -
6984      ;*
6985      ;* This routine is used to initialise both the DUT and the
6986      ;* transmission/reception control parameters to the correct
6987      ;* state, prior to a Single character or DMA transmission,
6988      ;* reception test.
6989      ;*
6990      ;* INPUTS:      R1 - TX, RX LPR contents.
6991      ;*              R2 - Start address of data pattern to transmit.
6992      ;*              R3 - Length of data pattern.
6993      ;*              R4 - Number of patterns to transmit.
6994      ;*              ACTLNS - Contains a bit map of all currently active lines.
6995      ;*              LOPBCK - Contains the type of loopback mode selected.
6996      ;*              CBB - Label at base of TX/RX control block.
6997      ;*
6998      ;* OUTPUTS:     The contents of the TX/RX control block (CCB) are destroyed.
6999      ;*              The indirect address field of the DUT CSR may be destroyed.
7000      ;*              The DUT's LPR's and LNC's may be modified.
7001      ;*              The following pointers and counters are initialised:
7002      ;*              CHCNT,CHRTOT,DPEND,DPLEN,EXCNT,RXCNT,RXPTR,TXCNT,
7003      ;*              TXPTR,TXRXL.
7004      ;*              CHRTOT, RXDNF, TXDNF and TXINTF are cleared.
7005      ;*
7006      ;* CALLING SEQUENCE:  JSR      PC,VANSUP
7007      ;*
7008      ;* COMMENTS:      Modem loopback mode is inhibited if it has been selected
7009      ;*                  via Hardware P-table questions, and internal loopback mode
7010      ;*                  is forced to take place.
7011      ;*
7012      ;*
7013      ;* SUBORDINATE ROUTINES CALLED: CONMAP,RXENBL,TXRINI.
7014      ;* -- *****
7015
7016      037466      004537      005474      VANSUP:: SAVE
7017      037472      005037      002646      CLR      CHRTOT      JSR      ;SAVE CONTENTS OF THE GPR'S R0 THRU R5.
7018      037476      005037      002420      CLR      TXINTF      ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7019      037502      005037      002652      CLR      TXDNF      ;CLEAR TOTAL RECEIVED CHAR COUNTER.
7020      037506      005037      002654      CLR      RXDNF      ;CLEAR FLAGS USED TO LOG DMA H.OVER ERRORS.
7021      ;*
7022      ;* Set up the Transmission/Reception Control block to the desired state.
7023      ;*
7024      037512      010137      003272      MOV      R1,CBB      ;SET CONTENTS OF LPR PARAMS IN TX/RX C.BLK.
7025      037516      012701      003272      MOV      @CBB,R1      ;GET BASE ADDRESS OF CONTROL BLOCK.
7026      037522      005201      INC      R1      ;INCREMENT ADDRESS FOR NEXT WORD
7027      037524      005201      INC      R1      ;INITIALISE THE FOLLOWING IN THE CNTRL.BLK:
7028      037526      012721      000004      MOV      @4,(R1)+      ; LNCRTL PARAMETER, ENABLE RECEIVERS.
7029      037532      010221      MOV      R2,(R1)+      ; START ADDRESS OF DATA PATTERN.
7030      037534      010321      MOV      R3,(R1)+      ; DATA PATTERN LENGTH.
7031      037536      010421      MOV      R4,(R1)+      ; NUMBER OF DATA PATTERNS TO TRANSMIT.
7032      037540      013721      002272      MOV      ACTLNS,(R1)+      ; BIT MAP OF LINES TO INITIALISE.
7033      037544      032737      000004      002274      BIT      @BIT2,LOPCK      ;TEST IF MODEM LOOPBACK MODE HAS BEEN SELECTED.
7034      037552      001404      BEQ      Z1      ;DONT SELECT INTERNAL LOOPBACK IF STAGRD OR LOCAL.
7035      037554      012702      000001      MOV      @1,R2      ;FORCE INTERNAL LOOPBACK MODE TO BE SELECTED.
7036      037560      110221      MOV      R2,(R1)+      ;INITIALISE LOOPBACK MODE IN CONTROL BLOCK.

```

GLOBAL SUBROUTINE

- VANSUP -

```

7037 037562 000402
7038 037564 113721 002274
7039 037570 005201
7040 037572 012711 000002
7041
7042
7043
7044
7045 037576 004737 036620
7046
7047
7048
7049 037602 012701 000377
7050 037606 013702 002272
7051 037612 005101
7052 037614 005102
7053 037616 040102
7054 037620 010237 003304
7055 037624 005037 003302
7056 037630 004737 036620
7057
7058
7059
7060
7061 037634 012705 000377
7062 037640 004737 035356
7063
7064
7065
7066 037644 013705 002272
7067 037650 004737 030314
7068 037654 004737 035452
7069 037660
      037660 004736
7070 037662 000207

      BR      4$      ;SKIP NEXT INSTRUCTION IF IN MODEM LOOPBACK.
2$:      MOVB   LOPBCK,(R1). ;SET LOOPBACK MODE.
4$:      INC    R1      ;INCREMENT ADDRESS FOR THE NEXT WORD.
      MOV     @2,(R1)    ;SET AMOUNT OF OFFSET EACH TX STARTS AT TO 2.
;+
; Initialise the DUT and the associated pointers and counters, to the state
; dictated by the contents of the TX/RX control block.
;-
      JSR     PC,TXRINI    ;INITIALISE DUT.
;+
; Initialise pointers and counters for inactive lines to zero.
;
      MOV     @MAPLNS,R1   ;GET THE LINE BIT MAP FOR ALL LINES.
      MOV     ACTLNS,R2   ;GET THE ACTIVE LINE BIT MAP.
      COM     R1
      COM     R2
      BIC     R1,R2       ;GENERATE AN IN-ACTIVE LINE BIT MAP.
      MOV     R2,CBMAPA   ;MOVE BIT MAP TO THE CONTROL BLOCK.
      CLR     CBOPNA      ;CLEAR THE REPEAT TX COUNT IN CNTRL BLCK.
      JSR     PC,TXRINI    ;SET UP PARAMETERS FOR INACTIVE LINES.
;+
; Disable Receivers on all lines to ensure correct initialisation of only the
; lines that are selected.
;-
      MOV     @MAPLNS,R5   ;SET UP BIT MAP FOR ALL LINES.
      JSR     PC,RXDSBL    ;DISABLE RX ON ALL LINES.
;+
; Enable receivers on associated (RX) lines.
;-
      MOV     ACTLNS,R5   ;GET THE ACTIVE LINE BIT MAP.
      JSR     PC,CONMAP    ;GENERATE AN ASSOCIATED LINE BIT MAP.
      JSR     PC,RXENBL    ;ENABLE RECEIVERS ON ASSOCIATED LINES
60$:      PASS           ;RESTORE GPR'S.
      JSR     PC,@(SP)+    ;RETURN TO PREG05 SUBRT.
      RTS     PC

```

GLOBAL SUBROUTINE

- WAIBIC -

```

7072 .SBTTL GLOBAL SUBROUTINE - WAIBIC -
7073 ;* *****
7074 ;* - Wait For Bit Clear Routine -
7075 ;* This subroutine waits for the specified bit to become clear. If the
7076 ;* specified bit goes to a clear state within the specified time-out
7077 ;* period a success indication is returned by this routine.
7078 ;* The last value which is read looking for the condition is returned to
7079 ;* allow the use of this routine to look for destructive read conditions.
7080 ;*
7081 ;* INPUTS: R1 - Time-out value and bit number indication:
7082 ;* Bits 15 thru 12 - Number of bit to test (range 0 thru 15).
7083 ;* Bits 11 thru 0 - Time-out value in milli-seconds (4095 max).
7084 ;* R2 - Address of word containing the bit to test.
7085 ;* MSLCNT.
7086 ;*
7087 ;* OUTPUTS: R2 - The last word which was read to check for the condition.
7088 ;* CARRY - Success flag (CARRY set if bit clr before time-out).
7089 ;*
7090 ;* CALLING SEQUENCE: MOV #130040,R1 ;PASS BIT 11 (13 OCTAL) AND
7091 ;* ; 32 (40 OCTAL) MS DELAY.
7092 ;* MOV #LABEL,R2 ;TEST BIT IN WORD AT "LABEL".
7093 ;* JSR PC,WAIBIC ;WAIT 32 MS FOR BIT 11 TO CLR.
7094 ;*
7095 ;* COMMENTS:
7096 ;*
7097 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
7098 ;* *****
7099
7100 037664 WAIBIC:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
037664 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7101 037670 010204 MOV R2,R4 ;SET UP THE ADDRESS PARAMETER FOR MSLGET.
7102 037672 010102 MOV R1,R2
7103 037674 042701 170000 BIC #170000,R1 ;SEPERATE DELAY COUNT OUT OF PASSED PARAMETER.
7104 037700 042702 007777 BIC #7777,R2 ;SEPERATE LINE NUMBER FIELD OF PASSED PARAM.
7105 037704 000302 SWAB R2 ;PUT LINE NUMBER FIELD IN LSBYTE.
7106 037706 006202 ASR R2 ;SHIFT THE LINE NUMBER FIELD INTO THE PROPER
7107 037710 006202 ASR R2 ; POSITION TO USE IT AS A WORD TABLE OFFSET
7108 037712 006202 ASR R2 ; FOR THE TABLE LOOKUP OF THE LINE BIT MAP.
7109 037714 016202 002534 MOV BITTBL(R2),R2 ;GET BIT MAP OF LINE TO TEST FROM TABLE.
7110 037720 005003 CLR R3 ;INDICATE THAT THE BIT SHOULD BE CLR.
7111 037722 004737 031726 JSR PC,MSLGET ;WAIT FOR THE BIT TO BE CLR WITHIN TIME-OUT.
7112 ; CARRY IS CORRECT UPON MSLGET RETURN.
7113 037726 010002 MOV R0,R2 ;PASS LAST VALUE READ AS OUTPUT PARAMETER.
7114 037730 010266 000006 60: PASS R2 ;RESTORE GPRS, EXCEPT THE FOLLOWING:
037730 004736 MOV R2,R2SLOT(SP) ;PUT R2 IN STACK SLOT.
037734 JSR PC,(SP) ;RETURN TO PREG05 SUBRT.
7115 ; R2 - LAST VALUE READ LOOKING FOR CONDITION.
7116 037736 000207 RTS PC ; CARRY - SUCCESS FLAG (SET IF BIT FOUND CLR).

```

GLOBAL SUBROUTINE

- WAIBIS -

```

7118 .SBTTL GLOBAL SUBROUTINE
7119
7120 ;* *****
7121 ;* - Wait For Bit Set Routine -
7122 ;* This subroutine waits for the specified bit to become set. If the
7123 ;* specified bit goes to a set state within the specified time-out
7124 ;* period a success indication is returned by this routine.
7125 ;* The last value which is read looking for the condition is returned to
7126 ;* allow the use of this routine to look for destructive read conditions.
7127 ;*
7128 ;* INPUTS: R1 - Time-out value and bit number indication:
7129 ;* Bits 15 thru 12 - Number of bit to test (range 0 thru 15).
7130 ;* Bits 11 thru 0 - Time-out value in milli-seconds (4095 max).
7131 ;* R2 - Address of word containing the bit to test.
7132 ;* MSLCNT.
7133 ;*
7134 ;* OUTPUTS: R2 - The last word which was read to check for the condition.
7135 ;* CARRY - Success flag (CARRY set if bit set before time-out).
7136 ;*
7137 ;* CALLING SEQUENCE: MOV #130040,R1 ;PASS BIT 11 (13 OCTAL) AND
7138 ;* ; 32 (40 OCTAL) MS DELAY.
7139 ;* MOV #LABEL,R2 ;TEST BIT IN WORD AT "LABEL".
7140 ;* JSR PC,WAIBIS ;WAIT 32 MS FOR BIT 11 TO SET.
7141 ;*
7142 ;* COMMENTS.
7143 ;*
7144 ;* SUBORDINATE ROUTINES CALLED: MSLGET.
7145 ;* -- *****
7146
7147 037740 004537 005474 WAIBIS:: SAVE
7148 037744 010204 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
7149 037750 042701 170000 MOV R2,R4 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT
7150 037754 042702 007777 MOV R1,R2 ;SET UP THE ADDRESS PARAMETER FOR MSLGET.
7151 037760 000302 BIC #170000,R1 ;SEPERATE DELAY COUNT OUT OF PASSED PARAMETER.
7152 037762 006202 BIC #,777,R2 ;SEPERATE LINE NUMBER FIELD OF PASSED PARAM.
7153 037764 006202 SWAB R2 ;PUT LINE NUMBER FIELD IN LSBYTE.
7154 037766 006202 ASR R2 ;SHIFT THE LINE NUMBER FIELD INTO THE PROPER
7155 037770 016202 002534 ASR R2 ; POSITION TO USE IT AS A WORD TABLE OFFSET
7156 037774 010203 MOV BITTBL(R2),R2 ; FOR THE TABLE LOOKUP OF THE LINE BIT MAP.
7157 037776 004737 031726 MOV R2,R3 ;GET BIT MAP OF LINE TO TEST FROM TABLE.
7158 JSR PC,MSLGET ;INDICATE THAT THE BIT SHOULD BE SET.
7159 040002 010002 MOV R0,R2 ;WAIT FOR THE BIT TO BE SET WITHIN TIME-OUT.
7160 040004 010266 000006 60$: PASS R2 ; CARRY IS CORRECT UPON MSLGET RETURN.
7161 040010 004736 MOV R2,R2SLOT(SP) ;PASS LAST VALUE READ AS OUTPUT PARAMETER.
7162 040012 000207 RTS PC ;RESTORE GPRS, EXCEPT THE FOLLOWING:
; R2,R2SLOT(SP) ;PUT R2 IN STACK SLOT.
; PC,8(SP)+ ;RETURN TO PREG05 SUBRT.
; R2 - LAST VALUE READ LOOKING FOR CONDITION.
; CARRY - SUCCESS FLAG (SET IF BIT FOUND SET).

```

GLOBAL SUBROUTINE

- WAITTX -

SEQ 0186

```

7164 .SBTTL GLOBAL SUBROUTINE WAITTX -
7165 ;** *****
7166 ;* - WAIT FOR TX TO FINISH -
7167 ;* This subroutine is used in the FIHAVL.TST.
7168 ;* It waits for transmission to complete ie TX_ACTION. Then delays
7169 ;* for 5 milliseconds to allow time for the last character to get into
7170 ;* the fifo.
7171 ;*
7172 ;* INPUTS: CSRA - Contains the address of the CSR.
7173 ;*
7174 ;* OUTPUTS: Carry - Set indicates success.
7175 ;*
7176 ;* CALLING SEQUENCE: JSR PC,WAITTX
7177 ;*
7178 ;* COMMENTS:
7179 ;*
7180 ;* SUBORDINATE ROUTINES CALLED: DELAY,WAIBIS.
7181 ;-- *****
7182
7183 040014 WAITTX:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
      040014 004537 005474 R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7184 040020 012701 170536 MOV #170536,R1 ;PASS TIME-OUT VALUE OF 350 MILLI SECS.
7185 040024 013702 002300 MOV CSRA,R2 ;PASS THE ADDRESS OF THE CSR.
7186 040030 004737 037740 JSR PC,WAIBIS ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
7187 040034 103005 BCC 60$ ;BRANCH IF FIFO EMPTY, ABORT THE TEST.
7188 040036 012704 000005 MOV #5,R4 ;PASS DELAY OF 5 MILLI SECS.
7189 040042 004737 030370 JSR PC,DELAY ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.
7190 040046 000261 SEC ;SET CARRY TO INDICATE SUCCESS.
7191
7192 040050 60$: PASS ;RESTORE GPRS.
      040050 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
7193
7194 040052 000207 RTS PC ;PASS THE CARRY BIT, SET INDICATES SUCCESS.

```

GLOBAL SUBROUTINE

- WDPDR -

SEQ 0187

```

7196 .SBTTL GLOBAL SUBROUTINE - WDPDR -
7197 ;** *****
7198 ;* - Write Data Pattern to Device Registers -
7199 ;* This routine writes a rotated data pattern to each of the 6 device
7200 ;* registers of each active line of the device under test.
7201 ;* The data pattern is rotated once after each write to a device register
7202 ;* on a particular line. The starting data pattern for each line
7203 ;* is rotated once after writing all the registers on a particular
7204 ;* line. This leads to the following data pattern:
7205 ;* Line 0, register 0 - shifted 0 bit positions
7206 ;* Line 0, register 1 - shifted 1 bit position
7207 ;*
7208 ;* Line 1, register 0 - shifted 1 bit position
7209 ;* Line 2, register 1 - shifted 2 bit positions
7210 ;*
7211 ;* Any bits fields in the device registers that cannot be altered
7212 ;* are masked out of the data pattern before it is written.
7213 ;* This routine will use either MOV, MOVB, BIS, BISB, BIC, or BICB
7214 ;* instructions. The upper or lower byte can be specified for writing.
7215 ;*
7216 ;* INPUTS: R2 - Used to pass in the data pattern to be rotated & written.
7217 ;* R3 - Byte indicator (- => lo byte, + => hi byte, 0 => both).
7218 ;* R4 - Operation type indicator (- => BIC, + => BIS, 0 => MOV).
7219 ;* ACTLNS - Bit map of the active lines on the device under test.
7220 ;* CSRA - Contains the CSR address of the Device under test.
7221 ;* DRADRT - Base address of device register address table.
7222 ;* LPRO - Equated to LPR reg offset from device CSR address.
7223 ;* NUMLNS - Number of lines on the device under test.
7224 ;* TXBFCO - Equated to TBUFFCT reg offset from device CSR address.
7225 ;* LNBTB - Base address of the unused bit table.
7226 ;*
7227 ;* OUTPUTS: Device registers on all active device lines are modified.
7228 ;*
7229 ;* CALLING SEQUENCE: JSR PC,WDPDR
7230 ;*
7231 ;* COMMENTS: This routine does not write any data to the TX.CHAR registers.
7232 ;* The CSR is cleared except for the IND.ADR.REG field.
7233 ;*
7234 ;* SUBORDINATE ROUTINES CALLED: ROLDAP.
7235 ;* *****
7236
7237 040054 WCPDR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
040054 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7238 ;*
7239 ;* Set up outer loop which writes the data pattern to each line's registers
7240 ;*
7241 040060 005005 CLR R5 ;CLEAR LINE COUNTER TO SELECT LINE 0.
7242 ;*
7243 ;* The outer loop follows. Each pass through this loop writes data to all of
7244 ;* the device registers for a particular line if it is active.
7245 ;*
7246 040062 010204 2$: MOV R2,R4 ;SAVE THE OUTER LOOP DATA PATTERN.
7247 040064 010577 142210 MOV R5,@CSRA ;SET CSR IND.ADR.REG FIELD TO THIS LINE.
7248 040070 006305 ASL R5 ;TURN LINE NUMBER INTO A WORD OFFSET.
7249 040072 036537 002534 002272 BIT BITTBL(R5),ACTLNS
7250 040100 001451 BEQ 20$ ;LINE ACTIVE? NO, SKIP THIS LINE.
7251 040102 012701 000004 MOV @LPRO,R1 ;YES, INITIALIZE THE REGISTER OFFSET.

```


GLOBAL SUBROUTINE

- WDPDR -

```

7252      ;+
7253      ; The inner loop follows. Each pass through this loop writes data to a
7254      ; device register.
7255      ;
7256 040106 010200      4$:  MOV    R2,R0
7257 040110 046100 002320      BIC    UNBTTB(R1),R0      ;CLEAR BIT FIELDS FOR UNUSED REGISTER BITS.
7258 040114 016103 002300      MOV    DRADRT(R1),R3      ;GET THE ADDRESS OF THE DEVICE REGISTER.
7259 040120 005766 000010      TST    R3SLOT(SP)      ;CHECK THE OPERAND TYPE INDICATOR.
7260 040124 003402      BLE     6$      ;HIGH BYTE? NO, SKIP HIGH BYTE ADDRESS SET UP.
7261 040126 005203      INC     R3      ;YES, SET THE REG ADDRESS TO THE HIGH BYTE.
7262 040130 000300      SWAB    R0      ;MOVE HIGH BYTE DATA INTO THE LOW BYTE.
7263 040132 005766 000010      6$:  TST    R3SLOT(SP)      ;CHECK THE OPERAND TYPE INDICATOR.
7264 040136 001412      BEQ     12$      ;WORD ACCESS? YES, GO PERFORM WORD ACCESS.
7265      ;+
7266      ;Perform byte access to the specified byte of the specified register.
7267      ;-
7268 040140 005766 000012      TST    R4SLOT(SP)      ;NO, CHECK THE ACCESS TYPE INDICATOR.
7269 040144 100403      BMI     8$      ;USE BIC? YES, GO PERFORM BICB INSTRUCTION.
7270 040146 001404      BEQ     10$      ;USE MOV? YES, GO PERFORM MOVB INSTRUCTION.
7271 040150 150013      BISB    R0,(R3)      ;NEITHER. PERFORM BISB ACCESS TO REGISTER.
7272 040152 000415      BR      18$
7273 040154 140013      8$:  BICB    R0,(R3)      ;PERFORM BICB ACCESS TO REGISTER.
7274 040156 000413      BR      18$
7275 040160 110013      10$:  MOVB    R0,(R3)      ;PERFORM MOVB ACCESS TO REGISTER.
7276 040162 000411      BR      18$
7277      ;+
7278      ;Perform word access to the specified register.
7279      ;-
7280 040164 005766 000012      12$:  TST    R4SLOT(SP)      ;CHECK THE ACCESS TYPE INDICATOR.
7281 040170 100403      BMI     14$      ;USE BIC? YES, GO PERFORM BIC INSTRUCTION.
7282 040172 001404      BEQ     16$      ;USE MOV? YES, GO PERFORM MOV INSTRUCTION.
7283 040174 050013      BIS     R0,(R3)      ;NEITHER. PERFORM BIS ACCESS TO REGISTER.
7284 040176 000403      BR      18$
7285 040200 040013      14$:  BIC     R0,(R3)      ;PERFORM BIC ACCESS TO REGISTER.
7286 040202 000401      BR      18$
7287 040204 010013      16$:  MOV     R0,(R3)      ;PERFORM MOV ACCESS TO REGISTER.
7288      ;+
7289      ; Prepare the data pattern and offset for the next register on this line.
7290      ;-
7291 040206 004737 034642      18$:  JSR     PC,ROLDAP      ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
7292 040212 062701 000002      ADD     #2,R1      ;INCREMENT OFFSET FOR NEXT REGISTER.
7293 040216 020127 000016      CMP     R1,#TXBFC0      ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
7294 040222 003731      BLE     4$      ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
7295      ;+
7296      ; Back into the outer loop. Now set up for next line. Loop if not done.
7297      ;-
7298 040224 010402      20$:  MOV     R4,R2      ;SET UP TO ROTATE THE DATA PATTERN.
7299 040226 004737 034642      JSR     PC,ROLDAP      ;ROTATE THE DATA PATTERN.
7300 040232 006205      ASR     R5      ;CONVERT BACK TO LINE NUMBER FROM WORD OFFSET.
7301 040234 005205      INC     R5      ;COUNT THIS LINE.
7302 040236 020527 000010      CMP     R5,#NUMLNS      ;COMPARE LINE COUNT WITH NUMBER OF LINES.
7303 040242 002707      BLT     2$      ;LOOP IF SOME LINES NOT DONE.
7304      ;+
7305 040244      60$:  PASS      ;RESTORE GPRS.
7306 040246 000207      RTS     PC      JSR     PC,@(SP),      ;RETURN TO PREG05 SUBRT.

```

GLOBAL SUBROUTINE

- WTWLNC -

```

7308 .SBTTL GLOBAL SUBROUTINE - WTWLNC -
7309 ;* *****
7310 ;* - Line Control Register Setup Routine -
7311 ;* This subroutine is used to set the Device Under Test (DUT) Line
7312 ;* Control Registers (LNCTRL) to the specified state. Only the LNCTRLS
7313 ;* for the specified lines are altered.
7314 ;*
7315 ;* INPUTS: R0 - New line parameters.
7316 ;* R5 - Bit map of lines to be altered.
7317 ;* CSRA - Contains address of the DUT CSR.
7318 ;* IESTAT - Contains the current state of the TX and RX interrupt
7319 ;* enable bits in the CSR.
7320 ;* LNCTRA - Contains address of the DUT LNCTRL registers.
7321 ;*
7322 ;* OUTPUTS: LNCTRL - Specified DUT Line Control Registers are altered.
7323 ;*
7324 ;* CALLING SEQUENCE: JSR PC,WTWLNC
7325 ;*
7326 ;* COMMENTS:
7327 ;*
7328 ;* SUBORDINATE ROUTINES CALLED: ALTFLD.
7329 ;* *****
7330
7331 040250 WTWLNC:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
040250 004537 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
005474
7332 ;*
7333 ; Set up the parameters for the call to ALTFLD.
7334 ;*
7335 040254 013701 002310 MOV LNCTRA,R1 ;SET UP THE REGISTER ADDRESS PARAMETER.
7336 040260 010002 MOV R0,R2 ;SET UP THE DESIRED REGISTER CONTENTS.
7337 040262 010503 MOV R5,R3 ;SET UP THE BIT MAP OF LINES TO ALTER.
7338 040264 012704 177777 MOV #-1,R4 ;SELECT ALL REGISTER BITS TO BE ALTERED.
7339 ;*
7340 ; Call the subroutine which alters the register contents.
7341 ;*
7342 040270 004737 026760 JSR PC,ALTFLD ;ALTER THE REGISTER CONTENTS.
7343
7344 040274 604: PASS ;RESTORE GPRS.
040274 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
7345 040276 000207 RTS PC

```


GLOBAL SUBROUTINE

- WTWLNS -

```

7347 .SBTTL GLOBAL SUBROUTINE - WTWLNS -
7348 ;*****
7349 ;* - Write Word to all Lines routine -
7350 ;* This subroutine writes a specified word to the specified DHV11-M device
7351 ;* register for each of the DHV11-M lines. It could be used to clear all
7352 ;* of the LNCTRL registers or to initialize all of the LPR registers to
7353 ;* the same parameters.
7354 ;*
7355 ;* INPUTS: R1 - Address of the specified registers.
7356 ;* R2 - Word to write into the specified registers.
7357 ;* IESTAT - Saved states of the TX.IE and RX.IE bits.
7358 ;* MAPLNS - Equated to bit map of lines on device (8 for DHV11).
7359 ;* CSRA.
7360 ;*
7361 ;* OUTPUTS: DEVICE REGISTERS - Specified registers given new value.
7362 ;* CSR IND.ADR.REG field Destroyed.
7363 ;* CSR Interrupt Enable bits - Set to states in IES AT.
7364 ;*
7365 ;* CALLING SEQUENCE: JSR PC,WTWLNS
7366 ;*
7367 ;* COMMENTS: Note that the specified registers for all lines are altered
7368 ;* by this routine. This routine should NOT be used to alter
7369 ;* the states of partial register fields or to alter a register
7370 ;* for fewer than all of the lines.
7371 ;* The specified registers are read before being written.
7372 ;*
7373 ;* SUBROUTINES CALLED: ALTFLD.
7374 ;*****
7375
7376 040300 WTWLNS:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
040300 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7377
7378 ; Set up the bit map of lines to change and mask of bits to alter parameters.
7379 ; -
7380 040304 012703 000377 MOV #MAPLNS,R3 ;GET THE BIT MAP OF LINES TO CHANGE.
7381 040310 012704 177777 MOV #-1,R4 ;INDICATE ALL 16 BITS TO BE CHANGED.
7382
7383 ; Call the subroutine to write the specified registers.
7384 ; -
7385 040314 004737 026760 JSR PC,ALTFLD ;CHANGE THE REGISTERS.
7386
7387 040320 60$: PASS ;RESTORE GPRS.
040320 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
7388 040322 000207 RTS PC

```

GLOBAL SUBROUTINE

WTWLPR -

```

7390 .SBTTL GLOBAL SUBROUTINE - WTWLPR -
7391 ;+ *****
7392 ;* - Line Parameter Register Setup Routine -
7393 ;* This subroutine is used to set the Device Under Test (DUT) Line
7394 ;* Parameter Registers (LPR) to the specified state. Only the LPRs for
7395 ;* the specified lines are altered.
7396 ;*
7397 ;* INPUTS: R0 - New line parameters.
7398 ;* R5 - Bit map of lines to be altered.
7399 ;* CSRA - Contains address of the DUT CSR.
7400 ;* IESTAT - Contains the current state of the TX and RX interrupt
7401 ;* enable bits in the CSR.
7402 ;* LPRA - Contains address of the DUT LPR.
7403 ;*
7404 ;* OUTPUTS: LPR - Specified DUT Line Parameter Registers are altered.
7405 ;*
7406 ;* CALLING SEQUENCE: JSR PC,WTWLPR
7407 ;*
7408 ;* COMMENTS:
7409 ;*
7410 ;* SUBORDINATE ROUTINES CALLED: ALTFLD.
7411 ;-- *****
7412
7413 040324 WTWLPR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
040324 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7414 ;+
7415 ; Set up the parameters for the call to ALTFLD.
7416 ;--
7417 040330 013701 002304 MOV LPRA,R1 ;SET UP THE REGISTER ADDRESS PARAMETER.
7418 040334 010002 MOV R0,R2 ;SET UP THE DESIRED REGISTER CONTENTS.
7419 040336 010503 MOV R5,R3 ;SET UP THE BIT MAP OF LINES TO ALTER.
7420 040340 012704 177777 MOV #-1,R4 ;SELECT ALL REGISTER BITS TO BE ALTERED.
7421 ;+
7422 ; Call the subroutine which alters the register contents.
7423 ;--
7424 040344 004737 026760 JSR PC,ALTFLD ;ALTER THE REGISTER CONTENTS.
7425
7426 040350 60$: PASS ;RESTORE GPRS.
040350 004736 JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.
7427 040352 000207 RTS PC

```

INTERRUPT SERVICE ROUTINE - CACHRX -

```

7429 .SBTTL INTERRUPT SERVICE ROUTINE - CACHRX -
7430 ;* *****
7431 ;* - Catch Receiver interrupt.
7432 ;* This routine is used in several tests, to log a count of the
7433 ;* number of receiver interrupts that occur.
7434 ;*
7435 ;* INPUTS: CSRA - Contains the address of the CSR.
7436 ;* RXINTC - Holds the count of the number of RX interrupts
7437 ;* that occurred.
7438 ;*
7439 ;* OUTPUTS: RXINTC - Contains the updated interrupt count.
7440 ;*
7441 ;*
7442 ;* CALLING SEQUENCE: Put the address of the label CACHRX in the vector
7443 ;* location.
7444 ;*
7445 ;* COMMENTS:
7446 ;*
7447 ;* SUBORDINATE ROUTINES CALLED: none
7448 ;* - *****
7449
7450 040354 CACHRX::SAVE
7451 040354 004537 005474 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
7452 040360 013701 002412 MOV RXINTC,R1 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7453 040366 102001 INC R1 ;GET THE RECEIVER INTERRUPT COUNT
7454 040370 005301 BVC 2$ ;INCREMENT THE COUNT
7455 040372 010137 002412 DEC R1 ;BRANCH IF NO OVERFLOW OCCURRED
7456 040376 2$: MOV R1,RXINTC ;RESET THE COUNT TO 177777
7457 040400 000002 60$: PASS ;SAVE NEW COUNT VALUE
;RESTORE GPRS.
PC,0(SP)+ ;RETURN TO PREG05 SUBRT.
RTI

```

INTERRUPT SERVICE ROUTINE - CACHTX -

```

7459 .SBTTL INTERRUPT SERVICE ROUTINE - CACHTX -
7460 ;** *****
7461 ;* - Catch Transmitter interrupt.
7462 ;* This routine is used in several tests, to log a count of the
7463 ;* number of transmission interrupts that occur.
7464 ;*
7465 ;* INPUTS: CSRA - Contains the address of the CSR.
7466 ;* TXINTC - Holds the count of the number of TX interrupts
7467 ;* that occurred.
7468 ;*
7469 ;* OUTPUTS: TXINTC - Contains the updated interrupt count.
7470 ;*
7471 ;*
7472 ;* CALLING SEQUENCE: Put the address of the label CACHTX in the vector
7473 ;* location.
7474 ;*
7475 ;* COMMENTS:
7476 ;*
7477 ;* SUBORDINATE ROUTINES CALLED: none
7478 ;-- *****
7479
7480 040402 CACHTX::SAVE
7481 040402 004537 005474 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
7482 040406 013701 002416 MOV TXINTC,R1 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7483 040412 005201 INC R1 ;GET THE TRANSMISSION INTERRUPT COUNT
7484 040414 102001 BVC 2$ ;INCREMENT THE COUNT
7485 040416 005301 DEC R1 ;BRANCH IF NO OVERFLOW OCCURRED
7486 040420 010137 002416 2$: MOV R1,TXINTC ;RESET THE COUNT TO 177777
7487 040424 004736 60$: PASS ;SAVE NEW COUNT VALUE
7488 040426 000002 RTI ;RESTORE GPRS.
7489 ;PC,0(SP)+ ;RETURN TO PREG05 SUBRT.

```

INTERRUPT SERVICE ROUTINE

- CLKINT -

SEQ 0194

```

7489 .SBTTL INTERRUPT SERVICE ROUTINE - CLKINT -
7490 ;** *****
7491 ;* This routine is executed CLKHRZ times per second. It decrements the
7492 ;* two timer counters down to zero.
7493 ;*
7494 ;* INPUTS:    TIMER1 - Timer counter #1.
7495 ;*           TIMER2 - Timer counter #2.
7496 ;*           TIMER3 - Timer counter for call of BREAK macro.
7497 ;*
7498 ;* OUTPUTS:   The 2 timer counters are decremented if they are not zero.
7499 ;*
7500 ;* CALLING SEQUENCE: Put #CLKINT in the clock interrupt vector slot.
7501 ;*                   Put the desired time period (seconds times CLKHRZ) in
7502 ;*                   either TIMER1 or TIMER2 and poll the respective timer
7503 ;*                   counter to detect its going to 0 on time-out.
7504 ;*
7505 ;* COMMENTS:   The 2 counters will not wraparound but will stop at 0. This
7506 ;*             allows the detection of a time-out any time after the time-out
7507 ;*             has occurred until the timer counter is set to another value.
7508 ;*
7509 ;* SUBORDINATE ROUTINES CALLED: None.
7510 ;-- *****
7511
7512 040430 005737 002450 CLKINT:: TST    TIMER1    ;CHECK FOR TIMER1 AT ZERO.
7513 040434 001402      BEQ     2$      ;BRANCH TO LEAVE IT AT ZERO IF IT IS ZERO.
7514 040436 005337 002450      DEC     TIMER1    ;DECREMENT TIME COUNT.
7515 040442 005737 002452 2$: TST    TIMER2    ;CHECK FOR TIMER2 AT ZERO.
7516 040446 001402      BEQ     4$      ;BRANCH TO LEAVE IT ALONE IF IT'S ALREADY ZERO.
7517 040450 005337 002452      DEC     TIMER2    ;DECREMENT TIME COUNT.
7518 040454 005337 002454 4$: DEC     TIMER3    ;DECREMENT THE BREAK COUNT.
7519 040460 001006      BNE     60$      ;EXIT IF NOT TIME TO CALL BREAK.
7520 040462 013737 002456 002454      MOV     BCOUNT,TIMER3 ;SET UP TIME TILL NEXT BREAK.
7521 040470 010046      MOV     R0,-(SP) ;SAVE CONTENTS OF R0 FROM BREAK MACRO.
7522 040472      BREAK      ;CHECK FOR OPERATOR CONTROL/C.
7523 040474 012600      MOV     (SP)+,R0      TRAP    C$BRK
7524 040476 000002 60$: RTI

```

INTERUPT SERVICE ROUTINE

- RXBRRT -

SEQ 0195

```

7526 .SBTTL INTERUPT SERVICE ROUTINE - RXBRRT -
7527 ;* *****
7528 ;* - BR Level Test Receive Interrupt Service Routine -
7529 ;* This service routine handles Receive Interrupts during the Interrupt
7530 ;* BR Level Test. This routine counts the interrupt and sets a flag
7531 ;* to indicate that the interrupt has occurred. It also checks the
7532 ;* flag which indicates that a TX interrupt has occurred. If the TX
7533 ;* interrupt flag is set, this routine sets an interrupt order error
7534 ;* flag indicating that a transmit interrupt was serviced before a
7535 ;* simultaneous receive interrupt.
7536 ;*
7537 ;* INPUTS: RXINTC - Holds the count of the number of RX interrupts.
7538 ;*          RXINTF - RX Interrupt flags.
7539 ;*
7540 ;* OUTPUTS: RXINTC - Contains the updated interrupt count.
7541 ;*          RXINTF - RX Int flags:
7542 ;*                  (Bit 0 set, bit 14 set if TXINTF bit 0 is set.)
7543 ;*
7544 ;* CALLING SEQUENCE: Put the address of the label RXBRRT in the vector
7545 ;*                  location.
7546 ;*
7547 ;* COMMENTS: NOTE: The FIFO is purged by this routine.
7548 ;*
7549 ;* SUBORDINATE ROUTINES CALLED: None.
7550 ;*
7551 ;* *****
7552 RXBRRT:: SAVE
7553 040500 004537 005474          MOV  BRBUFA,R0      ;SAVE CONTENTS OF GPRS R0 THRU R5.
7554 040510 013701 002412          MOV  RXINTC,R1      ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7555 040514 005201                INC  R1              ;READ THE CHAR OUT OF THE FIFO.
7556 040516 001402                BEQ   2$             ;GET THE INTERRUPT COUNT.
7557 040520 010137 002412          MOV  R1,RXINTC      ;INCREMENT THE COUNT.
7558 040524 013701 002414          MOV  RXINTF,R1      ;BY PASS UPDATING COUNT IF OVERFLOW OCCURRED.
7559 040530 052701 000001          BIS  #BIT0,R1      ;SAVE NEW COUNT VALUE.
7560 040534 032737 000001 002420  BIS  #BIT0,TXINTF   ;GET THE RX INTERRUPT FLAGS.
7561 040542 001402                BEQ   4$             ;SET THE RX INTERRUPT HAS OCCURRED FLAG.
7562 040544 052701 040000          BIS  #BIT14,R1     ;TEST THE "TX INT HAS OCCURRED" FLAG.
7563 ;*                                     ;SKIP SETTING ERROR FLAG IF NO TX INT.
7564 ;*                                     ;SET THE INTERRUPT ORDER ERROR FLAG.
7565 ;*
7566 ;* 8 FIFO codes will cause 8 interrupts, after these 8 codes we don't want
7567 ;* to check the interrupt order, because perhaps a BMP code has come in
7568 ;* between the servicing of the 8 FIFO code interrupts and the servicing
7569 ;* of one of the TX interrupts.
7569 040550 023737 002412 000010 4$: CMP  RXINTC,NUMLNS ;TEST FOR ALL SELFTEST CODE INTS DONE.
7570 040556 003002                BGT   60$           ;SKIP UPDATING RX INT FLAGS IF EXTRA RX INTS.
7571 040560 010137 002414          MOV  R1,RXINTF     ;UPDATE THE RX INTERRUPT FLAGS.
7572 040564                60$: PASS                  ;RESTORE GPRS.
7573 040566 004736                RTI                  ;RETURN TO PREG05 SUBRT.

```


INTERRUPT SERVICE ROUTINE - RXCHRS -

```

7575 .SBTTL INTERRUPT SERVICE ROUTINE - RXCHRS -
7576 ;* *****
7577 ;* - DMA RECEIVE Interrupt Service Routine -
7578 ;* This routine executes in response to an interrupt caused by the DUT
7579 ;* RX.DATA.AVAIL bit becoming active. This routine reads characters from
7580 ;* the DUT receive character FIFO and deposits them into the receive
7581 ;* buffer in memory. If the number of characters in the receive buffer
7582 ;* exceeds a specified threshold, transmission is halted (by clearing all
7583 ;* DUT TX.ENABLE bits) and if the receive buffer is full reception is
7584 ;* halted (by disabling RX interrupts). The routine exits if the receive
7585 ;* buffer becomes full or if a character is read from the FIFO with the
7586 ;* DATA.VALID bit clear.
7587 ;*
7588 ;* INPUTS: RBUFA - Contains address of the DUT RX character FIFO.
7589 ;*          RXBCNT - RX buffer character count.
7590 ;*          RXBDTX - Equated to RX buffer level at which to disable TX.
7591 ;*          RXBEND - Label after end of the RX buffer area in memory.
7592 ;*          RXBFUL - Equated to the capacity of the RX buffer.
7593 ;*          RXBIPT - Pointer to next available input slot of RX buffer.
7594 ;*          RXBSTA - Label at start of RX buffer area in memory.
7595 ;*
7596 ;* OUTPUTS: RXBIPT - Updated to point to next input slot of RX buffer.
7597 ;*          RXBCNT - RX buffer character count (Incremented).
7598 ;*          TXENBM - Map of previous DUT TX.ENABLE states.
7599 ;*          CARRY - "Success" flag (Set if buffer is not full).
7600 ;*
7601 ;* CALLING SEQUENCE: Put the address of the label RXCHRS in the vector
7602 ;* location.
7603 ;*
7604 ;* COMMENTS: If the RX buffer is full upon entry, this routine aborts the
7605 ;* program.
7606 ;*
7607 ;* SUBORDINATE ROUTINES CALLED: RXIE0, TXDSBL.
7608 ;* *****
7609
7610 040570 010246 RXCHRS:: MOV R2, -(SP) ;SAVE CONTENTS OF GPR R2.
7611 040572 017702 141504 2$: MOV @RBUFA, R2 ;READ A CHARACTER FROM THE DUT RX FIFO.
7612 040576 100054 BPL 60$ ;EXIT THE ROUTINE IF THE DATA.VALID BIT IS CLR.
7613
7614 040600 023727 003066 000100 CMP RXBCNT, @RXBFUL ;COMPARE BUFFER COUNT WITH BUFFER CAPACITY.
7615 040606 103402 BLO 4$ ;SKIP ABORT IF BUFFER IS NOT FULL.
7616 040610 004737 032412 JSR PC, OOPS ;ABORT, MUST BE A PROGRAM BUG.
7617 040614 010277 142244 4$: MOV R2, @RXBIPT ;PUT THE CHAR IN THE BUFFER.
7618 040620 062737 000002 003064 ADD @2, RXBIPT ;UPDATE POINTER TO THE NEXT BUFFER SLOT.
7619 040626 023727 003064 003270 CMP RXBIPT, @RXBEND ;CHECK IF POINTER SHOULD WRAP AROUND.
7620 040634 103403 BLO 6$ ;SKIP WRAPAROUND IF POINTER IS NOT AT END.
7621 040636 012737 003070 003064 MOV @RXBSTA, RXBIPT ;WRAP INPUT POINTER AROUND.
7622
7623 040644 005237 003066 6$: INC RXBCNT ;COUNT THIS CHARACTER AS BEING IN THE BUFFER.
7624 040650 023727 003066 000030 CMP RXBCNT, @RXBDTX ;CHECK FOR BUFFER AT DISABLE TX LEVEL.
7625 040656 001016 BNE 8$ ;SKIP DISABLING TX IF BUFFER LEVEL NOT CORRECT.
7626 040660 005737 002656 TST TXDBLF ;CHECK STATE OF TX DISABLE FLAG.
7627 040664 100413 BMI 8$ ;BRANCH IF TRANSMISSION ALREADY DISABLED.
7628 040666 010546 MOV R5, -(SP) ;SAVE THE VALUE OF GPR R5.
7629 040670 012705 000377 MOV @MAPLNS, R5 ;SPECIFY THAT ALL LINES SHOULD BE AFFECTED.
7630 040674 004737 036352 JSR PC, TXDSBL ;CLEAR THE TX ENBLF'S FOR ALL LINES.
7631 040700 010537 002410 MOV R5, TXENBM ;SAVE PREVIOUS TX.ENABLE STATES IN STORAGE.

```

INTERRUPT SERVICE ROUTINE - RXCHRS

7632	040704	012605			MOV	(SP)+,R5	;RESTORE GPR R5.	
7633	040706	012737	100000	002656	MOV	#BIT15,TXDBLF	;PREVENT TX FROM BEING DISABLED AGAIN.	
7634								
7635	040714	023727	003066	000100	81:	CMP	RXBCNT,#RXBFUL	;CHECK FOR BUFFER FULL CONDITION.
7636	040722	103723				BLO	21	;LOOP TO READ ANOTHER CHAR IF BUFFER NOT FULL.
7637								
7638	040724	004737	035546			JSR	PC,RXIE0	;BUFFER IS FULL, DISABLE RX INTERRUPTS.
7639								
7640	040730	012602		601:	MCV	(SP)+,R2	;RESTORE R2 TO ITS SAVED VALUE.	
7641	040732	000002			RTI			

INTERUPT SERVICE ROUTINE

RXINPT -

SEQ 0198

```

7643 .SBTTL INTERUPT SERVICE ROUTINE - RXINPT -
7644 ;* *****
7645 ;* - Receive Character Input Interrupt Service Routine -
7646 ;* This service routine inputs a character from the DUT and loads the
7647 ;* char (complete with status flags) into a receive char buffer in
7648 ;* memory. The interrupt is also counted. The receive char buffer is
7649 ;* monitored to ensure that it does not overflow.
7650 ;*
7651 ;* INPUTS: BUFEND - Labels the end of the host memory buffer.
7652 ;*          BUFPTR - Contains address of next free buffer location.
7653 ;*          CSRA - Contains the address of the DUT CSR.
7654 ;*          RBUFA - Contains the address of the RBUF DUT register.
7655 ;*          RXINTC - Holds the count of the number of RX interrupts.
7656 ;*          RXINTF - RX Interrupt flags.
7657 ;*
7658 ;* OUTPUTS: BUFPTR - Contains updated address of next free buffer location.
7659 ;*          RXINTC - Contains the updated interrupt count.
7660 ;*          RXINTF - RX Int flags (bit 15 set if RX.DATA.AVAIL is clear).
7661 ;*
7662 ;* CALLING SEQUENCE: Put the address of the label RXINPT in the vector
7663 ;* location.
7664 ;*
7665 ;* COMMENTS: In case of overflow of the memory buffer, BUFPTR will be
7666 ;* maintained equal to BUFEND and the word at BUFPTR will be
7667 ;* the last word read from the DUT FIFO.
7668 ;* NOTE: This routine can destroy TX.ACTIONS by reading the CSR.
7669 ;*
7670 ;* SUBORDINATE ROUTINES CALLED: None.
7671 ;*
7672 ;* *****
7673 040734 RXINPT:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
7674 040734 004537 005474 JSR R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7675 040740 032777 000200 141332 BIT #BIT7,CSRA ;TEST RX.DATA.AVAIL BIT OF THE CSR (READS CSR).
7676 040746 001003 BNE 2$ ;BRANCH AROUND SETTING FLAG IF BIT IS SET.
7677 040750 052737 100000 002414 BIS #BIT15,RXINTF ;SET THE RX.DATA.AVAIL CLEAR FLAG.
7678 040756 013701 002412 2$: MOV RXINTC,R1 ;GET THE INTERRUPT COUNT.
7679 040762 C JCVL INC R1 ;INCREMENT THE COUNT.
7680 040764 001402 BEQ 4$ ;BYPASS UPDATING COUNT IF OVERFLOW OCCURRED.
7681 040766 010137 002412 MOV R1,RXINTC ;SAVE NEW COUNT VALUE.
7682 040772 013702 002360 4$: MOV BUFPTR,R2 ;GET THE POINTER TO NEXT FREE BUFFER WORD.
7683 040776 017722 141300 MOV @RBUFA,(R2)+ ;READ A CHAR FROM THE FIFO INTO BUFFER.
7684 041002 020237 005012 CMP R2,BUFND ;TEST FOR POINTER BEYOND END OF BUFFER.
7685 041006 103002 BHS 60$ ;SKIP THE PTR UPDATE IF PTR OUT OF BOUNDS.
7686 041010 010237 002360 MOV R2,BUFPTR ;UPDATE THE BUFFER POINTER.
7687 041014 004736 60$: PASS ;RESTORE GPRS.
7687 041016 000002 RTI JSR PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

```

TRAP SERVICE ROUTINE

- TP4BRT

SEQ 0199

```

7689 .SBTTL TRAP SERVICE ROUTINE - TP4BRT -
7690 ;*****
7691 ;* Bus Time-out Trap (004 trap) Service Routine -
7692 ;* This routine is used during the DMA ADDRESS TEST.
7693 ;* It determines if the 004 trap was caused by an "expected" error or
7694 ;* not by examining the return PC value on the stack. If the trap is
7695 ;* unexpected, this routine jumps to the normal Diagnostic Supervisor
7696 ;* 004 trap handling routine.
7697 ;*
7698 ;* INPUTS: SP - Points to the PC where the trap occurred.
7699 ;* TRPAD2 - Label at the address where "expected" traps occur.
7700 ;* TP4FLG - 004 trap flags.
7701 ;*
7702 ;* OUTPUTS: TP4FLG - Bit 15 is set if "expected" trap occurred.
7703 ;*
7704 ;* CALLING SEQUENCE: Put address pointed to by TP4BRT in 004 vector.
7705 ;* Occurrence of 004 trap vectors to this routine.
7706 ;*
7707 ;* COMMENTS: Any 004 trap which occurs at an address other than that labeled
7708 ;* TRPAD2 will be handled by the normal 004 trap service routine.
7709 ;* This routine is used in conjunction with CKTRPB subroutine.
7710 ;*
7711 ;* SUBORDINATE ROUTINES CALLED: None.
7712 ;*****
7713
7714 041020 021627 030154 TP4BRT:: CMP (SP),#TRPAD2 ;COMPARE EXPECTED ADDR WITH TRAP RET PC.
7715 041024 001402 BEQ 2$ ;IF THEY MATCH, CONTINUE THIS ROUTINE.
7716 041026 000177 141370 JMP @TP4VEC ;IF NOT, JUMP TO NORMAL 004 TRAP SERVICE RTN.
7717 041032 052737 100000 002424 2$: BIS @BIT15,TP4FLG ;SET THE 004 TRAP OCCURED FLAG.
7718 041040 000002 RTI ;ALL DONE, GO BACK TO THE TEST.

```

GLOBAL TRAP SERVICE ROUTINE - TP4RTN -

SEQ 0200

```

7720 .SBTTL GLOBAL TRAP SERVICE ROUTINE - TP4RTN -
7721 ;*****
7722 ;* Bus Time-out Trap (004 trap) Service Routine -
7723 ;* This routine is used during the Device Register Address Access Test.
7724 ;* It determines if the 004 trap was caused by an "expected" error or
7725 ;* not by examining the return PC value on the stack. If the trap is
7726 ;* unexpected, this routine jumps to the normal Diagnostic Supervisor
7727 ;* 004 trap handling routine.
7728 ;*
7729 ;* INPUTS: SP - Points to the PC where the trap occurred.
7730 ;* ADRPTR - Label at the address where "expected" traps occur.
7731 ;* TP4FLG - 004 trap flags.
7732 ;*
7733 ;* OUTPUTS: TP4FLG - Bit 15 is set if "expected" trap occurred.
7734 ;*
7735 ;* CALLING SEQUENCE: Put address pointed to by TP4RTN in 004 vector.
7736 ;* Occurrence of 004 trap vectors to this routine.
7737 ;*
7738 ;* COMMENTS: Any 004 trap which occurs at an address other than that labeled
7739 ;* ADRPTR will be handled by the normal 004 trap service routine.
7740 ;*
7741 ;* SUBORDINATE ROUTINES CALLED: None.
7742 ;*****
7743
7744 041042 021627 030124 TP4RTN:: CMP (SP),#ADRPTR ;COMPARE EXPECTED ADR AGAINST TRAP RET PC.
7745 041046 001402 BEQ 2$ ;IF THEY MATCH, CONTINUE THIS ROUTINE.
7746 041050 000177 141346 JMP @TP4VEC ;IF NOT, JUMP TO NORMAL 004 TRAP SERVICE RTN.
7747 041054 052737 100000 002424 2$: BIS #BIT15,TP4FLG ;SET THE 004 TRAP OCCURED FLAG.
7748 041062 000002 RTI ;ALL DONE, GO BACK TO THE TEST.

```

INTERRUPT SERVICE ROUTINE - TXDMA -

```

7750 .SBTTL INTERRUPT SERVICE ROUTINE - TXDMA -
7751 ;* *****
7752 ;* - DMA Transmit Interrupt Service Routine -
7753 ;* This routine executes in response to an interrupt caused by the DUT
7754 ;* TX.ACTION bit becoming active. This routine initiates the TX of a
7755 ;* new DMA buffer of characters or sets the TX Done Flag for the correct
7756 ;* line if TX is complete on that line.
7757 ;*
7758 ;* INPUTS: BITTBL - Label of table of words each with a bit set.
7759 ;*          CNCNTB - Base of # of chars to TX/RX table.
7760 ;*          CSRA - Contains the address of the DUT CSR.
7761 ;*          DPENDB - Base of the data pattern end table (entry per line).
7762 ;*          DPLENB - Base of the data pattern length table.
7763 ;*          IESTAT - Preserved states of the DUT interrupt enable bits.
7764 ;*          TXCNTB - Label at base of the TX character counter table.
7765 ;*          TXPTRB - Label at base of the TX data pattern pointers table.
7766 ;*
7767 ;* OUTPUTS: TXCNTx - Counters incremented for lines on which chars sent.
7768 ;*          TXDNF - TX done flags set for lines which have sent all chars.
7769 ;*          TXINTF - TX int flags (bit set if DMA.HO found set on line).
7770 ;*
7771 ;* CALLING SEQUENCE: Put the address of the label TXDMA in the vector
7772 ;* location.
7773 ;*
7774 ;* COMMENTS:
7775 ;*
7776 ;* SUBORDINATE ROUTINES CALLED: DODMA.
7777 ;* *****
7778
7779 041064 TXDMA:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
7780 041064 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7781 041070 017701 141204 ;MOV @CSRA,R1 ;READ THE CONTENTS OF THE DUT CSR.
7782 041074 010100 ;MOV R1,R0 ;SAVE INITIAL CONTENTS OF IND.ADR.REG FIELD.
7783 041076 000402 ;BR 4$ ;BRANCH TO SKIP DOUBL READING OF DUT CSR.
7784 ;*
7785 ; Read the contents of the DUT CSR. This will clear the TX.ACTION CSR bit.
7786 ; If TX.ACTION is not set, exit this routine.
7787 ; Determine the line for which the TX.ACTION was set.
7788 ; Calculate an offset for use in accessing tables (2 times the line number).
7789 ; Get the bit map of this line.
7790 041100 017701 141174 2$: MOV @CSRA,R1 ;READ THE CONTENTS OF THE DUT CSR.
7791 041104 100033 4$: BPL 60$ ;EXIT ROUTINE IF TX.ACTION IS CLEAR.
7792 041106 000301 ;SWAB R1 ;CALCULATE THE LINE NUMBER OF THE LINE WHICH IS
7793 041110 042701 177760 ;BIC @177760,R1 ; ASSOCIATED WITH THE TX.ACTION.
7794 041114 010104 ;MOV R1,R4 ;CALCULATE AN OFFSET FOR USE IN ACCESSING
7795 041116 006304 ;ASL R4 ; LINE COUNTER AND POINTER IN TABLES.
7796 041120 016405 002534 ;MOV BITTBL(R4),R5 ;GET THE BIT MAP OF THIS LINE.
7797 ;*
7798 ; Get the TX character counter for this line.
7799 ; If all the characters have been sent for this line:
7800 ; Set the TX done flag for this line.
7801 ; Don't send a char to the line (no more TX.ACTIONS on this line).
7802 ; Loop to check the TX.ACTION for another line.
7803 ;*
7804 041124 026464 003652 003612 ;CMP TXCNTB(R4),CNCNTB(R4) ;COMPARE # CHARS SENT AND TX COUNT.
7805 041132 103403 ;BLO 6$ ;GO TO SEND A CHAR IF NOT ALL CHARS SENT.

```

INTERRUPT SERVICE ROUTINE

- TXDMA -

```

7806 041134 050537 002652      BIS    R5,TXDONF      ;SET THIS LINE'S TX DONE FLAG.
7807 041140 000757              BR      2$      ;LOOP TO CHECK TX.ACTION AGAIN.
7808                               ;+
7809                               ; Start the DMA of the next buffer (data pattern) on this line.
7810                               ;   Get the data pattern length for this line.
7811                               ;   Get the start address of the data pattern.
7812                               ;-
7813 041142 016403 003352      6$:   MOV    DPLENB(R4),R3      ;PASS DATA PATTERN LENGTH FOR LINE TO DODMA.
7814 041146 016402 003512      MOV    TXPTRB(R4),R2      ;PASS THE TX START ADR TO DODMA.
7815                               ;+
7816                               ; Write DMA parameters to the DUT.
7817                               ;-
7818 041152 004737 030624      JSR     PC,DODMA
7819 041156 103403              BCS     8$      ;SKIP ERROR IF DODMA WAS SUCCESSFUL.
7820                               ;+
7821                               ; Set the proper bit of the TX interrupt flags to indicate the line error.
7822                               ;-
7823 041160 050537 002420      BIS     R5,TXINTF      ;INDICATE THE ERROR.
7824 041164 000402              BR      10$      ;SKIP UPDATING POINTERS AND COUNTERS.
7825                               ;+
7826                               ; Update the TX character for this line.
7827                               ; Update the TX buffer pointer for this line.
7828                               ;-
7829 041166 060364 003652      8$:   ADD     R3,TXCNTB(R4)      ;ADD THE DATA PAT LENGTH TO THE TX COUNT.
7830                               ;+
7831                               ; Loop to check the TX.ACTION bit for another line.
7832                               ;-
7833 041172 000742              10$:  BR      2$      ;LOOP BACK TO CHECK TX.ACTION BIT AGAIN.
7834                               ;+
7835 041174 013701 002374      60$:  MOV     IESTAT,R1      ;GET THE PRESENT STATES OF TX.IE & RX.IE BITS.
7836 041200 042700 177760      BIC     @177760,R0      ;GET SAVED IND.ADR.REG FIELD BITS.
7837 041204 050001              BIS     R0,R1      ;COMBINE IND.ADR.REG FIELD BITS WITH IE BITS.
7838 041206 010177 141066      MOV     R1,@CSRA      ;RESTORE THE DUT CSR IND.ADR.REG FIELD.
7839 041212              PASS      ;RESTORE GPRS.
7840 041214 000002              JSR     PC,@(SP)+      ;RETURN TO PREG05 SUBRT.
                               RTI

```

INTERUPT SERVICE ROUTINE

- TXINTR -

```

7842 .SBTTL INTERUPT SERVICE ROUTINE - TXINTR -
7843 ;** *****
7844 ;* - Transmit Interrupt Service Routine -
7845 ;* This routine handles a transmit interrupt from the Device Under Test
7846 ;* (DUT) by counting the interrupt and reading the DUT CSR to clear the
7847 ;* interrupt request. This routine also sets a flag to indicate that
7848 ;* a TX interrupt has occurred and sets a flag if the TX.ACTION bit is
7849 ;* not set in the read contents of the DUT CSR.
7850 ;*
7851 ;* INPUTS: CSRA - Contains the address of the CSR.
7852 ;* TXINTC - Holds the count of the number of TX interrupts.
7853 ;* TXINTF - TX Interrupt flags.
7854 ;*
7855 ;* OUTPUTS: TXINTC - Contains the updated TX interrupt count.
7856 ;* TXINTF - TX Int flags (bit 0 set, bit 15 set if TX.ACTION clr).
7857 ;*
7858 ;* CALLING SEQUENCE: Put the address of the label TXINTR in the vector
7859 ;* location.
7860 ;*
7861 ;* COMMENTS:
7862 ;*
7863 ;* SUBORDINATE ROUTINES CALLED: none
7864 ;-- *****
7865
7866 041216 TXINTR:: SAVE
7867 041216 004537 005474 JSR ;SAVE CONTENTS OF GPRS R0 THRU R5.
7868 041222 013701 002416 MOV TXINTC,R1 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7869 041226 005201 INC R1 ;GET THE TX INTERRUPT COUNT.
7870 041230 102001 BVC 2$ ;INCREMENT THE COUNT.
7871 041232 005301 DEC R1 ;BRANCH IF NO OVERFLOW OCCURRED.
7872 041234 010137 002416 2$: MOV R1,TXINTC ;RESET THE COUNT TO 177777.
7873 041240 013703 002420 MOV TXINTF,R3 ;SAVE NEW COUNT VALUE.
7874 041244 017702 141030 MOV @CSRA,R2 ;GET THE TX INTERRUPT FLAGS.
7875 041250 100402 BMI 4$ ;READ THE CSR.
7876 041252 052703 100000 BIS @BIT15,R3 ;SKIP SETTING OF FLAG IF TX.ACTION IS SET.
7877 041256 052703 000001 4$: BIS @BIT0,R3 ;SET THE TX.ACTION CLEAR FLAG.
7878 041262 010337 002420 MOV R3,TXINTF ;SET THE TX INT HAS OCCURRED FLAG.
7879 041266 004736 60$: PASS ;UPDATE THE TX INTERRUPT FLAGS.
7879 041270 000002 RTI ;RESTORE GPRS.
;PC,@(SP)+ ;RETURN TO PREG05 SUBRT.

```


INTERRUPT SERVICE ROUTINE

- TXSCHR -

SEQ 0204

```

7881 .SBTTL INTERRUPT SERVICE ROUTINE - TXSCHR -
7882 ;* *****
7883 ;* - Single Character Mode Transmit Interrupt Service Routine -
7884 ;* This routine executes in response to an interrupt caused by the DUT
7885 ;* TX.ACTION bit becoming active. This routine sends the next TX char
7886 ;* or sets the TX Done Flag for the correct line if TX is complete on
7887 ;* that line.
7888 ;*
7889 ;* INPUTS: ACTLNS - Bit map of active DUT lines.
7890 ;* BITTBL - Label of table of words each with a bit set.
7891 ;* CHCNTB - Base of # of chars to TX/RX table.
7892 ;* CSRA - Contains the address of the CSR.
7893 ;* DPENDB - Base of the data pattern end table (entry per line).
7894 ;* DPLENB - Base of the data pattern length table.
7895 ;* IACBIT - Bit mask of inactive TX/RX bits.
7896 ;* IBM - Inactive bits mask (reflecting bits per char).
7897 ;* TXCHRA - Contains the address of the DUT TXCHAR register.
7898 ;* TXCNTB - Label at base of TX character counters table.
7899 ;* TXPTRB - Label at the base address of the TX pointers table.
7900 ;*
7901 ;* OUTPUTS: CSR - DUT CSR IND.ADR.REG field is destroyed.
7902 ;* TXCHAR - DUT TXCHARs have words written to them.
7903 ;* TXCNTx - Counters incremented for lines on which chars sent.
7904 ;* TXDONF - TX done flags set for lines which have sent all chars.
7905 ;* TXPTRB - Each pointer in table points to next TX char for line.
7906 ;*
7907 ;* CALLING SEQUENCE: Put the address of the label TXSCHR in the vector
7908 ;* location.
7909 ;*
7910 ;* COMMENTS:
7911 ;*
7912 ;* SUBORDINATE ROUTINES CALLED: None.
7913 ;* -- *****
7914
7915 041272 TXSCHR:: SAVE ;SAVE CONTENTS OF GPRS R0 THRU R5.
7916 041272 004537 005474 ;R5,PREG05 ;CALL REGISTER SAVE SUBRT.
7917 041276 017701 140776 ;MOV @CSRA,R1 ;READ THE CONTENTS OF THE DUT CSR.
7918 041302 010105 ;MOV R1,R5 ;SAVE THE CONTENTS OF THE DUT IND.ADR.REG
7919 041304 042705 137660 ;BIC @137660,R5 ; FIELD FOR RESTORATION BEFORE RETURN.
7920 041310 010546 ;MOV R5,-(SP) ;SAVE THE DUT IN.ADD FIELD ON THE STACK.
7921 041312 005005 ;CLR R5 ;CLEAR CHARACTER TRANSMISSION COUNTER.
7922 041314 005701 ;TST R1 ;SET FLAG FOR TEST AFTER THE FOLLOWING BRANCH.
7923 041316 000402 ;BR 4$ ;GO HANDLE THE LINE THAT GOT THE TX.ACTION.
7924 ;*
7925 ; Read the contents of the DUT CSR. This will clear the TX.ACTION CSR bit.
7926 ; If TX.ACTION is not set, exit this routine.
7927 ; Determine the line for which the TX.ACTION was set.
7928 ; Calculate an offset for use in accessing tables (2 times the line number).
7929 ; Determine the states of the DUT CSR interrupt enable bits.
7930 041320 017701 140754 ;-
7931 041324 100051 2$: MOV @CSRA,R1 ;READ THE CONTENTS OF THE DUT CSR.
7932 041326 010102 4$: BPL 60$ ;EXIT ROUTINE IF TX.ACTION IS CLEAR.
7933 041330 000302 ;MOV R1,R2 ;CALCULATE THE LINE NUMBER
7934 041332 042702 177760 ;SWAB R2 ; OF THE LINE WHICH IS
7935 041336 010203 ;BIC @177760,R2 ; ASSOCIATED WITH THE TX.ACTION.
7936 041340 006303 ;MOV R2,R3 ;CALCULATE AN OFFSET FOR USE IN ACCESSING
; LINE COUNTER AND POINTER IN TABLES.

```

INTERRUPT SERVICE ROUTINE - TXSCHR -

```

7937 041342 042701 137677      BIC    #137677,R1      ;GET BIT MASK OF INTERRUPT ENABLE STATES.
7938
7939      ;*
7939      ; Get the TX character counter for this line.
7940      ; If all the characters have been sent for this line:
7941      ;   Set the TX done flag for this line.
7942      ;   Don't send a char to the line (no more TX.ACTIONS on this line).
7943      ;   Loop to check the TX.ACTION for another line.
7944      ;-
7945 041346 026363 003652 003612      CMP    TXCNTB(R3),CHCNTB(R3) ;COMPARE TX CHAR COUNT AND # TO TX.
7946 041354 103404                BLO    6$          ;GO ,0 SEND A CHAR IF NOT ALL CHARS SENT.
7947 041356 056337 002534 002652      BIS    BITTBL(R3),TXDNF      ;SET THIS LINE'S TX DONE FLAG.
7948 041364 000757                BR     4$          ;LOOP TO CHECK TX.ACTION AGAIN.
7949
7950      ;*
7950      ; Send the next char to the specified line.
7951      ;   Srt up the IND.ADR.REG field of the DUT CSR using the previously read
7952      ;   states of the interrupt enable bits.
7953      ;   Fetch the correct character from the data pattern.
7954      ;   Update the data pattern pointer for this line using wraparound.
7955      ;   Mask out inactive data bits and send the character.
7956      ;   Count the character on the TX char counter for the line.
7957      ;   Decrement the FIFO slack count to reserve room for this char in the FIFO.
7958      ;   Exit if a maximum of 8 characters have been transmitted, ie.give reception
7959      ;   a chance to remove characters from the FIFO.
7960      ;-
7961 041366 050201                6$:    BIS    R2,R1          ;SET UP THE IND.ADR.REG FIELD OF THE DUT CSR
7962 041370 010177 140704          MOV    R1,@CSRA      ; WITHOUT AFFECTING THE INTERRUPT ENABLES.
7963 041374 016304 003512          MOV    TXPTRB(R3),R4    ;FETCH THE TX POINTER FOR THIS LINE.
7964 041400 112400                MOV    (R4)+,R0        ;GET THE NEXT CHAR FOR THIS LINE.
7965 041402 020463 003312          CMP    R4,DPEMDB(R3)    ;COMPARE POINTER WITH END OF DATA PATTERN.
7966 041406 103402                BLO    8$          ;SKIP RESETTING OF POINTER IF NOT PAST END.
7967 041410 166304 003352          SUB    DPLENB(R3),R4    ;WRAP POINTER AROUND TO BEGINNING OF PATTERN.
7968 041414 010463 003512          8$:    MOV    R4, TXPTRB(R3) ;UPDATE THE TX POINTER FOR THIS LINE.
7969 041420 043700 002366          BIC    IBM,R0          ;CLEAR UNUSED BITS OF THE TX CHAR WORD.
7970 041424 052700 100000          BIS    #BIT15,R0      ;SET THE TX.DATA.VALID BIT OF TX CHAR WORD.
7971 041430 010077 140646          MOV    R0,@TXCHA      ;SEND THE CHAR TO THE DUT.
7972 041434 005205                INC    R5              ;INCREMENT TX CHAR COUNT.
7973 041436 005263 003652          INC    TXCNTB(R3)      ;INCREMENT THE TX CHAR COUNT FOR THIS LINE.
7974
7975      ;*
7975      ; Loop to check the TX.ACTION bit for another line.
7976      ;-
7977 041442 020527 000010          CMP    R5,#NUMLNS      ;CHECK IF MAX NUMBER OF CHAR HAVE BEEN TX'D.
7978 041446 103724                BLO    2$          ;LOOP BACK TO CHECK TX.ACTION BIT AGAIN.
7979 041450 012677 140624          60$:    MOV    (SP)+,@CSRA    ;RESTORE THE IND.ADR.REG FIELD OF THE DUT CSR.
7980 041454                PASC                ;RESTORE GPRS.
7981 041456 000002                RTI                    ;RETURN TO PREG05 SUBRT.

```


INTERRUPT SERVICE ROUTINE - TXSCHR -

```
7983
7984 ;*****
** 7985 ;
7986 ; VDHE.RPT
7987 ;
7988 ;*****
7989
7990
7991
7992 .SBTTL REPORT CODING SECTION
7993
7994 ;**
7995 ; THE REPORT CODING SECTION CONTAINS THE
7996 ; "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
7997 ;--
7998
7999 041460 BGNRPT
      041460
8000
8001 041460 EXIT RPT
      041460 000167
      041462 000000
8002
8003
8004
8005 041464
      041464
      041464 104425

                                L$RPT::
                                .WORD J$JMP
                                .WORD L10025-2-.

                                L10025:
                                TRAP C$RPT
```

PROTECTION TABLE

```

8007      .SETTL  PROTECTION TABLE
8008
8009      ;*****
8010      ;
8011      ;           FVTSKL4.P11
8012      ;
8013      ;*****
8014
8015
8016
8017      ;**
8018      ; THIS TABLE IS USED BY THE RUNTIME SERVICES
8019      ; TO PROTECT THE LOAD MEDIA.
8020      ;--
8021
8022 041466      BGNPROT
8023 041466
8024      L#PROT::
8024 041466 177777      -1      ;OFFSET INTO P-TABLE FOR CSR ADDRESS
8025 041470 177777      -1      ;OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
8026 041472 177777      -1      ;OFFSET INTO P-TABLE FOR DRIVE NUMBER
8027
8028 041474      ENDPROT
8029

```


INITIALIZE SECTION

```

      041550 010001
8085 041552 012137 002440      MOV      (R1)+,CLKCSR      ;STORE CLOCK CSR ADDRESS.
8086 041556 012137 002442      MOV      (R1)+,CLKBRL      ;STORE CLOCK BUS REQ INT LEVEL.
8087 041562 012137 002444      MOV      (R1)+,CLKVEC      ;STORE CLOCK INTERRUPT VECTOR.
8088 041566 012137 002446      MOV      (R1)+,CLKHRZ      ;STORE CLOCK FREQUENCY.
8089 041572 023727 002446 000062      CMP      CLKHRZ,#50.      ;TEST FOR 50HZ LINE FREQUENCY.
8090 041600 001004      BNE      2$      ;BRANCH IF CLOCK IS NOT 50HZ.
8091 041602 012737 000024 002460      MOV      #20.,MSTICK      ;INDICATE 20MS PER CLOCK TICK.
8092 041610 000403      BR      4$
8093 041612 012737 000021 002460 2$:      MOV      #17.,MSTICK      ;INDICATE 17 MS PER CLOCK TICK.
8094 041620 4$:      SETVEC      CLKVEC,#CLKINT,#PRI06      ;INITIALIZE CLOCK INTERRUPT VECTOR.
      041620 012746 000300      MOV      #PRI06,-(SP)
      041624 012746 040430      MOV      #CLKINT,-(SP)
      041630 013746 002444      MOV      CLKVEC,(SP)
      041634 012746 000003      MOV      #3,(SP)
      041640 104437      TRAP      C$SVEC
      041642 062706 000010      ADD      #10,SP
8095 041646 013700 002446      MOV      CLKHRZ,R0      ;INITIALIZE THE BREAK COUNT
8096 041652 006200      ASR      R0      ; TO CAUSE A BREAK
8097 041654 010037 002456      MOV      R0,BCOUNT      ; EVERY 1/2 SECOND.
8098 041660 106427 000240      MTPS      #PRI05      ;ALLOW CLOCK INTERRUPTS DISABLE OTHERS.
8099
8100      ; Enable the Line Time Clock (LTC) checking to make sure that the CSR
8101      ; is accessible.
8102      ; First set up to catch any 004 traps which occur:
8103      ;-
8104 041664 013737 000004 002422      MOV      #04,TP4VEC      ;SAVE THE EXISTING 004 TRAP VECTOR.
8105 041672 012737 041042 000004      MOV      #TP4RTN,#04      ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
8106
8107      ; Enable LTC checking for 004 trap in case CSR is not there.
8108      ;-
8109 041700 005037 002424      CLR      TP4FLG      ;CLEAR THE 004 TRAP FLAG.
8110 041704 012737 000100 002400      MOV      #BIT6,WORD1      ;SET UP TO SET BIT6 OF THE LTC CSR.
8111 041712 012700 002400      MOV      #WORD1,R0      ;SET UP WORD1 AS THE CKTRAP MOVE SOURCE.
8112 041716 013701 002440      MOV      CLKCSR,R1      ;SET UP LTC CSR AS DESTINATION FOR CKTRAP MOVE.
8113 041722 004737 030112      JSR      PC,CKTRAP      ;MOVE AND CHECK FOR TRAP.
8114 041726 013737 002422 000004      MOV      TP4VEC,#04      ;RESTORE THE NORMAL 004 TRAP VECTOR.
8115 041734 103403      BCS      6$      ;IF NO TRAP, LTC IS THERE SO CONTINUE.
8116 041736 005037 002446      CLR      CLKHRZ      ;CLEAR LTC FREQUENCY WORD TO INDICATE NO LTC.
8117 041742 000402      BR      8$      ;BYPASS THE FOLLOWING CALIBRATION PROCEDURES.
8118
8119      ; Calibrate the DELAY routine milli second delay count value.
8120      ;-
8121 041744 004737 027032      6$:      JSR      PC,CALMSL
8122
8123      ; Check for Memory Management present on th's machine.
8124      ; If MEM MGT is present, disable it.
8125      ;-
8126 041750 013737 000004 002422 8$:      MOV      #04,TP4VEC      ;SAVE THE EXISTING 004 TRAP VECTOR.
8127 041756 012737 041042 000004      MOV      #TP4RTN,#04      ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
8128 041764 005037 002424      CLR      TP4FLG      ;CLEAR THE 004 TRAP FLAG.
8129 041770 005037 002400      CLR      WORD1      ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
8130 041774 012700 002400      MOV      #WORD1,R0      ;SELECT CLEARED WORD AS CKTRAP RTN SOURCE.
8131 042000 013701 002464      MOV      #MSRO,R1      ;SELECT MEM MGT SRO REGISTER AS DESTINATION.
8132 042004 005037 002470      CLR      #MPRES      ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
8133 042010 005037 002472      CLR      #MENAB      ;INDICATE MEM MGT IS NOT ENABLED.
8134 042014 004737 030112      JSR      PC,CKTRAP      ;CLEAR THE MEM MGT SRO REG AND CHECK FOR TRAP.

```

INITIALIZE SECTION

```

8135 042020 013737 002422 000004      MOV    TP4VEC,004      ;RESTORE THE NORMAL 004 TRAP VECTOR.
8136 042026 103003                    BCC    10$      ;SKIP INDICATING MEM MGT PRESENT IF IT ISN'T.
8137 042030 012737 000001 002470      MOV    01,MMPRES      ;INDICATE THAT MEM MGT IS PRESENT.
8138 042036 005037 002376      10$: CLR    PASCNT      ;CLR COUNTER USED IN REPORTING ROM VERSION #.
8139 042042 000137 042054      JMP     NEWPAS      ;SKIP AROUND THE BUS RESET, IT'S BEEN DONE.
8140
8141 042046                    NEWRES: BRESET      ;RESET THE BUS TO PREVENT ILLEGAL INTERRUPTS.
      042046 104433                    TRAP    C$RESET
8142 042050 005037 002376      CLR    PASCNT      ;CLR COUNTER USED IN REPORTING ROM VERSION #.
8143
8144 042054                    NEWPAS:
8145 042054 012737 177777 002276      MOV    0-1,UNITN      ;RESET LOGICAL DEVICE TO -1
8146
8147      ;+
8148      ; Increment the pass counter, correct for any overflow.
8149      ; This counter is used in the Rom version test.
8150 042062 005237 002376      INC     PASCNT      ;INCREMENT THE PASS COUNTER.
8151 042066 001002                    BNE     GETPRM      ;BRANCH IF WE HAVE NOT YET! OVERFLOWED.
8152 042070 005337 002376      DEC     PASCNT      ;SET PASS COUNT TO 177777 OCTAL.
8153
8154      ; GET THE HARDWARE PARAMETERS FOR THIS UNIT.
8155 042074                    GETPRM:
8156 042074 005237 002276      INC     UNITN      ;INCREMENT LOGICAL DEVICE NUMBER
8157 042100 023737 002276 002012      CMP     UNITN,L$UNIT      ;SEE IF MAXIMUM UNIT NO. EXCEEDED
8158 042106 002362                    BGE     NEWPAS      ;BR IF YES
8159
8160 042110                    GPHARD  UNITN,R1      ;GET P-TABLE POINTER INTO R1
      042110 013700 002276
      042114 104442
      042116 010001
8161 042120                    BCOMPLETE 30$      ;BR IF DEVICE AVAILABLE
      042120 103401
8162 042122 000764      BR     GETPRM      ;SKIP THIS DEVICE
8163
8164
8165      ;***** HARDWARE PARAMETER MOVING CODE *****
8166 042124 012137 002300      30$: MOV    (R1)+,CSRA      ;STORE DHV11-M CSR ADDRESS IN DEV.REG.ADDRESS TABLE
8167 042130 012102      MOV    (R1)+,R2      ;GET THE RX INTERRUPT VECTOR ADDRESS.
8168 042132 010237 002266      MOV    R2,RXVECA      ;STORE RX INT VECTOR ADDRESS.
8169 042136 062702 000004      ADD     04,R2      ;CALCULATE TX INTERRUPT VECTOR ADDRESS.
8170 042142 010237 002270      MOV    R2,TXVECA      ;STORE TX INT VECTOR ADDRESS.
8171 042146 012102      MOV    (R1)+,R2      ;REMOVE DATA FROM EMPTY P-TABLE SLOT.
8172 042150 012737 000377 002272      MOV    0MAPLNS,ACTLNS      ;INDICATE THAT ALL POSSIBLE LINES ARE ACTIVE.
8173 042156 112102      MOV     (R1)+,R2      ;REMOVE DATA FROM EMPTY P-TABLE SLOT.
8174 042160 112737 000001 002274      MOV     01,LOPBCK      ;FORCE INTERNAL LOOPBACK MODE
8175 042166 112137 002275      MOV     (R1)+,BRLEVL      ;STORE DHV11-M INTERRUPT BUS REQUEST LEVEL
8176
8177      ;+
8178      ; CALCULATE DEVICE REGISTER ADDRESSES,AND PUT THEM IN THE
8179      ; DEVICE REGISTER ADDRESS TABLE.
8180 042172 013701 002300      MOV     CSRA,R1      ;COPY CSR ADDRESS
8181 042176 005201      INC     R1      ;INCREMENT CSR ADDRESS
8182 042200 005201      INC     R1      ; COPY BY 2.
8183 042202 012703 000007      MOV     07,R3      ;SET UP REGISTER COUNT
8184 042206 012702 002302      MOV     0RBUFA,R2      ;GET LOCATION WHERE RBUF ADDRESS GOES IN TABLE
8185 042212 010122      12$: MOV    R1,(R2)+      ;STORE REGISTER ADDRESS IN TABLE
8186 042214 005201      INC     R1      ;INCREMENT REGISTER ADDRESS

```

INITIALIZE SECTION

```

8187 042216 005201          INC    R1          ; BY 2, FOR THE NEXT DEVICE REGISTER.
8188 042220 005303          DEC    R3          ; DECREMENT REGISTER COUNT
8189 042222 001373          BNE    12$         ; LOOP IF NOT DONE
8190
8191
8192          ;+
8193          ; Initialise the BMP code queue.
8194 042224 012700 002662          MOV    @BMPQCB,R0      ; GET THE START ADDRESS OF THE QUEUE.
8195 042230 012701 003062          MOV    @BMPQCE,R1      ; GET THE END ADDRESS OF THE QUEUE.
8196 042234 010037 002660          MOV    R0,BMPQCP      ; SET THE POINTER TO THE START OF THE QUEUE.
8197 042240 005020          14$: CLR    (R0)+          ; CLEAR OUT THE CONTENTS OF THE QUEUE.
8198 042242 020001          CMP    R0,R1          ; CHECK IF END OF QUEUE HAS BEEN REACHED.
8199 042244 103775          BLO    14$           ; LOOP IF NOT ALL DONE.
8200 042246
8201
8202 042246 005037 002362          ENDIT: CLR    CTRLCF      ; CLR THE CTRL-C TEST ABORT FLAG.
8203
8204          ;+
8205          ; Set the processor priority to disable all but LTC interrupts.
8206 042252 106427 000240          MTPS    @PRI05          ; SET PROCESSOR PRIORITY TO 5.
8207
8208          ;+
8209          ; Enable Line Time Clock if one is available.
8210 042256 005737 002446          TST    CLKHRZ          ; CHECK FOR A LTC BEING PRESENT.
8211 042262 001403          BEQ    18$           ; LTC PRESENT? NO, SKIP LTC ENABLE.
8212 042264 012777 000100 140146          MOV    @BIT6,@CLKCSR      ; YES, ENABLE THE LTC.
8213 042272
8214
8215 042272          ENDINIT
8216 042272
8217 042272 104411          L10027: TRAP    C$INIT
8218          TNUM == 0          ; INITIALIZE THE ASSEMBLER TEST NUMBER VARIABLE.

```


INITIALIZE SECTION

```

8220 ;*****
8221 ;
8222 ;           VDHE.ATD
8223 ;
8224 ;*****
8226 ;
8227 ;
8228 .SBTTL AUTODROP SECTION
8229
8230
8231 ;++
8232 ; THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
8233 ; THE "ADR" FLAG WAS SET. THE UNIT(S) UNDER TEST ARE CHECKED TO
8234 ; SEE IF THEY WILL RESPOND. THOSE THAT DON'T ARE IMMEDIATELY
8235 ; DROPPED FROM TESTING.
8236 ;--
8237
8238 042274          BGNAUTO
8239 042274
8239
8246
8247 042274          ENDAUTO
8247 042274
8247 042274 104461

```

L\$AUTO::

L10030: TRAP C\$AUTO

AUTODROP SECTION

```

8249
8250
8251 ;*****
8252 ;
8253 ;               VDHE.CUC
8254 ;*****
8255
8256
8257
8258 .SBTTL  CLEANUP CODING SECTION
8259
8260 ;**
8261 ; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
8262 ; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
8263 ;--
8264
8265 042276      BGNCLN
8266 042276
8266
8266 L$CLEAN::
8275
8276 042276 005737 002362      TST      CTRLCF      ;DID WE GET HERE BY CTRL-C FROM TEST?
8277 042302 001401      BEQ      2$      ;CTRL-C FROM TEST? NO, SKIP BUS RESET.
8278 042304      BRESET      ;YES, CLR ANY DMAS OR OUTSTANDING INTERRUPTS.
8279 042304 104433      TRAP      C$RESET
8279 042306
8280 042306 005737 002446      2$:      TST      CLKHRZ      ;SEE IF CLOCK WAS ENABLED
8281 042312 001402      BEQ      3$      ;IF NOT BRANCH
8282 042314 005077 140120      CLR      @CLKCSR
8283 042320      3$:      EXIT      CLN
8284 042320
8284 042320 104432      TRAP      C$EXIT
8284 042322 000002      .WORD      L10031-.
8285
8297
8298 .EVEN
8299
8300 042324      ENDCLN
8300 042324
8300 042324 104412      L10031:      TRAP      C$CLEAN

```


CLEANUP CODING SECTION

```

8302
8303 ;*****
8304 ;
8305 ;           VDHE.DRP
8306 ;
8307 ;*****
8308
8309
8310
8311 .SBTTL  DROP UNIT SECTION
8312
8313 ;**
8314 ; THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
8315 ; TO NO LONGER BE TESTED.
8316 ;--
8317
8318 042326          BGNDU
8319 042326
8319
8320 ;*****
8321 ;           INSERT DROP CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
8322 ;           A "DROP" COMMAND OR A "DODU" MACRO EXECUTION.  THE PURPOSE
8323 ;           OF THIS CODE IS TO DO ANY NECESSARY HOUSEKEEPING AFTER A
8324 ;           UNIT HAS BEEN DROPPED.  THIS SECTION IS OPTIONAL.
8325 ;*****
8326 042326          PRINTF  *DROP,RO          ;REPORT UNIT THAT HAS BEEN DROPPED.
8326 042326 010046          MOV      RO,-(SP)
8326 042330 012746 042352          MOV      *DROP,-(SP)
8326 042334 012746 000002          MOV      *2,-(SP)
8326 042340 010600          MOV      SP,RO
8326 042342 104417          TRAP     C$PNTF
8326 042344 062706 000006          ADD      *6,SP
8327 042350 000427          BR      EDROP          ;BRANCH AROUND THE MESSAGE.
8328
8329 042352          045      101      040  DROP:  .ASCIZ/*A UNIT*D6*A DROPPED FROM FURTHER TESTING.*N/
8329 042355          125      116      111
8329 042360          124      045      104
8329 042363          066      045      101
8329 042366          040      104      122
8329 042371          117      120      120
8329 042374          105      104      040
8329 042377          106      122      117
8329 042402          115      040      106
8329 042405          125      122      124
8329 042410          110      105      122
8329 042413          040      124      105
8329 042416          123      124      111
8329 042421          116      107      056
8329 042424          045      116      000
8330
8331 042430          EDROP:  .EVEN
8332
8333 042430          EXIT      DU
8333 042430 000167          .WORD      J$JMP
8333 042432 000000          .WORD      L10032-2-.
8334
8335

```

DROP UNIT SECTION

8336 042434
042434
042434 104453

ENDDU

L10032: TRAP C#DU

DROP UNIT SECTION

```

8338
8339
8340 ;*****
8341 ;
8342 ;           VDHE.ADD
8343 ;
8344 ;*****
8345
8346
8347 .SBTTL  ADD UNIT SECTION
8348
8349 ;**
8350 ; THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
8351 ; TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
8352 ; TO THE TEST CYCLE.
8353 ;
8354
8355 042436      BGNAU
      042436
8356
8357
8358 ;*****
8359 ;           INSERT ADD CODE HERE.  THIS CODE WILL BE EXECUTED AFTER
8360 ;           AN "ADD" COMMAND.  THE PURPOSE OF THIS CODE IS TO DO ANY
8361 ;           HOUSEKEEPING THAT MAY BE NECESSARY AFTER A UNIT HAS BEEN ADDED.
8362 ;           THIS SECTION IS OPTIONAL.
8363 ;*****
8364 042436      EXIT    AU
      042436      000167
      042440      000000
8365
8366
8367
8368
8369 042442
      042442
      042442      104452
8370

```

L\$AU::

.WORD J\$JMP
.WORD L10033-2-.

L10033: TRAP C\$AU

HARDWARE TEST

- ADRA -

SEQ 0217

```

8372 .SBTTL  HARDWARE TEST          - ADRA -
8373 ;++
8374 ;*****
8375 ;
8376 ;               - REGISTER ADDRESS TEST -
8377 ;
8378 ;       This test verifies that the Q-bus can read and write to the DHV11
8379 ;       device registers.  If the DHV11 does not respond to the access
8380 ;       attempts (If the DHV11 is at the wrong address, for example) the
8381 ;       004 bus time-out trap is detected by this routine and an error
8382 ;       is reported.
8383 ;
8384 ;*****
8385 ;--
8386 BGNTST
8387
8388 042444 000001          TNUM == TNUM + 1          T1::
8389 042452 012737 000001 002364      MOV  #TNUM,ISTNUM      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8390 042452 012737 177777 002362      MOV  #1,CTRLCF        ;SET UP THE TEST NUMBER. (1)
8391 ;                               ;INDICATE THAT WE ARE IN A TEST.
8392 ; Set up to catch any 004 traps which occur:
8393 042460 013737 000004 002422      MOV  #04,TP4VEC        ;SAVE THE EXISTING 004 TRAP VECTOR.
8394 042466 012737 041042 000004      MOV  #TP4RTN,#04      ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
8395 042474 005005          CLR  R5                        ;CLEAR THE ERROR FLAGS.
8396
8397 ;
8398 ; Set up for the initial iteration of the test loop:
8399
8400 042476 005004          CLR  R4                        ;CLEAR THE LINE COUNTER.
8401
8402 ;
8403 ; Here begins the loop to test the registers for a line.
8404 ; First test the CSR and set the IND.ADR.REG (I.A.R) field.
8405
8406 042500 005037 002424      2$: CLR  TP4FLG            ;CLEAR THE 004 TRAP FLAG.
8407 042504 013700 002300      MOV  CSRA,R0              ;SET UP CSR AS THE CKTRAP MOVE SOURCE.
8408 042510 012701 042724      MOV  #52$,R1              ;SET UP DESTINATION LOCATION FOR CKTRAP MOVE.
8409 042514 004737 030112      JSR  PC,CKTRAP            ;MOVE AND CHECK FOR TRAP.
8410 042520 103402          BCS  4$                      ;IF NO TRAP, BYPASS ERROR.
8411 042522 052705 100001      BIS  #100001,R5           ;SET FATAL READ ERROR FLAGS.
8412 042526 042737 000017 042724 4$: BIC  #17,52$        ;CLEAR THE I.A.R FIELD OF THE CSR DATA.
8413 042534 050437 042724      BIS  R4,52$              ;OR IN THE LINE COUNTER TO THE I.A.R FIELD.
8414 042540 010100          MOV  R1,R0                  ;USE OLD DESTINATION FOR SOURCE OF CKTRAP MOVE.
8415 042542 013701 002300      MOV  CSRA,R1              ;SET UP CSR AS THE CKTRAP MOVE DESTINATION.
8416 042546 004737 030112      JSR  PC,CKTRAP            ;MOVE AND CHECK FOR TRAP.
8417 042552 103403          BCS  6$                      ;IF NO TRAP, BYPASS ERROR.
8418 042554 052705 100002      BIS  #100002,R5           ;SET FATAL WRITE ERROR FLAGS.
8419 042560 000440          BR   40$                     ;EXIT AND REPORT FATAL ERROR.
8420
8421 ;
8422 ; Now, we test each register for this line.
8423
8423 042562 012702 000010      6$: MOV  #10,R2            ;INIT REGISTER COUNTER TO 8.
8424 042566 013737 002300 042722      MOV  CSRA,50$        ;INITIALIZE THE REGISTER POINTER.
8425 042574 012700 042722      8$: MOV  #50$,R0           ;SET UP REGISTER AS THE SOURCE FOR CKTRAP MOVE.
8426 042600 012701 042724      MOV  #52$,R1              ;SET UP LOCAL STORAGE AS THE DES FOR CKTRAP.
8427 042604 004737 030112      JSR  PC,CKTRAP            ;PERFORM THE MOVE, CHECK FOR TRAP.

```

HARDWARE TEST

- ADRA -

```

8428 042610 103402          BCS 10$          ;IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.
8429 042612 052705 100001    BIS #100001,R5 ;SET FATAL READ ERROR FLAGS.
8430 042616 010100          MOV R1,R0        ;USE OLD DEST AS SRC FOR CKTRAP MOVE.
8431 042620 012701 042722    10$: MOV #50$,R1 ;SET UP REGISTER AS THE DEST FOR CKTRAP MOVE.
8432 042624 004737 030112    JSR PC,CKTRAP    ;PERFORM THE MOVE, CHECK FOR TRAP.
8433 042630 103402          BCS 12$          ;IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.
8434 042632 052705 100002    BIS #100002,R5 ;SET FATAL WRITE ERROR FLAGS.
8435 042636 005237 042722    12$: INC 50$    ;INCREMENT THE REGISTER
8436 042642 005237 042722    INC 50$        ; POINTER BY 2.
8437 042646 005302          DEC R2           ;COUNT THE REGISTER.
8438 042650 001351          BNE 8$           ;LOOP TO TEST THE NEXT REGISTER ADDRESS.
8439
8440          ;+
8441          ; Now we set up to test the next line, or to exit if we are done.
8442          ;-
8443 042652 005204          INC R4            ;INCREMENT THE LINE COUNTER.
8444 042654 020427 000010    CMP R4,#NUMLNS   ;COMPARE LINE COUNTER AGAINST NUMBER OF LINES.
8445 042660 002707          BLT 2$           ;LOOP TO TEST THE NEXT LINE IF WE'RE NOT DONE.
8446
8447          ;+
8448          ; Done checking device register addresses.
8449          ; Report any errors and exit.
8450          ;-
8451 042662 013737 002422 000004 40$: MOV TP4VEC,8$4 ;RESTORE THE NORMAL 004 TRAP VECTOR.
8452 042670 005705          TST R5           ;CHECK THE ERROR FLAGS.
8453 042672 100015          BPL 60$         ;EXIT ROUTINE IF NO ERRORS.
8454          ; REPORT "DEVICE REGISTER ACCESS ERRORS"
8455 042674          ERDF 101,EM0103,ER0101; >>>> ERROR #101 <<<<.
      042674 104455          TRAP C$ERDF
      042676 000145          .WORD 101
      042700 010727          .WORD EM0103
      042702 024240          .WORD ER0101
8456
8457 042704          DODU UNITN              ;DROP THIS UNIT FROM FUTHER TESTING.
      042704 013700 002276          MOV UNITN,R0
      042710 104451          TRAP C$DODU
8458 042712 005037 002362          CLR CTRLCF ;INDICATE NO CTRL-C ABORT FROM TEST.
8459 042716          DOCLN                 ;ABORT THIS SUB PASS.
      042716 104444          TRAP C$DCLN
8460 042720 000402          BR 60$          ;
8461          ;+
8462          ; Local storage.
8463          ;-
8464 042722 000000          50$: .WORD 0      ;STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.
8465 042724 000000          52$: .WORD 0      ;STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.
8466 042726 005037 002362          60$: CLR CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.
8467 042732          ENDTST
      042732          L10034:
      042732 104401          TRAP C$ETST

```

HARDWARE TEST

- MRSTA -

```

8469 .SBTTL HARDWARE TEST - MRSTA -
8470 ;++ *****
8471 ;* - Master Reset With Selftest Test -
8472 ;* This test verifies that the Master Reset bit will clear after a device
8473 ;* reset and the performance of the DUT ROM based selftest.
8474 ;*
8475 ;-- *****
8476 BGNTST
8477
8478 000002 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8479 042734 012737 000002 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (2)
8479 042742 012737 177777 002362 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
8480 042750 012737 000001 005464 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
8481 042756 012737 010765 005470 MOV #EM0201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
8482 042764 012737 024562 005472 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
8483
8484 ;+
8485 ; Wait up to 5 seconds for the DUT Master Reset bit to clear.
8486 042772 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
8487 042776 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
8488 043002 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
8489 043004 013704 002300 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
8490 043010 004737 031726 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8491 043014 103410 BCS 2# ;SKIP TO RESET DUT IF MR CLEAR.
8492
8493 ;+
8494 ; DUT Master Reset bit did not go clear. Device may be stuck in some
8495 ; odd state. Try to reset device with a bus reset.
8496 043016 BRESET ;NO, TRY TO JOG DEVICE WITH BUS RESET.
8497 043020 004737 035740 JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST. TRAP C#RESET
8498 043024 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
8499 043030 004737 031726 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8500 043034 103016 BCC 4# ;GO REPORT ERROR IF MR BIT DID NOT CLEAR.
8501
8502 ;+
8503 ; Set the Master Reset bit and verify that it clears within the proper time.
8504 043036 012701 011610 2# MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
8505 043042 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
8506 043044 004737 031726 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8507 043050 103010 BCC 4# ;GO REPORT ERROR IF MR BIT DID NOT CLEAR.
8508 043052 012702 011610 MOV #5000.,R2
8509 043056 160102 SUB R1,R2 ;CALCULATE # OF MS FOR MR TO CLEAR.
8510 043060 001413 BEQ 6# ;GO REPORT ERROR IF MR CLEAR IMMEDIATELY.
8511 043062 020227 000764 CMP R2,#500.
8512 043066 002417 BLT 8# ;GO REPORT ERROR IF MR CLEAR IN < 1/2 SECOND.
8513 043070 000424 BR 60# ;EXIT THE TEST WITHOUT ERROR.
8514
8515 ;+
8516 ; Error reports:
8517 ;--
8518 043072 012737 000311 005466 4# ;Report MR bit would not clear after a DUT reset.
8519 043100 012701 011033 MOV #201.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
8520 043104 104460 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
8521 043106 000415 ERROR ;REPORT ERROR. >>>> ERROR #201 <<<<
8522 TRAP C#ERROR
      BR 60# ;EXIT THE TEST.

```


HARDWARE TEST

- MRSTA -

```

8523
8524 043110 012737 000312 005466 6+: ;Report MR bit clear immediately after DUT reset.
8525 043116 012701 011206      MOV    #202.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
8526 043122      MOV    #EM0203,R1 ;SELECT ERROR MESSAGE.
      ERROR      ;REPORT ERROR.          >>>>> ERROR #202 <<<<<
      BR        60+      ;EXIT THE TEST.          TRAP    C#ERROR
8527 043124 000406
8528
8529
8530 043126 012737 000313 005466 8+: ;Report MR clear within 1/2 second of DUT reset.
8531 043134 012701 011351      MOV    #203.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
8532 043140      MOV    #EM0204,R1 ;SELECT ERROR MESSAGE.
      ERROR      ;REPORT ERROR.          >>>>> ERROR #203 <<<<<
      BR        60+      ;EXIT THE TEST.          TRAP    C#ERROR
8533
8534 043142 005037 002362      60+: CLR    CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
8535
8536 043146      ENDTST
      043146
      043146 104401
      L10035:
      TRAP    C#ETST

```

HARDWARE TEST

- MRSSTA

8538
8539
8540
8541
8542
8543
8544
8545 043150
043150

8546 000003
8547 043150 012737 000003 002364
8548 043156 012737 177777 002362
8549 043164 012737 000001 005464
8550 043172 012737 011530 005470
8551 043200 012737 024562 005472

8552
8553
8554
8555 043206 012701 011610
8556 043212 012702 000040
8557 043216 005003
8558 043220 013704 002300
8559 043224 004737 031726
8560 043230 103410

8561
8562
8563
8564
8565 043232
043232 104433
8566 043234 004737 035740
8567 043240 012701 011610
8568 043244 004737 031726
8569 043250 103024

8570
8571
8572
8573
8574 043252 012701 000310
8575 043256 010214
8576 043260 004737 035740
8577 043264 004737 031726
8578 043270 103007
8579 043272 012702 000310
8580 043276 160102
8581 043300 020227 000012
8582 043304 002415
8583 043306 000431

8584
8585
8586
8587 043310 012701 011300
8588 043314 004737 031726
8589 043320 103416
8590
8591
8592

.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.
TRAP C\$RESET
JSR PC,SKPSTS ;TRY TO SKIP THE SELFTEST.
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:

;

- MRSSTA -

Address	Hex	Dec	Hex	Dec	Label	Instruction	Comment
8593							
8594	043322	012737	000455	005466	6\$:	MOV #0301.,ERRNBR	;Report MR bit would not clear after a DUT reset. ;SET THE ERROR NUMBER IN ERROR TABLE.
8595	043330	012701	011033			MOV #EM0202,R1	;SELECT ERROR MESSAGE.
8596	043334					ERROR	;REPORT ERROR. >>>> ERROR #0301 <<<<
	043334	104460					TRAP C\$ERROR
8597	043336	000415				BR 60\$	;EXIT THE TEST.
8598							
8599							
8600	043340	012737	000456	005466	8\$:	MOV #0302.,ERRNBR	;Report MR bit clear within 10 ms after DUT reset. ;SET THE ERROR NUMBER IN ERROR TABLE.
8601	043346	012701	011573			MOV #EM0302,R1	;SELECT ERROR MESSAGE.
8602	043352					ERROR	;REPORT ERROR. >>>> ERROR #0302 <<<<
	043352	104460					TRAP C\$ERROR
8603	043354	000406				BR 60\$	;EXIT THE TEST.
8604							
8605							
8606	043356	012737	000457	005466	10\$:	MOV #0303.,ERRNBR	;Report MR cleared between 1/5 second and 5 seconds of DUT reset. ;SET THE ERROR NUMBER IN ERROR TABLE.
8607	043364	012701	011733			MOV #EM0303,R1	;SELECT ERROR MESSAGE.
8608	043370					ERROR	;REPORT ERROR. >>>> ERROR #0303 <<<<
	043370	104460					TRAP C\$ERROR
8609							
8610	043372	005037	002362		60\$:	CLR CTRLCF	;INDICATE THAT WE COMPLETED THE TEST.
8611							
8612	043376					ENDTST	
	043376						
	043376	104401					L10036: TRAP C\$ETST

HARDWARE TEST

- RXCHRA -

SEQ 0223

```

8614 .SBTTL HARDWARE TEST - RXCHRA -
8615 ;* *****
8616 ;* - RBUF Register RX Character Field Test -
8617 ;* This test verifies that the RX Character Field of the DUT RBUF register
8618 ;* appears to be functioning correctly. This test uses the codes which
8619 ;* should be in the FIFO after a board reset and skip selftest sequence.
8620 ;*
8621 ;-- *****
8622 043400 BGNTST
      043400
8623      000004      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8624 043400 012737 000004 002364      MOV #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (4)
8625 043406 012737 177777 002362      MOV #-1,CTRLCF      ;INDICATE THAT WE ARE WITHIN A TEST.
8626 043414 012737 000001 005464      MOV #1,ERRTYP      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
8627 043422 012737 012112 005470      MOV #EM0401,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
8628 043430 012737 024562 005472      MOV #ER0201,ERRBLK      ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
8629 ;*
8630 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
8631 ; and wait up to 5 seconds for the MR bit to clear.
8632 ;--
8633 043436 012701 011610      MOV #5000.,R1      ;TIME-OUT VALUE IS 5.0 SECONDS.
8634 043442 012702 000040      MOV #BIT05,R2      ;WAITING FOR MASTER RESET BIT.
8635 043446 005003      CLR R3      ;WAITING FOR BIT TO CLEAR.
8636 043450 013704 002300      MOV CSRA,R4      ;BIT IS IN THE DUT'S CSR.
8637 043454 010214      MOV R2,(R4)      ;SET THE DUT MASTER RESET BIT.
8638 043456 004737 035740      JSR PC,SKPSTS      ;SKIP THE SELFTEST.
8639 043462 004737 031726      JSR PC,MSLGET      ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8640 043466 103015      BCC 4$      ;GO REPORT ERROR IF MR DID NOT CLEAR.
8641 ;*
8642 ; Read 6 characters from the DUT and verify that they are valid selftest
8643 ; codes.
8644 ;--
8645 043470 012400      MOV (R4)+,R0      ;INCREMENT POINTER TO POINT TO DUT RBUF REGSTR.
8646 043472 012701 000006      MOV #6,R1      ;INITIALIZE THE LOOP COUNTER.
8647 043476 011402      MOV (R4),R2      ;READ A CHARACTER FROM THE DUT RBUF REGISTER.
8648 043500 010200      MOV R2,R0
8649 043502 042700 177476      BIC #177476,R0      ;REMOVE ALL BUT BITS SPECIFIC TO SELFTEST CODE.
8650 043506 020027 000201      CMP R0,#201      ;CHECK THAT BITS 0,6, AND 7 ARE CORRECT.
8651 043512 001012      BNE 6$      ;GO REPORT ERROR IF CODE IS NOT SELFTEST CODE.
8652 043514 005301      DEC R1      ;COUNT THIS LOOP ITERATION.
8653 043516 001367      BNE 2$      ;LOOP IF NOT ALL LINES DONE.
8654 043520 000415      BR 60$      ;EXIT TEST, NO ERROR FOUND.
8655 ;*
8656 ; Error reports:
8657 ;--
8658 ;Report MR bit would not clear after a DUT reset.
8659 4$:      MOV #0401.,ERRNBR      ;SET THE ERROR NUMBER IN ERROR TABLE.
8660 043522 012737 000621 005466      MOV #EM0202,R1      ;SELECT ERROR MESSAGE.
8661 043530 012701 011033      ERROR      ;REPORT ERROR. >>>> ERROR #0401 <<<<
8662 043534 104460      BR 60$      ;EXIT THE TEST. TRAP C$ERROR
8663 043536 000406
8664 ;Report improper code found in DUT RBUF after reset (skip selftest).
8665 6$:      MOV #0402.,ERRNBR      ;SET THE ERROR NUMBER IN ERROR TABLE.
8666 043540 012737 000622 005466      MOV #EM0402,R1      ;SELECT ERROR MESSAGE.
8667 043546 012701 012161      ERROR      ;REPORT ERROR. >>>> ERROR #0402 <<<<
8668 043552

```

HARDWARE TEST - RXCHRA -

8669	043552	104460						TRAP	C#ERROR
8670	043554	005037	002362	60#:	CLR	CTRLCF		;INDICATE THAT WE COMPLETED THE TEST.	
8671									
8672	043560				ENDTST				
	043560								
	043560	104401					L10037:	TRAP	C#ETST

HARDWARE TEST

- RXFFDA -

```

8674 .SBTTL  HARDWARE TEST          - RXFFDA -
8675 ;+ *****
8676 ;*      - RBUF Register RX Flag Field Test -
8677 ;*      This test verifies that the Field of 3 flag bits in the RBUF reads
8678 ;*      as all ones when the Selftest codes are being read from the DUT
8679 ;*      after a board reset and skip selftest sequence.
8680 ;*
8681 ;-- *****
8682 043562 BGNTST
      043562
8683      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8684 043562 012737 000005 002364  MOV    #TNUM,TSTNUM          ;SET UP THE TEST NUMBER.      (5)
8685 043570 012737 177777 002362  MOV    #-1,CTRLCF          ;INDICATE THAT WE ARE WITHIN A TEST.
8686 043576 012737 000001 005464  MOV    #1,ERRTYP          ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
8687 043604 012737 012331 005470  MOV    #EM0501,ERRMSG        ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
8688 043612 012737 024562 005472  MOV    #ER0201,ERRBLK        ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
8689
8690 ;+
8691 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
8692 ; and wait up to 5 seconds for the MR bit to clear.
8693 043620 012701 011610
8694 043624 012702 000040
8695 043630 005003
8696 043632 013704 002300
8697 043636 010214
8698 043640 004737 035740
8699 043644 004737 031726
8700 043650 103013
8701
8702 ;+
8703 ; Read 8 characters from the DUT and verify that all 3 RX error flags are
8704 ; set for each characters.
8705 043652 012400
8706 043654 012701 000010
8707 043660 011402
8708 043662 012700 070000
8709 043666 040200
8710 043670 001012
8711 043672 005301
8712 043674 001371
8713 043676 000415
8714
8715 ;+
8716 ; Error reports:
8717 ;--
8718 ;Report MR bit would not clear after a DUT reset.
8719 043700 012737 000765 005466 4$:  MOV    #0501,ERRNBR        ;SET THE ERROR NUMBER IN ERROR TABLE.
8720 043706 012701 011033          MOV    #EM0202,R1          ;SELECT ERROR MESSAGE.
8721 043712          ERROR          ;REPORT ERROR.          >>>> ERROR #0501 <<<<
      043712 104460          TRAP    C$ERROR
8722 043714 000406          BR      60$          ;EXIT THE TEST.
8723
8724 ;Report one or more RX error flags found set with selftest code.
8725 043716 012737 000766 005466 6$:  MOV    #0502,ERRNBR        ;SET THE ERROR NUMBER IN ERROR TABLE.
8726 043724 012701 012377          MOV    #EM0502,R1          ;SELECT ERROR MESSAGE.
8727 043730          ERROR          ;REPORT ERROR.          >>>> ERROR #0502 <<<<
      043730 104460          TRAP    C$ERROR

```

HARDWARE TEST

RXFFDA -

8728

8729 043732 005037 002362

8730

8731 043736

043736

043736 104401

60\$:

CLR

CTRLCF

;INDICATE THAT WE COMPLETED THE TEST.

ENDTST

L10040:

TRAP

C\$ETST

HARDWARE TEST

- RDAA -

```

8733 .SBTTL HARDWARE TEST - RDAA -
8734 ;* *****
8735 ;* - CSR RX Data Available Bit Test -
8736 ;* This test verifies that the DUT CSR RX Data Available Bit is set by the
8737 ;* inclusion of the selftest codes in the DUT FIFO and that the bit clears
8738 ;* after the FIFO has been emptied.
8739 ;*
8740 ;-- *****
8741 043740 BGNTST
      043740
8742      000006      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8743 043740 012737 000006 002364      MOV #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (6)
8744 043746 012737 177777 002362      MOV #-1,CTRLCF      ;INDICATE THAT WE ARE WITHIN A TEST.
8745 043754 012737 000001 005464      MOV #1,ERRTYP      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
8746 043762 012737 012737 005470      MOV #EM0601,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
8747 043770 012737 024562 005472      MOV #ER0201,ERRBLK      ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
8748 ;*
8749 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
8750 ; and wait up to 5 seconds for the MR bit to clear.
8751 ;--
8752 043776 012701 011610      MOV #5000.,R1      ;TIME-OUT VALUE IS 5.0 SECONDS.
8753 044002 012702 000040      MOV #BIT05,R2      ;WAITING FOR MASTER RESET BIT.
8754 044006 005003      CLR R3      ;WAITING FOR BIT TO CLEAR.
8755 044010 013704 002300      MOV CSRA,R4      ;BIT IS IN THE DUT'S CSR.
8756 044014 010214      MOV R2,(R4)      ;SET THE DUT MASTER RESET BIT.
8757 044016 004737 035740      JSR PC,SKPSTS      ;SKIP THE SELFTEST.
8758 044022 004737 031726      JSR PC,MSLGET      ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8759 044026 103016      BCC 4$      ;GO REPORT ERROR IF MR DID NOT CLEAR.
8760 ;*
8761 ; Check that the RX Data Available bit is set.
8762 ;--
8763 044030 032714 000200      BIT #BIT7,(R4)      ;TEST THE DUT RX.DATA.AVAIL BIT.
8764 044034 001422      BEQ 6$      ;GO REPORT ERROR IF BIT IS NOT SET.
8765 ;*
8766 ; Read characters from the DUT RX FIFO and wait for RX.DATA.AVAIL to go clear.
8767 ;--
8768 044036 012705 001130      MOV #600.,R5      ;ALLOW READING 600 CHARS BEFORE ERROR.
8769 044042 010403      MOV R4,R3
8770 044044 012300      MOV (R3)+,R0      ;CALCULATE THE RBUF ADDRESS.
8771 044046 011300      MOV (R3),R0      ;READ A CHARACTER FROM THE RX FIFO.
8772 044050 032714 000200      BIT #BIT7,(R4)      ;TEST THE DUT RX.DATA.AVAIL BIT.
8773 044054 001427      BEQ 60$      ;EXIT TEST WITHOUT ERROR IF RX.DATA.AVAIL CLR.
8774 044056 005305      DEC R5      ;COUNT THE CHARACTER JUST READ.
8775 044060 001372      BNE 2$      ;LOOP IF NOT TOO MANY CHARS READ FROM FIFO.
8776 044062 000416      BR 8$      ;GO REPORT ERROR IF RX.DATA.AVAIL WOULDN'T CLR.
8777 ;*
8778 ; Error reports:
8779 ;--
8780 ;Report MR bit would not clear after a DUT reset.
8781 4$: MOV #0601.,ERRNBR      ;SET THE ERROR NUMBER IN ERROR TABLE.
8782 044064 012737 001131 005466 4$: MOV #EM0202,R1      ;SELECT ERROR MESSAGE.
8783 044072 012701 011033      ERROR      ;REPORT ERROR. >>>> ERROR #0601 <<<<
8784 044076 104460      TRAP C$ERROR
8785 044100 000415      BR 60$      ;EXIT THE TEST.
8786 ;Report that RX.DATA.AVAIL bit was not set after a reset completion.
8787

```


SEQ 0228

HARDWARE TEST

- RDAA -

```
8788 044102 012737 001132 005466 6$:   MOV     #0602.,ERRNBR    ;SET THE ERROR NUMBER IN ERROR TABLE.  
8789 044110 012701 012773              MOV     #EM0602,R1      ;SELECT ERROR MESSAGE.  
8790          044114 104460            ERROR                ;REPORT ERROR.             >>>> ERROR #0602 <<<<  
           044116 0C0406                      BR         60$               TRAP       C#ERROR  
8791                                     :EXIT THE TEST.  
8792  
8793                               ;Report that RX.DATA.AVAIL bit could not be cleared by purging FIFO.  
8794 044120 012737 001133 005466 8$:   MOV     #0603.,ERRNBR    ;SET THE ERROR NUMBER IN ERROR TABLE.  
8795 044126 012701 013153              MOV     #EM0603,R1      ;SELECT ERROR MESSAGE.  
8796          044132 104460            ERROR                ;REPORT ERROR.             >>>> ERROR #0603 <<<<  
           044132                  TRAP       C#ERROR  
8797  
8798 044134 005037 002362        60$:   CLR      CTRLCF          ;INDICATE THAT WE COMPLETED THE TEST.  
8799  
8800          044140                    ENDTST  
           044140                                  L10041:  
           044140 104401                                TRAP       C#ETST
```

HARDWARE TEST

- RDVA -

```

8802 .SBTTL HARDWARE TEST - RDVA -
8803 ;++ *****
8804 ;* - RBUF RX Data Valid Bit Test -
8805 ;* This test verifies that the DUT RBUF RX Data Valid Bit is set by the
8806 ;* inclusion of the selftest codes in the DUT FIFO and that the bit clears
8807 ;* after the FIFO has been emptied.
8808 ;*
8809 ;-- *****
8810 044142 BGNTST
      044142
8811      000007 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8812 044142 012737 000007 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (7)
8813 044150 012737 177777 002362 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
8814 044156 012737 000001 005464 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
8815 044164 012737 013336 005470 MOV #EM0701,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
8816 044172 012737 024562 005472 MOV #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
8817 ;+
8818 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
8819 ; and wait up to 5 seconds for the MR bit to clear.
8820 ;--
8821 044200 012701 011610 MOV #5000.,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
8822 044204 012702 000040 MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
8823 044210 005003 CLR R3 ;WAITING FOR BIT TO CLEAR.
8824 044212 013704 002300 MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
8825 044216 010214 MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
8826 044220 004737 035740 JSR PC,SKPSTS ;SKIP THE SELFTEST.
8827 044224 004737 031726 JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8828 044230 103012 BCC 4$ ;GO REPORT ERROR IF MR DID NOT CLEAR.
8829 ;+
8830 ; Check that the RX Data Valid bit is set.
8831 ;--
8832 044232 012400 MOV (R4)+,R0 ;INCREMENT POINTER TO PNT TO DUT RBUF REG.
8833 044234 005714 TST (R4) ;TEST THE DUT RX.DATA.VALID BIT.
8834 044236 100016 BPL 6$ ;GO REPORT ERROR IF BIT IS NOT SET.
8835 ;+
8836 ; Read characters from the DUT RX FIFO and wait for RX.DATA.VALID to go clear.
8837 ;--
8838 044240 012705 001130 MOV #600.,R5 ;ALLOW READING 600 CHARS BEFORE ERROR.
8839 044244 011400 2$: MOV (R4),R0 ;READ A CHARACTER FROM THE RX FIFO.
8840 044246 100027 BPL 60$ ;EXIT TEST WITHOUT ERROR IF BIT IS CLEAR.
8841 044250 005305 DEC R5 ;COUNT THE CHARACTER JUST READ.
8842 044252 001374 BNE 2$ ;LOOP IF NOT TOO MANY CHARS READ FROM FIFO.
8843 044254 000416 BR 8$ ;GO REPORT ERROR IF RX.DATA.VALID WOULDN'T CLR.
8844 ;+
8845 ; Error reports:
8846 ;--
8847 ;Report MR bit would not clear after a DUT reset.
8848 4$: MOV #0701.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
8849 044256 012737 001275 005466 MOV #EM0202,R1 ;SELECT ERROR MESSAGE.
8850 044264 012701 011033 ERROR ;REPORT ERROR. >>>> ERROR #0701 <<<<
8851 044270 104460 TRAP C$ERROR
      044270 104460
8852 044272 000415 BR 60$ ;EXIT THE TEST.
8853 ;+
8854 ;Report that RX.DATA.VALID bit was not set after a reset completion.
8855 044274 012737 001276 005466 6$: MOV #0702.,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
8856 044302 012701 013373 MOV #EM0702,R1 ;SELECT ERROR MESSAGE.

```


HARDWARE TEST

- RDVA -

```
8857 044306          ERROR          ;REPORT ERROR.          >>>> ERROR #0702 <<<<<
      044306 104460          TRAP          C#ERROR
8858 044310 000406          BR          60$          ;EXIT THE TEST.
8859
8860          ;Report that RX.DATA.VALID bit could not be cleared by purging FIFO.
8861 044312 012737 001277 005466 8$:      MOV          #0703,ERRNBR      ;SET THE ERROR NUMBER IN ERROR TABLE.
8862 044320 012701 013553          MOV          #EM0703,R1      ;SELECT ERROR MESSAGE.
8863 044324          ERROR          ;REPORT ERROR.          >>>> ERROR #0703 <<<<<
      044324 104460          TRAP          C#ERROR
8864
8865 044326 005037 002362          60$:      CLR          CTRLCF          ;INDICATE THAT WE COMPLETED THE TEST.
8866
8867 044332          ENDTST
      044332
      044332 104401          L10042:
                                TRAP          C#ETST
```

HARDWARE TEST

- RLNA -

```

8869 .SBTTL HARDWARE TEST - RLNA -
8870 ;* *****
8871 ;* - RBUF RX Line Number Field Test -
8872 ;* This test verifies that the DUT RBUF RX Line Number field is working
8873 ;* correctly by utilizing the selftest codes which are put in the RX
8874 ;* FIFO after a board reset.
8875 ;*
8876 ;*
8877 044334 BGNTST
      044334
8878      000010      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8879 044334 012737 000010 002364      MOV #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (8)
8880 044342 012737 177777 002362      MOV #-1,CTRLCF      ;INDICATE THAT WE ARE WITHIN A TEST.
8881 044350 012737 000001 005464      MOV #1,ERRTYP      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
8882 044356 012737 013736 005470      MOV #EM0801,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
8883 ;*
8884 ; Set the DUT CSR Master Reset (MR) bit, perform the skip selftest sequence,
8885 ; and wait up to 5 seconds for the MR bit to clear.
8886 ;
8887 044364 012701 011610      MOV #5000.,R1      ;TIME-OUT VALUE IS 5.0 SECONDS.
8888 044370 012702 000040      MOV #BIT05,R2      ;WAITING FOR MASTER RESET BIT.
8889 044374 005003      CLR R3      ;WAITING FOR BIT TO CLEAR.
8890 044376 013704 002300      MOV CSRA,R4      ;BIT IS IN THE DUT'S CSR.
8891 044402 010214      MOV R2,(R4)      ;SET THE DUT MASTER RESET BIT.
8892 044404 004737 035740      JSR PC,SKPSTS      ;SKIP THE SELFTEST.
8893 044410 004737 031726      JSR PC,MSLGET      ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8894 044414 103016      BCC 4$      ;GO REPORT ERROR IF MR DID NOT CLEAR.
8895 ;*
8896 ; Read characters from the DUT RX FIFO and verify that the line numbers are
8897 ; correct.
8898 ; One character is read from the FIFO for each possible line on the DUT.
8899 ;
8900 044416 005001      CLR R1      ;CLEAR THE LINE COUNTER.
8901 044420 0124J0      MOV (R4)+,R0      ;INCREMENT POINTER TO PNT TO THE DUT RBUF REG.
8902 044422 011402      MOV (R4),R2      ;READ A CHARACTER FROM THE DUT RX FIFO.
8903 044424 010203      MOV R2,R3
8904 044426 000303      SWAB R3
8905 044430 042703 177760      BIC #177760,R3      ;REMOVE ALL BUT LINE NUMBER BITS.
8906 044434 020301      CMP R3,R1      ;COMPARE WITH EXPECTED LINE NUMBER.
8907 044436 001017      BNE 6$      ;GO REPORT ERROR IF LINE NUMBERS DON'T MATCH.
8908 044440 005201      INC R1      ;INCREMENT THE EXPECTED LINE NUMBER.
8909 044442 020127 000010      CMP R1,#NUMLNS      ;COMPARE WITH NUMBER OF LINES ON DUT.
8910 044446 001365      BNE 2$      ;LOOP UNTIL CODES FOR ALL LINES ARE READ.
8911 044450 000423      BR 60$      ;EXIT TEST WITHOUT ERROR.
8912 ;*
8913 ; Error reports:
8914 ;
8915 ;
8916 ;Report MR bit would not clear after a DUT reset.
8917 044452 012737 001441 005466 4$: MOV #0801.,ERRNBR      ;SET THE ERROR NUMBER IN ERROR TABLE.
8918 044460 012737 024644 005472      MOV #ER0503,ERRBLK      ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
8919 044466 012701 011033      MOV #EM0202,R1      ;SELECT ERROR MESSAGE.
8920 044472      ERROR      ;REPORT ERROR. >>>> ERROR #0801 <<<<
      044472 104460      TRAP C#ERROR
8921 044474 000411      BR 60$      ;EXIT THE TEST.
8922
8923 ;Report that RX Line Number field is wrong for selftest code.

```

HARDWARE TEST

- RLNA -

```
8924 044476 012737 001442 005466 64:  MOV  #0802,ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
8925 044504 012737 024562 005472  MOV  #ER0201,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
8926 044512 012701 01377C  MOV  #EM0802,R1 ;SELECT ERROR MESSAGE.
8927 044516  ERROR  ;REPORT ERROR. >>>> ERROR #0802 <<<<
      044516 104460  TRAP  C#ERROR
8928
8929 044520 005037 002362 604: CLR  CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
8930
8931 044524  ENDTST
      044524
      044524 104401  L10043:
                        TRAP  C#ETST
```

HARDWARE TEST

- BMPCHK -

SEQ 0233

```

8933 .SBTTL HARDWARE TEST - BMPCHK -
8934 ;* *****
8935 ;* - BMP check Test -
8936 ;* This test is used to verify that the DUT does not immediately fail
8937 ;* the on-board background-monitor program, and hence invalidate
8938 ;* succeeding tests.
8939 ;* This test looks for BMP codes in the fifo for a set period immediately
8940 ;* after the self-test is skipped.
8941 ;* Any BMP codes that are found are saved on the queue and are also
8942 ;* reported in this test.
8943 ;*
8944 ;*
8945 ;* -- *****
      BGNTST
8946      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
8947      MOVL #TNUM,TSTNUM        ;SET UP THE TEST NUMBER. (9)
8948      MOV #1,CTRLCF            ;INDICATE THAT WE ARE WITHIN A TEST.
8949      MOV #1,ERRTYP            ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
8950      MOV #0901,ERRNBR        ;SET THE ERROR NUMBER.
8951 ;*
8952 ; Wait up to 3 seconds for the DUT Master Reset bit to clear.
8953 ; If time-out occurs, then exit this test.
8954 ;*
8955      MOV #3000.,R1            ;TIME-OUT VALUE IS 3.0 SECONDS.
8956      MOV #BIT05,R2            ;WAITING FOR MASTER RESET BIT.
8957      CLR R3                    ;WAITING FOR BIT TO CLEAR.
8958      MOV CSRA,R4              ;BIT IS IN THE DUT'S CSR.
8959      JSR PC,MSLGET            ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
8960      BCC 50$                 ;ABORT THE TEST IF MR DID NOT CLEAR.
8961 ;*
8962 ; Reset the DUT, skip the self-test.
8963 ;*
8964      MOV R2,(R4)              ;SET THE DUT MASTER RESET BIT.
8965      JSR PC,SKPSTS            ;WRITE THE SKIP SELFTEST CODES TO THE DUT.
8966 ;*
8967 ; Wait for Master reset to clear. Delay for 500 milli-sec before purging
8968 ; the fifo.
8969 ;*
8970      MOV #500.,R4            ;TIME-OUT VALUE IS 500 MILLI-SECONDS.
8971      JSR PC,DELAY            ;WAIT FOR BMP TO BEGIN EXECUTION.
8972      JSR PC,PUFIFO           ;PURGE THE FIFO, SAVING ANY BMP CODES.
8973      BCC 50$                 ;ABORT THE TEST IF THE FIFO DID NOT CLEAR.
8974 ;*
8975 ; Report the error if any BMP codes were found.
8976 ;*
8977      MOV BMPCP,R2            ;GET THE CONTENTS OF THE POINTER TO THE BMP Q.
8978      MOV #BMPQ08,R3          ;GET THE START ADDRESS OF THE QUEUE.
8979      CMP R2,R3                ;SEE IF THE POINTER HAS MOVED FROM THE BASE.
8980      BEQ 60$                 ;EXIT NO CODES IN THE QUEUE.
8981 ;*
8982 ; There is at least one BMP code in the queue. Report the error.
8983 ;*
8984 ;Report error BMP CODE FOUND IN TEST nn, BMP CODE:nnnnnn"
8985      MOV #EM0902,R1          ;PASS THE MESSAGE TO BE REPORTED.
8986      ERROF 0901,EM0901,ER9301 ;
                                     >>>> ERROR 0901 <<<<.
                                     TRAP C1ERDF
                                     .WORD 901

```

SEQ 0234

HARDWARE TEST

- ВМРСНЖ -

ADDRESS	HEX	ASSEMBLY	OPERANDS	OPERATION	REMARKS
044652	014051				
044654	026600				
8987	044656	000405	BR	60:	
8988					
8989	044660	012737 001606 005466 50:	MOV	#902.,ERRNBR	;SET >>>> ERROR #0902 <<<<.
8990	044666	004737 036106	JSR	PC,TSABRT	;REPORT NON-TEST RELATED ERROR.
8991					
8992	044672	005037 002362 60:	CLR	CTRLCF	;INDICATE THAT WE COMPLETED THE TEST.
8993					
8994	044676		ENDTST		
	044676				
	044676	104401			
					L10044:
					TRAP C\$ETST

HARDWARE TEST

- BMPCHK -

8996

8997

8998

8999

9000

9001

9002

9003

9004

9005

9006 044700

044700

9007

000012

9008 044700

012737

000012

002364

9009 044706

012737

177777

002362

9010 044714

012737

000001

005464

9011 044722

012737

014136

005470

9012 044730

012737

024644

005472

9013

9014

9015

9016

9017 044736

012701

005670

9018 044742

012702

000040

9019 044746

005003

9020 044750

013704

002300

9021 044754

004737

031726

9022 044760

103037

9023

9024

9025

9026

9027

9028 044762

012701

000062

9029 044766

010214

9030 044770

004737

035740

9031 044774

004737

031726

9032 045000

103011

9033 045002

020127

000050

9034 045006

003015

9035

9036

9037

9038

9039

9040

9041 045010

012737

001753

005466

9042 045016

004737

034734

9043 045022

000423

9044

9045

9046

9047

9048 045024

012737

001751

005466

9049 045032

012701

014202

9050 045036

045036 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.

;+ *****

;+ *****

BGNTST

T10::

TNUM == TNUM + 1

;INCREMENT THE ASSEMBLY TIME TEST COUNTER.

MOV @TNUM,TSTNUM

;SET UP THE TEST NUMBER. (10)

MOV @-1,CTRLCF

;INDICATE THAT WE ARE WITHIN A TEST.

MOV @1,ERRTYP

;SET ERROR TYPE AS FATAL IN ERROR TABLE.

MOV @EM1001,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.,R..

;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.

BGT 4\$

;GO REP ERR IF SELFTEST COMPLETED IN < 10 MS.

;+

; 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 subroutine 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.

2\$:

MOV @1001.,ERRNBR

;SET THE ERROR NUMBER IN THE ERROR TABLE.

MOV @EM1002,R1

;SELECT ERROR MESSAGE.

ERROR

;REPORT ERROR.

>>>> ERROR #1001 <<<<

TRAP C\$ERROR

HARDWARE TEST

- SKSELF -

```

9051 045040 000414          BR      60$          ;EXIT THE TEST.
9052
9053
9054 045042 012737 001752 005466 4$: ;Report Skip Self-test completed too soon.
9055 045050 012701 014267          MOV      #1002.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
9056 045054          MOV      #EM1003,R1 ;SELECT ERROR MESSAGE.
          ERROR          ;REPORT ERROR.          >>>>> ERROR #1002 <<<<<
          TRAP      C$ERROR
9057 045056 000405          BR      60$          ;EXIT THE TEST.
9058
9059 045060 012737 001753 005466 50$: MOV      #1003.,ERRNBR ;SET ERROR NUMBER.
9060 045066 004737 036106          JSR      PC,TSABRT ;REPORT NON-TEST RELATED ERROR.
9061
9062 045072 005037 002362          60$: CLR      CTRLCF          ;INDICATE THAT WE COMPLETED THE TEST.
9063
9064 045076          ENDTST
          045076
          045076 104401
          L10045:
          TRAP      C$ETST

```

HARDWARE TEST

- SKSELF -

```

9066
9067
9068
9069
9070
9071
9072
9073
9074
9075 045100
      045100
9076      000013
9077 045100 012737 000013 002364
9078 045106 012737 177777 002362
9079 045114 012737 000001 005464
9080 045122 012737 014345 005470
9081 045130 012737 024644 005472
9082
9083
9084
9085
9086 045136 012701 005670
9087 045142 012702 000040
9088 045146 005003
9089 045150 013704 002300
9090 045154 004737 031726
9091 045160 103044
9092
9093
9094
9095 045162 010214
9096 045164 004737 035740
9097
9098
9099
9100
9101 045170 012701 000005
9102 045174 012702 020000
9103 045200 010203
9104 045202 013704 002300
9105 045206 004737 031726
9106 045212 103020
9107
9108
9109
9110
9111
9112
9113
9114 045214 012701 000017
9115 045220 005003
9116 045222 004737 031726
9117 045226 103012
9118 045230 010105
9119 045232 012701 000001
9120 045236 052703 020000
9121 045242 004737 031726

.SBTTL HARDWARE TEST - DFSKST -
;-- *****
;* - Diagnostic fail bit, skip self-test test -
;* This test verifies that the diagnostic fail bit of the DUT, correctly
;* changes state as the on-board selftest is skipped.
;*
;-- *****
      BGNTST
;--
      T11::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV @TNUM,TSTNUM ;SET UP THE TEST NUMBER. (11)
      MOV @-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
      MOV @1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV @EM1101,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.
;+
; Reset the DUT, skip the self-test.
;-
      MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
      JSR PC,SKPSTS ;WRITE THE SKIP SELFTEST CODES TO THE DUT.
;+
; Set time out of 5 milli seconds, wait for Diag_fail bit to set.
; If time-out occurs go report the error.
;-
      MOV @5,R1 ;TIME-OUT VALUE IS 5 MILLI SECONDS.
      MOV @BIT13,R2 ;WAITING FOR DIAGNOSTIC FAIL BIT.
      MOV R2,R3 ;WAITING FOR BIT TO SET.
      MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
      JSR PC,MSLGET ;WAIT FOR DUT_CSR_DF BIT TO CLEAR.
      BCC 4$ ;IF DIAG_FAIL DID NOT SET, GO REPORT ERROR.
;+
; Set time-out of 15 milli secs, wait for Diag_Fail to clear.
; If time-out occurs go report the error.
; Verify the Diag_fail bit is in a stable state before continuing. Loop
; back if the state was transitory, using the remainder of the 15 ms time-out.
;-
      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.
2$:

```


HARDWARE TEST

- DFSKST -

```

9122 045246 103016          BCC      60$      ;EXIT IF DIAG_FAIL BIT STILL CLEAR.
9123 045250 010501          MOV      R5,R1    ;PASS THE REMAINING TIME-OUT VALUE.
9124 045252 000762          BR       2$      ;LOUP TO CHECK AGAIN.
9125                          ;+
9126                          ; Error reports:
9127                          ; -
9128                          ;Report Diagnostic fail bit bad.
9129 045254 012737 002115 005466 4$:      MOV      #1101.,ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
9130 045262 012701 014606          MOV      #EM1205,R1 ;SELECT ERROR MESSAGE.
9131 045266          ERROR          ;REPORT ERROR.          >>>>> ERROR #1101 <<<<<
9132 045270 000405          BR       60$      TRAP      C$ERROR
9133                          ;EXIT THE TEST.
9134 045272 012737 002116 005466 50$:      MOV      #1102.,ERRNBR ;SET THE ERROR NUMBER FOR TSABRT RTN.
9135 045300 004737 036106          JSR      PC,TSABRT ;REPORT NON-TEST RELATED ERROR.
9136
9137 045304 005037 002362          60$:      CLR      CTRLCF ;INDICATE THAT WE COMPLETED A TEST.
9138
9139 045310          ENDTST
          045310
          045310 104401          L10046:
                                TRAP      C$ETST

```

HARDWARE TEST

- SELFTS -

```

9141 .SBTTL  HARDWARE TEST          - SELFTS -
9142 ;* *****
9143 ;* - Self-test Test -
9144 ;* This test verifies that the DUT's Self-Test executes within the
9145 ;* time allowed, and that the fifo contains the correct codes after its
9146 ;* completion.
9147 ;*
9148 ;* *****
9149 ;* BGNTST
9150 ;*
9151 ;* T12::
9152 ;* TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9153 ;* MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (12)
9154 ;* MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
9155 ;* MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
9156 ;* MOV #EM1201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
9157 ;* MOV #ER0503,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
9158 ;*
9159 ;* ; Wait up to 3 seconds for the DUT Master Reset bit to clear.
9160 ;* ; If time-out occurs, then exit this test.
9161 ;*
9162 ;* MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
9163 ;* MOV #BIT05,R2 ;WAITING FOR MASTER RESET BIT.
9164 ;* CLR R3 ;WAITING FOR BIT TO CLEAR.
9165 ;* MOV CSRA,R4 ;BIT IS IN THE DUT'S CSR.
9166 ;* JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
9167 ;* BCC 50$ ;ABORT THE TEST IF MR DID NOT CLEAR.
9168 ;*
9169 ;* ; Determine if the Self-test takes too short or too long a time to complete.
9170 ;* ; Set-up a time-out of 3 second, if MR is clear in less than 1/2 second, or
9171 ;* ; greater than 3 seconds, report the error.
9172 ;*
9173 ;* MOV #3000.,R1 ;TIME-OUT VALUE IS 3.0 SECONDS.
9174 ;* MOV R2,(R4) ;SET THE DUT MASTER RESET BIT.
9175 ;* JSR PC,MSLGET ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
9176 ;* BCC 4$ ;GO REPORT ERROR SELFTEST TOOK TOO LONG.
9177 ;* MOV #3000.,R2
9178 ;* SUB R1,R2 ;CALCULATE # OF MS SELFTEST TO COMPLETE.
9179 ;* CMP R2,#50.
9180 ;* BLT 6$ ;SELFTEST SKIPPED? YES, GO REPORT ERROR.
9181 ;* CMP R2,#500.
9182 ;* BLT 8$ ;GO REP ERR IF SELFTEST COMPLETED IN < 1/2 SEC.
9183 ;*
9184 ;* ; Self-test completed within 1sec to 3 seconds.
9185 ;* ; Check the state of the Diagnostic Fail bit, report error if it is set.
9186 ;*
9187 ;* BIT #BIT13,(R4) ;DETERMINE IF THE DIAG_FAIL BIT IS CLEAR.
9188 ;* BEQ 2$ ;SKIP ERROR REPORT IF BIT IS CLEAR.
9189 ;* ;Report Diagnostic fail bit bad.
9190 ;* MOV #1204.,ERRNBR ;SET ERROR NUMBER TO IN ERROR TABLE.
9191 ;* MOV #EM1205,R1 ;SELECT THE ERROR MESSAGE.
9192 ;* ERROR ;
9193 ;* >>>> ERROR #1204 <<<<
9194 ;* TRAP C#ERROR
9195 ;*
9196 ;* ; Verify that the self-test codes in the fifo are "good" codes ,ie the DUT
9197 ;* ; successfully completed the self-test.
9198 ;* ; This subrout ne reports errors with numbers >>>> 1205 thru 1209 <<<<.
9199 ;*

```

HARDWARE TEST

- SELFTS -

```

9196 045454 012737 002265 005466 2$:  MOV    #1205.,ERRNBR  ;SET ERROR NUMBER TO 1205.
9197 045462 004737 034734          JSR    PC,RSTRPT  ;CHECK SELF-TEST CODES IN THE FIFO.
9198 045466 000431          BR     60$      ;EXIT TEST.
9199
9200          ;+
9201          ; Error reports:
9202          ;-
9203 045470 012737 002261 005466 4$:  ;Report Self-test took too long to complete.
9204 045476 012701 014410          MOV    #1201.,ERRNBR  ;SET THE ERROR NUMBER IN THE ERROR TABLE.
9205 045502          MOV    #EM1202,R1  ;SELECT ERROR MESSAGE.
9206 045504 104460          ERROR      ;REPORT ERROR.          >>>>> ERROR #1201 <<<<<
9207          BR     60$      ;EXIT THE TEST.          TRAP    C$ERROR
9208
9209 045506 012737 002262 005466 6$:  ;Report Self-test did not execute after DUT reset.
9210 045514 012701 014552          MOV    #1202.,ERRNBR  ;SET THE ERROR NUMBER IN ERROR TABLE.
9211 045520          MOV    #EM1204,R1  ;SELECT ERROR MESSAGE.
9212 045520 104460          ERROR      ;REPORT ERROR.          >>>>> ERROR #1202 <<<<<
9213          TRAP    C$ERROR
9214 045522 012737 002263 005466 8$:  ;Report Self-test competed too soon.
9215 045530 012701 014474          MOV    #1203.,ERRNBR  ;SET THE ERROR NUMBER IN THE ERROR TABLE.
9216 045534          MOV    #EM1203,R1  ;SELECT ERROR MESSAGE.
9217 045536 104460          ERROR      ;REPORT ERROR.          >>>>> ERROR #1203 <<<<<
9218 045536 000405          BR     60$      ;EXIT THE TEST.          TRAP    C$ERROR
9219 045540 012737 002272 005466 50$:  MOV    #1210.,ERRNBR  ;SET THE ERROR NUMBER FOR TSABRT RTN.
9220 045546 004737 036106          JSR    PC,TSABRT  ;REPORT NON-TEST RELATED ERROR.
9221
9222 045552 005037 002362          60$:  CLR     CTRLCF      ;INDICATE THAT WE COMPLETED THE TEST.
9223
9224 045556          ENDTST
          045556
          045556 104401

```

L10047:
TRAP C\$ETST

HARDWARE TEST

- SELFTS -

```

9226
9227
9228
9229
9230
9231
9232
9233
9234
9235
9236 045560
      045560
9237      000015
9238 045560 012737 000015 002364
9239 045566 012737 177777 002362
9240 045574 012737 000001 005464
9241 045602 012737 014632 005470
9242 045610 012737 024644 005472
9243 045616 012737 002425 005466
9244
9245
9246
9247
9248 045624 012701 005670
9249 045630 012702 000040
9250 045634 005003
9251 045636 013704 002300
9252 045642 004737 031726
9253 045646 103120
9254
9255
9256
9257
9258 045650 010214
9259 045652 004737 035740
9260 045656 012701 000764
9261 045662 004737 031726
9262 045666 103110
9263 045670 005237 005466
9264 045674 012700 000010
9265 045700 005003
9266 045702 013704 002302
9267 045706 011402
9268 045710 100077
9269 045712 042702 007402
9270 045716 020227 170001
9271 045722 001002
9272 045724 012703 177777
9273 045730 005300
9274 045732 001365
9275 045734 005703
9276 045736 100466
9277
9278
9279
9280
9281

```

```

.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 "Proc1 to rom error" to be placed in the fifo.
;+ *****
;-- BGNTST
      T13::
      TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (13)
      MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
      MOV #1,ERRRTP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV #EM1301,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
      MOV #ERO503,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
      MOV #1301,ERRNBR ;SET ERROR NUMBER TO 1301.
;+
; 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 TBUFFT register on channel 0.
;-

```

HARDWARE TEST

- STFAIL -

```

9282 045740 012777 000040 134332      MOV    #BIT05,0CSRA    ;SET DUT MASTER RESET BIT, SELECT CHANNEL 0.
9283 045746 012704 000031              MOV    #25.,R4        ;PASS DELAY PERIOD OF 25 MILLI SECS.
9284 045752 004737 030370              JSR    PC,DELAY        ;WAIT FOR SELFTEST TO INITIALISE.
9285 045756 012777 146314 134332      MOV    #146314,0TXBFCA ;WRITE THE FAIL SELF-TEST CODE TO TBUFFCT REG.
9286
9287      ;+
9288      ; Wait up to 2 seconds for the self-test to complete.
9289      ; If time-out occurs, then exit this test.
9290
9290 045764 005237 005466              INC    ERRNBR        ;SET ERROR NUMBER TO 1303.
9291 045770 012701 003720              MOV    #2000.,R1      ;TIME-OUT VALUE IS 2.0 SECONDS.
9292 045774 012702 000040              MOV    #BIT05,R2      ;PASS THE BIT MAP OF THE BIT TO TEST.
9293 046000 005003                      CLR    R3            ;SET UP THE EXPECTED STATE.
9294 046002 013704 002300              MOV    CSRA,R4        ;BIT IS IN THE DUT'S CSR.
9295 046006 004737 031726              JSR    PC,MSLGET       ;WAIT FOR DUT_CSR_MR BIT TO CLEAR.
9296 046012 103036                      BCC    50$           ;GO REPORT ERROR IF MR DID NOT CLEAR.
9297
9298      ;+
9299      ; Verify the diagnostic fail bit is set, indicating the error.
9300      ; Report error if diagnostic fail bit is clear.
9301
9301 046014 005237 005466              INC    ERRNBR        ;SET ERROR NUMBER TO 1304.
9302 046020 032714 020000              BIT    #BIT13,(R4)    ;CHECK THE STATE OF THE DIAG_FAIL BIT.
9303 046024 001425                      BEQ    10$           ;GO REPORT ERROR IF DIAG_FAIL BIT CLEAR.
9304
9305      ;+
9306      ; Remove the 8 self-test codes form the fifo, and verify that at least
9307      ; one is a Procl to ram error code (231).
9308
9308 046026 005237 005466              INC    ERRNBR        ;SET ERROR NUMBER TO 1305.
9309 046032 012700 000010              MOV    #8.,R0        ;SET MAXIMUM READ COUNT.
9310 046036 005001                      CLR    R1            ;CLEAR THE CORRECT CODE COUNTER.
9311 046040 013704 002302              MOV    RBUFA,R4      ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
9312 046044 011402                      MOV    (R4),R2        ;READ A CODE FROM THE FIFO.
9313 046046 100020                      BPL    50$           ;GO REPORT ERROR IF THE FIFO IS EMPTY.
9314 046050 042702 007400              BIC    #7400,R2      ;REMOVE THE LINE NUMBER FROM THE CODE.
9315 046054 120227 170231              CMPB   R2,#170231   ;IS IT THE CORRECT ERROR CODE?.
9316 046060 001001                      BNE    8$            ;SKIP NEXT INSTRUCTION, IF NOT A 231 CODE.
9317 046062 005201                      INC    R1            ;INCREMENT COUNTER.
9318 046064 005300                      DEC    R0            ;DECREMENT MAX READ COUNTER.
9319 046066 001366                      BNE    6$            ;LOOP IF 8 CODES HAVE NOT BEEN READ.
9320 046070 005701                      TST    R1            ;WERE ANY 231 CODES FOUND?.
9321 046072 001010                      BNE    60$           ;YES, THEN EXIT.
9322 046074 005237 005466              INC    ERRNBR        ;NO, SET ERROR NUMBER TO 1306 AND REPORT ERROR.
9323      ;Report Self-test error reporting bad.
9324 046100 012701 014656      10$:    MOV    #EM1302,R1    ;SELECT ERROR MESSAGE.
9325 046104 104460              ERROR          ;REPORT ERROR.
9326 046106 000402              BR     60$           ;EXIT THE TEST.
9327
9328 046110 004737 036106      50$:    JSR    PC,TSABRT      ;REPORT NON-RELATED TEST ERROR.
9329
9330 046114 005037 002362      60$:    CLR    CTRLCF        ;INDICATE THAT WE COMPLETED THE TEST.
9331
9332 046120                      ENDTST
          L10050:
          046120                      TRAP    C#ETST
          046120 104401

```

HARDWARE TEST

- STFAIL -

```

9334
9335
9336 .SBTTL  HARDWARE TEST          - REGWRW -
9337 ;++ *****
9338 ;*          - Device Register Word Access Read and Write Test -
9339 ;*
9340 ;* This test verifies that the device registers can be read and written
9341 ;* correctly using word accesses.
9342 ;*
9343 ;-- *****
9344
9345 046122      BGNTST
9346 046122
9347 046122 000016      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9348 046130 012737 000016 002364      MOV    #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (16)
9349 046136 012737 177777 002362      MOV    #-1,CTRLCF      ;INDICATE THAT WE ARE WITHIN A TEST.
9350 046144 012737 000001 005464      MOV    #1,ERRTYP      ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
9351 046152 012737 003101 005466      MOV    #1601,ERRNBR      ;SET UP ERROR NUMBER IN THE ERROR TABLE.
9352 046160 005037 002650      MOV    #EM1604,ERRMSG      ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
9353 046164 012700 003452      CLR    ERSRMR      ;CLEAR THE ERROR SUMMARY FLAGS.
9354 046170 004737 030214      MOV    #ERCNTB,R0
9355      JSR    PC,CLR16W      ;CLEAR THE ERROR COUNTER TABLE.
9356
9357 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
9358 ; Clear TX and RX interrupt enable bits in the CSR.
9359 ; This subroutine reports errors >>>> 1601 <<<<.
9360 046174 004737 034530      JSR    PC,RESETT      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9361 046200 103402      BCS    .+6      ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
9362 046202 000137 046324      JMP    60$      ;YES, EXIT THE TEST.
9363
9364 ; Verify read/write capability to indirect address field of CSR
9365 ;--
9366 046206 005237 005466      INC    ERRNBR      ;SET THE ERROR REPORT NUMBER TO 1602.
9367 046212 012702 000017      MOV    #17,R2      ;SET LOOP COUNT.
9368 046216 013704 002300      MOV    CSRA,R4      ;GET CSR ADDRESS.
9369 046222 010214      2$: MOV    R2,(R4)      ;WRITE COUNT TO CSR.
9370 046224 011401      MOV    (R4),R1      ;READ BACK THE CONTENTS OF THE CSR
9371 046226 042701 177760      BIC    #177760,R1      ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
9372 046232 020102      CMP    R1,R2      ;CHECK FOR CORRECT DATA WRITTEN/READ.
9373 046234 001406      BEQ    4$      ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
9374      ;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
9375 046236 012737 024736 005472      MOV    #ER1601,ERRBLK      ;SELECT THE PROPER ERROR REPORT ROUTINE.
9376 046244 005003      CLR    R3      ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
9377 046246 005005      CLR    R5      ;CAUSE REPORT OF LINE 0.
9378 046250      ERROR      ; >>>> ERROR # 1602 <<<<
9379 046252 005302      4$: DEC    R2      TRAP    C$ERROR      ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
9380 046254 002362      BGE    2$      ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.
9381
9382 ; Write and verify 16 data patterns in all used bits of all registers on all
9383 ; active lines. Before writing each pattern, clear all the bits.
9384 ; REGTST routine reports errors with numbers >>>> ERROR 1603 - 1605 <<<<.
9385 ;--
9386 046256 005237 005466      INC    ERRNBR      ;SET THE ERROR NUMBER TO 1603.
9387 046262 005003      CLR    R3      ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
9388 046264 012704 000002      MOV    #2,R4      ;INDICATE R/W ACCESS, CLEAR FIRST.

```


- REGWRW -

```
JSR      PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
```

```

; *
; Write and verify 16 data patterns in all used bits of all registers on all
; active lines. Before writing each pattern, set all the bits.
; REGTST routine reports errors with numbers >>>> ERROR 1606 - 1608 <<<<.

```

; Write and verify 16 data patterns in all used bits of all registers on all
; active lines. Before writing each pattern, set all the bits.

```

; REGTST routine reports errors with numbers >>>> ERROR 1606 - 1608 <<<<

```

;

```
MOV    #1606.,ERRNBR    ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
```

```

CLR      R3          ;SET OF ERROR NUMBER FOR REGIST ROUTINE.
                     ;INDICATE THAT WORD ACCESSSES ARE TO BE USED.

```

NEG R4 _____, INDICATE R/W ACCESS, SET FIRST.

```
JSR      PC,REGTST      ;ENHANCE NOW ACCESS, SET FIRST.
                        ;WRITE AND VERIFY DATA PATTERNS.
```

◆ ◆

```
; Print error summary reports if necessary.
```

```

; The following routine reports errors with number >>>> ERROR # 1609 <<<<

```

;

```
MOV    @1609.,ERRNBR    ;SET UP ERROR NUMBER FOR NEXT RTN.
```

JSR PC,REPSMR ;REPORT ERROR SUMMARY IF NECESSARY.

60\$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.

ENDTST

L10051:

TRAP C8ETST

HARDWARE TEST

- REGWRM -

SEQ 0245

```

9408 .SBTTL HARDWARE TEST - REGWRM -
9409 ;* *****
9410 ;* - Device Register Word Access Read/Modify/Write Test -
9411 ;*
9412 ;* This test verifies that the device registers can be written correctly
9413 ;* using word read/modify/write accesses.
9414 ;*
9415 ;*
9416 ;*
9417 046332 BGNSTST
          046332
9418          000017          TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9419 046332 012737 000017 002364          MOV #TNUM,TSTNUM          ;SET UP THE TEST NUMBER. (17)
9420 046340 012737 177777 002362          MOV #-1,CTRLCF          ;INDICATE THAT WE ARE WITHIN A TEST.
9421 046346 012737 000001 005464          MOV #1,ERRTYP          ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
9422 046354 012737 003245 005466          MOV #1701,ERRNBR          ;SET UP ERROR NUMBER IN THE ERROR TABLE.
9423 046362 012737 015357 005470          MOV #EM1701,ERRMSG          ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
9424 046370 005037 002650          CLR ERSMRF          ;CLEAR THE ERROR SUMMARY FLAGS.
9425 046374 012700 003452          MOV #ERCNTB,R0
9426 046400 004737 030214          JSR PC,CLR16W          ;CLEAR THE ERROR COUNTER TABLE.
9427
9428 ;*
9429 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
9430 ; Clear TX and RX interrupt enable bits in the CSR.
9431 ; This subroutine reports errors >>>> 1701 <<<<.
9432 046404 004737 034530          JSR PC,RESETT          ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9433 046410 103402          BCS .+6          ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
9434 046412 000137 046540          JMP 60$          ;YES, EXIT THE TEST.
9435
9436 ;*
9437 ; Verify read/modify/write capability to indirect address field of CSR
9438 046416 005237 005466          INC ERRNBR          ;SET THE ERROR REPORT NUMBER TO 1702.
9439 046422 012702 000017          MOV #17,R2          ;SET LOOP COUNT.
9440 046426 013704 002300          MOV CSRA,R4          ;GET CSR ADDRESS.
9441 046432 042714 000017          2$: BIC #17,(R4)          ;CLEAR THE DUT CSR USING READ/MODIFY/WRITE.
9442 046436 050214          BIS R2,(R4)          ;WRITE COUNT TO CSR USING READ/MODIFY/WRITE.
9443 046440 011401          MOV (R4),R1          ;READ BACK THE CONTENTS OF THE CSR
9444 046442 042701 177760          BIC #177760,R1          ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
9445 046446 020102          CMP R1,R2          ;CHECK FOR CORRECT DATA WRITTEN/READ.
9446 046450 001406          BEQ 4$          ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
9447          ;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
9448 046452 012737 024736 005472          MOV #ER1601,ERRBLK          ;SELECT THE PROPER ERROR REPORT ROUTINE.
9449 046460 005003          CLR R3          ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
9450 046462 005005          CLR R5          ;CAUSE REPORT OF LINE 0.
9451 046464          ERROR          ; >>>> ERROR # 1702 <<<<
9452 046466 005302          4$: DEC R2          ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
9453 046470 002360          BGE 2$          ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.
9454
9455 ;*
9456 ; Write and verify 16 data patterns in all used bits of all registers on all
9457 ; active lines using R/M/W. Before writing each pattern, clear all the bits.
9458 ; REGTST routine reports errors with numbers >>>> ERROR 1703 - 1705 <<<<.
9459 046472 005237 005466          INC ERRNBR          ;SET THE ERROR NUMBER TO 1703.
9460 046476 005003          CLR R3          ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
9461 046500 012704 000001          MOV #1,R4          ;INDICATE R/M/W ACCESS, CLEAR FIRST.
9462 046504 004737 034140          JSR PC,REGTST          ;WRITE AND VERIFY DATA PATTERNS.

```


HARDWARE TEST

- REGWRM -

```

9463
9464
9465
9466
9467
9468 046510 012737 003252 005466
9469 046516 005003
9470 046520 005404
9471 046522 004737 034140
9472
9473
9474
9475
9476 046526 012737 003255 005466
9477 046534 004737 034502
9478 046540 005037 002362
9479 046544
    046544
    046544 104401

```

```

;+
; 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, set all the bits.
; REGTST routine reports errors with numbers >>>> ERROR 1706 - 1708 <<<<.
;-
    MOV    #1706.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
    CLR    R3             ;INDICATE THAT WORD ACCESSES ARE TO BE USED.
    NEG    R4             ;INDICATE R/M/W ACCESS, SET FIRST.
    JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.

;+
; Print error summary reports if necessary.
; The following routine reports errors with number >>>> ERROR # 1709 <<<<
;-
    MOV    #1709.,ERRNBR ;SET UP ERROR NUMBER FOR NEXT RTN.
    JSR    PC,REPSMR      ;REPORT ERROR SUMMARY IF NECESSARY.
604:    CLR    CTRLCF      ;INDICATE THAT WE COMPLETED THE TEST.
    ENDTST

                                L10052:
                                TRAP    C#ETST

```

HARDWARE TEST

- REGBRW -

```

9481 .SBTTL  HARDWARE TEST          - REGBRW -
9482 ;* *****
9483 ;*          - Device Register Byte Access Read and Write Test -
9484 ;*
9485 ;* This test verifies that the device registers can be read and written
9486 ;* correctly using byte accesses.
9487 ;*
9488 ;* *****
9489 ;*
9490 046546          BGNTST
9491 046546          T16::
9491          TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9492 046546 012737 000020 002364  MOV  #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (18)
9493 046554 012737 177777 002362  MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE WITHIN A TEST.
9494 046562 012737 000001 005464  MOV  #1,ERRTYP      ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
9495 046570 012737 003411 005466  MOV  #1801,ERRNBR     ;SET UP ERROR NUMBER IN THE ERROR TABLE.
9496 046576 012737 015434 005470  MOV  #EM1801,ERRMSG    ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
9497 046604 005037 002650          CLR  ERMRF           ;CLEAR THE ERROR SUMMARY FLAGS.
9498 046610 012700 003452          MOV  #ERCNTB,R0
9499 046614 004737 030214          JSR   PC,CLR16W      ;CLEAR THE ERROR COUNTER TABLE.
9500
9501 ;*
9502 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
9503 ; Clear TX and RX interrupt enable bits in the CSR.
9504 ; This subroutine reports errors >>>> 1801 <<<<.
9505 046620 004737 034530          JSR   PC,RESETT      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9506 046624 103402          BCS  .+6                  ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
9507 046626 000137 047004          JMP  60$           ;YES, EXIT THE TEST.
9508 046632 012737 003412 005466  MOV  #1802,ERRNBR     ;SET THE ERROR REPORT NUMBER TO 1802.
9509
9510 ;*
9511 ; Verify read/write capability to indirect address field of CSR.
9512 ; Use byte accesses.
9513 046640 012702 000017          MOV  #17,R2          ;SET LOOP COUNT.
9514 046644 013704 002300          MOV  CSRA,R4          ;GET CSR ADDRESS.
9515 046650 110214          2$:  MOV  R2,(R4)          ;WRITE COUNT TO CSR.
9516 046652 111401          MOV  (R4),R1              ;READ BACK THE CONTENTS OF THE CSR
9517 046654 042701 177760          BIC  #177760,R1      ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
9518 046660 020102          CMP  R1,R2                ;CHECK FOR CORRECT DATA WRITTEN/READ.
9519 046662 001406          BEQ  4$                    ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
9520 ;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
9521 046664 012737 024736 005472  MOV  #ER1601,ERRBLK    ;SELECT THE PROPER ERROR REPORT ROUTINE.
9522 046672 005003          CLR  R3                    ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
9523 046674 005005          CLR  R5                    ;CAUSE REPORT OF LINE 0.
9524 046676          ERROR          ; >>>> ERROR # 1802 <<<<
9525 046700 005302          4$:  DEC  R2                  ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
9526 046702 002362          BGE  2$                    ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.
9527
9528 ;*
9529 ; Write and verify 16 data patterns in all used bits of all lower bytes of all
9530 ; registers on all active lines. Use READ/WRITE accesses. Before writing
9531 ; each pattern, clear all the used bits of all active registers.
9532 ; REGTST routine reports errors with numbers >>>> ERROR 1803 - 1805 <<<<.
9533 046704 005237 005466          INC  ERRNBR          ;SET THE ERROR NUMBER TO 1803.
9534 046710 012703 177777          MOV  #-1 3          ;INDICATE THAT LO BYTE ACCESSSES ARE TO BE USED.
9535 046714 012704 000002          MOV  #2,R4          ;INDICATE R/W ACCESS, CLEAR FIRST.

```

HARDWARE TEST

- REG8RW -

```

9536 046720 004737 034140      JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
9537                               ;*
9538       ; Write and verify 16 data patterns in all used bits of all high bytes of all
9539       ; registers on all active lines. Use READ/WRITE accesses. Before writing
9540       ; each pattern, clear all the used bits of all active registers.
9541       ; REGTST routine reports errors with numbers >>>> ERROR 1806 - 1808 <<<<.
9542       ;
9543 046724 012737 003416 005466      MOV    #1806.,ERRNBR    ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
9544 046732 005403                  NEG    R3              ;INDICATE THAT HI BYTE ACCESSSES ARE TO BE USED.
9545 046734 004737 034140      JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
9546                               ;*
9547       ; Write and verify 16 data patterns in all used bits of all lower bytes of all
9548       ; registers on all active lines. Use READ/WRITE accesses. Before writing
9549       ; each pattern, set all the used bits of all active registers.
9550       ; REGTST routine reports errors with numbers >>>> ERROR 1809 1811 <<<<.
9551       ;
9552 046740 012737 003421 005466      MOV    #1809.,ERRNBR    ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
9553 046746 005403                  NEG    R3              ;INDICATE THAT LO BYTE ACCESSSES ARE TO BE USED.
9554 046750 005404                  NEG    R4              ;INDICATE R/W ACCESS, SET FIRST.
9555 046752 004737 034140      JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
9556                               ;*
9557       ; Write and verify 16 data patterns in all used bits of all high bytes of all
9558       ; registers on all active lines. Use READ/WRITE accesses. Before writing
9559       ; each pattern, set all the used bits of all active registers.
9560       ; REGTST routine reports errors with numbers >>>> ERROR 1812 - 1814 <<<<.
9561       ;
9562 046756 012737 003424 005466      MOV    #1812.,ERRNBR    ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
9563 046764 005403                  NEG    R3              ;INDICATE THAT HI BYTE ACCESSSES ARE TO BE USED.
9564 046766 004737 034140      JSR    PC,REGTST      ;WRITE AND VERIFY DATA PATTERNS.
9565                               ;*
9566       ; Print error summary reports if necessary.
9567       ; The following routine reports errors with number >>>> ERROR # 1815 <<<<
9568       ;
9569 046772 012737 003427 005466      MOV    #1815.,ERRNBR    ;SET UP ERROR NUMBER FOR NEXT RTN.
9570 047000 004737 034502      JSR    PC,REPSMR      ;REPORT ERROR SUMMARY IF NECESSARY.
9571 047004 005037 002362      CLR    CTRLCF          ;INDICATE THAT WE COMPLETED THE TEST.
9572 047010                  ENDTST
          047010
          047010 104401

```

L10053: TRAP C#ETST

HARDWARE TEST

- REGBRM -

```

9574 .SBTTL HARDWARE TEST - REGBRM -
9575 ;* *****
9576 ;* Device Register Byte Access Read/Modify/Write Test -
9577 ;*
9578 ;* This test verifies that the device registers can be read and written
9579 ;* correctly using byte accesses in Read/Modify/Write mode.
9580 ;*
9581 ;* *****
9582
9583 047012 BGNTST
9584 047012 T17::
9585 000021 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9586 047012 012737 000021 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (19)
9587 047020 012737 177777 002362 MOV #-1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
9588 047026 012737 000001 005464 MOV #1,ERRTYP ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
9589 047034 012737 003555 005466 MOV #1901,ERRNBR ;SET UP ERROR NUMBER IN THE ERROR TABLE.
9590 047050 005037 002650 MOV #EM1901,ERRMSG ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
9591 047054 012700 003452 CLR ERSMRF ;CLEAR THE ERROR SUMMARY FLAGS.
9592 047060 004737 030214 JSR PC,CLR16W ;CLEAR THE ERROR COUNTER TABLE.
9593
9594 ;*
9595 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
9596 ; Clear TX and RX interrupt enable bits in the CSR.
9597 ; This subroutine reports errors >>>> 1901 <<<<.
9598 047064 004737 034530 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9599 047070 103402 BCS .+6 ;FATAL RESET ERROR? NO, CONTINUE WITH TEST.
9600 047072 000137 047254 JMP 60$ ;YES, EXIT THE TEST.
9601 047076 012737 003556 005466 MOV #1902,ERRNBR ;SET THE ERROR REPORT NUMBER TO 1902.
9602
9603 ;*
9604 ; Verify read/write capability to indirect address field of CSR.
9605 ; Use byte accesses.
9606 047104 012702 000017 MOV #17,R2 ;SET LOOP COUNT.
9607 047110 013704 002300 MOV CSRA,R4 ;GET CSR ADDRESS.
9608 047114 142714 000017 2$: BICB #17,(R4) ;CLEAR THE DUT CSR USING READ/MODIFY/WRITE.
9609 047120 150214 BISB R2,(R4) ;WRITE COUNT TO CSR USING READ/MODIFY/WRITE.
9610 047122 111401 MOV B (R4),R1 ;READ BACK THE CONTENTS OF THE CSR
9611 047124 042701 177760 BIC #177760,R1 ;MASK OUT ALL BUT THE IND.ADR.REG FIELD.
9612 047130 020102 CMP R1,R2 ;CHECK FOR CORRECT DATA WRITTEN/READ.
9613 047132 001406 BEQ 4$ ;IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.
9614 ;Report "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (D)."
9615 047134 012737 024736 005472 MOV #ER1601,ERRBLK ;SELECT THE PROPER ERROR REPORT ROUTINE.
9616 047142 005003 CLR R3 ;SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.
9617 047144 005005 CLR R5 ;CAUSE REPORT OF LINE 0.
9618 047146 ERROR ; >>>> ERROR # 1902 <<<<
9619 047150 005302 4$: DEC R2 ;DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.
9620 047152 002360 BGE 2$ ;LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.
9621
9622 ;*
9623 ; Write and verify 16 data patterns in all used bits of all lower bytes of all
9624 ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
9625 ; writing each pattern, clear all the used bits of all active registers.
9626 ; REGTST routine reports errors with numbers >>>> ERROR 1903 - 1905 <<<<.
9627 047154 005237 005466 INC ERRNBR ;SET THE ERROR NUMBER TO 1903.
9628 047160 012703 177777 MOV #-1,R3 ;INDICATE THAT LO BYTE ACCESSSES ARE TO BE USED.

```

HARDWARE TEST

- REGBRM -

```

9629 047164 012704 000001      MOV    #1,R4      ;INDICATE R/M/W ACCESS, CLEAR FIRST.
9630 047170 004737 034140      JSR    PC,REGTST  ;WRITE AND VERIFY DATA PATTERNS.
9631                               ;*
9632                               ; Write and verify 16 data patterns in all used bits of all high bytes of all
9633                               ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
9634                               ; writing each pattern, clear all the used bits of all active registers.
9635                               ; REGTST routine reports errors with numbers >>>> ERROR 1906 - 1908 <<<<.
9636                               ;-
9637 047174 012737 003562 005466      MOV    #1906.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
9638 047202 005403                NEG    R3          ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
9639 047204 004737 034140      JSR    PC,REGTST  ;WRITE AND VERIFY DATA PATTERNS.
9640                               ;*
9641                               ; Write and verify 16 data patterns in all used bits of all lower bytes of all
9642                               ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
9643                               ; writing each pattern, set all the used bits of all active registers.
9644                               ; REGTST routine reports errors with numbers >>>> ERROR 1909 - 1911 <<<<.
9645                               ;-
9646 047210 012737 003565 005466      MOV    #1909.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
9647 047216 005403                NEG    R3          ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
9648 047220 005404                NEG    R4          ;INDICATE R/M/W ACCESS, SET FIRST.
9649 047222 004737 034140      JSR    PC,REGTST  ;WRITE AND VERIFY DATA PATTERNS.
9650                               ;*
9651                               ; Write and verify 16 data patterns in all used bits of all high bytes of all
9652                               ; registers on all active lines. Use READ/MODIFY/WRITE accesses. Before
9653                               ; writing each pattern, set all the used bits of all active registers.
9654                               ; REGTST routine reports errors with numbers >>>> ERROR 1912 - 1914 <<<<.
9655                               ;-
9656 047226 012737 003570 005466      MOV    #1912.,ERRNBR ;SET UP ERROR NUMBER FOR REGTST ROUTINE.
9657 047234 005403                NEG    R3          ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
9658 047236 004737 034140      JSR    PC,REGTST  ;WRITE AND VERIFY DATA PATTERNS.
9659                               ;*
9660                               ; Print error summary reports if necessary.
9661                               ; The following routine reports errors with number >>>> ERROR # 1915 <<<<
9662                               ;-
9663 047242 012737 003573 005466      MOV    #1915.,ERRNBR ;SET UP ERROR NUMBER FOR NEXT RTN.
9664 047250 004737 034502      JSR    PC,REPSMR  ;REPORT ERROR SUMMARY IF NECESSARY.
9665 047254 005037 002362      60$: CLR    CTRLCF  ;INDICATE THAT WE COMPLETED THE TEST.
9666 047260                ENDTST
          047260 104401

```

L10054:

TRAP C\$ETST

HARDWARE TEST

- IDBIT -

```

9668 .SBTTL  HARDWARE TEST          - IDBIT -
9669 ;* *****
9670 ;* - Device Register ID Bit Test -
9671 ;*
9672 ;* This test verifies that the DUT STAT register ID bit reads as clear.
9673 ;*
9674 ;* --- *****
9675
9676 047262      BGNTST
          047262
9677          000022      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9678 047262 012737 000022 002364      MOV  @TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (20)
9679 047270 012737 177777 002362      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
9680 047276 012737 000001 005464      MOV  #1,ERRTYP      ;SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.
9681 047304 012737 003721 005466      MOV  #2001.,ERRNBR      ;SET UP ERROR NUMBER IN THE ERROR TABLE.
9682 047312 012737 015557 005470      MOV  #EM2001,ERRMSG      ;SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.
9683
9684 ;*
9685 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
9686 ; Clear TX and RX interrupt enable bits in the CSR.
9687 ; This subroutine reports errors >>>> 2001 <<<<.
9688 047320 004737 034530      JSR  PC,RESETT      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9689 047324 103016      BCC  60$      ;FATAL RESET ERROR? YES, EXIT THE TEST.
9690
9691 ;*
9692 ; Read the STAT register ID bit and verify that it is clear.
9693 047326 017701 132754      MOV  @STAT,R1      ;READ THE STAT REGISTER CONTENTS.
9694 047332 032701 000400      BIT  @BIT8,R1      ;CHECK THE ID BIT.
9695 047336 001411      BEQ  60$      ;ID BIT CLEAR? YES, EXIT THE TEST.
9696 047340 012737 003722 005466      MOV  #2002.,ERRNBR      ;NO, SET THE ERROR REPORT NUMBER TO 2002.
9697 047346 012701 015621      MOV  #EM2002,R1      ;GET THE PROPER ERROR MESSAGE.
9698 047352 012737 024644 005472      MOV  #ER0503,ERRBLK      ;SELECT THE PROPER ERROR REPORT ROUTINE.
9699 047360      ERROR      ;ERROR NUMBER >>>> 2002 <<<<
          047360 104460
9700 047362 005037 002362      60$: CLR  CTRLCF      ;INDICATE THAT WE COMPLETED THE TEST.
9701 047366      ENDTST
          047366
          047366 104401

```

L10055:

```

TRAP  C$ERROR
TRAP  C$ETST

```


HARDWARE TEST

- NOTXDV -

```

9703
9704
9705
9706
9707
9708
9709
9710
9711
9712 047370
      047370
9713      000023
9714 047370 012737 000023 002364
9715 047376 012737 177777 002362
9716 047404 012737 000001 005464
9717 047412 012737 004065 005466
9718 047420 012737 015674 005470
9719 047426 012737 026440 005472
9720
9721
9722
9723
9724
9725 047434 004737 030172
9726 047440 103054
9727 047442 005237 005466
9728
9729
9730
9731
9732
9733
9734 047446 013705 002272
9735 047452 012700 000200
9736 047456 004737 040250
9737 047462 012700 177670
9738 047466 004737 040324
9739 047472 012704 000012
9740 047476 004737 030370
9741 047502 004737 036352
9742
9743
9744
9745
9746
9747 047506 013705 002272
9748 047512 005004
9749 047514 000241
9750 047516 006005
9751 047520 103020
9752
9753
9754
9755
9756
9757 047522 010477 132552
9758 047526 012777 000012 132546

```

```

.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.
;*
;-- *****
      BGNTS
;+
; T19:
      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV     #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (21)
      MOV     #-1,CTRLCF   ;INDICATE THAT WE ARE IN A TEST.
      MOV     #1,ERRTYP    ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV     #2101,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
      MOV     #EM2101,ERRMSG ;SET ERROR MESSAGE ADDRESS IN 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 >>>> 2101 <<<<.
;--
      JSR     PC,CLNRST    ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCC     60$          ;RESET FAILURE?, ABORT THIS TEST.
      INC     ERRNBR       ;SET THE ERROR NUMBER TO 2102.
;+
; Set internal loopback on all active lines.
; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
; 2 stop bits.
; Disable transmitters on all active lines.
;--
      MOV     ACTLNS,R5    ;PASS THE ACTIVE LINE BIT MAP.
      MOV     #200,R0      ;PASS THE LNCTRL CONTENTS.
      JSR     PC,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 SECS.
      JSR     PC,DELAY     ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
      JSR     PC,TXDSBL    ;DISABLE TRANSMITTERS ON ALL ACTIVE LINES.
;+
; Test all active lines individually.
; Write a data word to the TXCHAR register with TX_DATA_VALID clear.
; Verify no TX_ACTION is generated.
;--
      MOV     ACTLNS,R5    ;GET THE ACTIVE LINE BIT MAP.
      CLR     R4           ;CLEAR THE LINE NUMBER COUNTER.
2$:    CLC               ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
      ROR     R5          ;SHIFT THE BIT MAP INTO THE CARRY BIT.
      BCC     4$          ;DO NOT TEST THE LINE IF IT IS INACTIVE.
;+
; Select the line under test.
; Write data word (ASCII <LF>) to TXCHR register with the most significant
; bit (TX_DATA_VALID) clear.
;--
      MOV     R4,@CSRA     ;SELECT THE LINE CURRENTLY UNDER TEST.
      MOV     #12,@TXCHA   ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.

```

HARDWARE TEST

- NOTXDV -

```

9759          ;+
9760          ; Wait for a TX_ACTION to be returned, report error if TX_ACTION found
9761          ; before time-out occurs.
9762          ;-
9763 047534 012701 170002      MOV     #170002,R1      ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
9764 047540 013702 002300      MOV     CSRA,R2        ;PASS THE ADDRESS OF THE REGISTER TO TEST.
9765 047544 004737 037740      JSR     PC,WAIBIS      ;WAIT FOR TX_ACTION TO COME BACK.
9766 047550 103004            BCC     4$          ;SKIP ERROR REPORT IF TX-ACTION NOT FOUND.
9767
9768 047552 010401            MOV     R4,R1          ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
9769 047554 012702 015737      MOV     #EM2102,R2      ;PASS THE ERROR MESSAGE TO BE REPORTED.
9770
9771 047560            ERROR    ;"TX_ACT FOUND AFTER INVALID DATA WORD WRITTEN"
047560 104460            ;          >>>>> ERROR #4102 <<<<<.
9772                                TRAP     C#ERROR
9773          ;~
9774          ; Verify all active lines have been tested.
9775          ;-
9775 047562 005204      4$:      INC     R4          ;INCREMENT THE LINE NUMBER COUNTER.
9776 047564 005705      TST     R5          ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
9777 047566 001352      BNE     2$          ;YES; BRANCH TO TEST THE NEXT LINE.
9778 047570 000400      BR      60$         ;NO; EXIT THIS TEST.
9779
9780 047572 005037 002362      60$:      CLR     CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
9781 047576      ENDTST
047576
047576 104401
                                L10056:
                                TRAP     C#E1ST

```


HARDWARE TEST

- TXDVAL -

```

9783 .SBTTL HARDWARE TEST - TXDVAL -
9784 ;* *****
9785 ;* - TX_DATA_VALID/TX_ACTION Test -
9786 ;* This test verifies that if a data word is written to the TXCHAR register
9787 ;* with the TX_DATA_VALID bit set, a corresponding TX_ACTION will be
9788 ;* generated.
9789 ;* To ensure data is not accidentally transmitted, the test is performed
9790 ;* in internal loopback, and on all active lines.
9791 ;*
9792 ;*
9793 ;*
9794 047600 BGNTST
9795 047600 T20::
9796 047600 000024 TNUM ** TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9797 047606 012737 000024 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (22)
9798 047614 012737 177777 002362 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
9799 047622 012737 000001 005464 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
9800 047630 012737 004231 005466 MOV #2201.,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
9801 047636 012737 016033 005470 MOV #EM2201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRTABL.
9802 047636 012737 026440 005472 MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
9803 ;*
9804 ; Reset the DUT to a known state, remove the status codes from the fifo.
9805 ; Clear TX and RX interrupt enable bits in the CSR.
9806 ; This subroutine reports error >>>> 2201 <<<<.
9807 047644 004737 030172 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9808 047650 103066 BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.
9809 ;*
9810 ; Set internal loopback on all active lines.
9811 ; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
9812 ; 2 stop bits.
9813 ; Disable transmitters on all active lines.
9814 ;*
9815 047652 013705 002272 MOV ACTLNS,R5 ;PASS THE ACTIVE LINE BIT MAP.
9816 047656 012700 000200 MOV #200,R0 ;PASS THE LNCTRL CONTENTS.
9817 047662 004737 040250 JSR PC,WTWLNLC ;INITIALISE THE LNCTRL REGISTERS.
9818 047666 012700 177670 MOV #177670,R0 ;PASS THE LPR CONTENTS.
9819 047672 004737 040324 JSR PC,WTWLPR ;INITIALISE THE LPR REGISTERS ON ALL LINES.
9820 047676 012704 000012 MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI-SECONDS.
9821 047702 004737 030370 JSR PC,DELAY ;WAIT FOR LNCTR AND LPR REGS TO BE UPDATED.
9822 047706 004737 036352 JSR PC,TXDSBL ;DISABLE TRANSMITTERS ON ALL ACTIVE LINES.
9823 ;*
9824 ; Test all active lines individually.
9825 ; Write a data word to the TXCHAR register with TX DATA_VALID set.
9826 ; Verify that a corresponding TX_ACTION is generated.
9827 ;*
9828 047712 013705 002272 MOV ACTLNS,R5 ;GET THE ACTIVE LINE BIT MAP.
9829 047716 005004 CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
9830 047720 012737 004232 005466 2$: MOV #2202.,ERRNBR ;SET THE ERROR NUMBER TO 2202.
9831 047726 000241 CLC ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
9832 047730 006005 ROR R5 ;SHIFT THE BIT MAP INTO THE CARRY BIT.
9833 047732 103032 BCC 8$ ;DO NOT TEST THE LINE IF IT IS INACTIVE.
9834 ;*
9835 ; Select the line under test.
9836 ; Write data word (ASCII <LF>) to TXCHR register with the most significant
9837 ; bit (TX_DATA_VALID) set.
9838 ;*

```

HARDWARE TEST

- TXDVAL -

```

9839 047734 010477 132340      MOV    R4,0CSRA      ;SELECT THE LINE CURRENTLY UNDER TEST.
9840 047740 012777 100012 132334  MOV    #100012,0TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
9841                               ;+
9842                               ; Wait for a TX_ACTION to be returned, report error if no TX_ACTION
9843                               ; found before time-out occurs.
9844                               ; -
9845 047746 012701 170002      MOV    #170002,R1      ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
9846 047752 013702 002300      MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
9847 047756 004737 037740      JSR    PC,WAIBIS    ;WAIT FOR TX_ACTION TO COME BACK.
9848 047762 103403             BCS    4$          ;SKIP ERROR REPORT IF TX-ACTION FOUND.
9849 047764 012702 016070      MOV    #EM2202,R2      ;PASS THE ERROR MESSAGE TO BE REPORTED.
9850                               ; "NO TX_ACT FOUND AFTER VALID DATA WORD TX'D".
9851 047770 000411             BR     6$          ;GO REPORT THE ERROR.
9852                               ;+
9853                               ; Verify TX_ACTION returned from correct line.
9854                               ; -
9855 047772 005237 005466      4$: INC    ERRNBR      ;INCREMENT ERROR NUMBER TO 2103.
9856 047776 000302             SWAB   R2          ;GET THE LINE NUMBER IN THE LOW BYTE.
9857 050000 042702 177760      BIC    #177760,R2    ;CLEAR THE UNWANTED BITS.
9858 050004 020204             CMP    R2,R4        ;IS IT THE CORRECT LINE NUMBER?.
9859 050006 001404             BEQ    8$          ;YES; SKIP THE ERROR REPORT.
9860 050010 012702 016162      MOV    #EM2203,R2    ;PASS THE ERROR MESSAGE TO BE REPORTED.
9861                               ; "INCORRECT LINE # RETURNED WITH TX_ACT"
9862 050014 010401             6$: MOV    R4,R1      ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
9863 050016             ERROR                      ; >>>> ERROR <<<<<
9864                               TRAP    C$ERROR
9865                               ;+
9866                               ; Verify all active lines have been tested.
9867                               ; -
9868 050020 005204             8$: INC    R4        ;INCREMENT THE LINE NUMBER COUNTER.
9869 050022 005705             TST    R5          ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
9870 050024 001335             BNE    2$          ;YES; BRANCH TO TEST THE NEXT LINE.
9871                               ; -
9872 050026 005037 002362      60$: CLR    CTRLCF    ;INDICATE THAT WE ARE NOT WITHIN A TEST.
9873 050032             ENDTST
9874 050032             L10057:
9875 050032 104401             TRAP    C$ETST

```

HARDWARE TEST

- TXENBI-

```

9875 .SBTTL HARDWARE TEST - TXENBI-
9876 ;++ *****
9877 ;* - TX_ENABLE (Inactive) Test -
9878 ;* This test verifies that when the line under test's TX_ENABLE bit is
9879 ;* clear, transmission will not take place on that line.
9880 ;* This test is performed in internal loopback, and on all active lines.
9881 ;*
9882 ;-- *****
9883
9884 050034 BGNTST
9885 050034
9885 000025 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9886 050034 012737 000025 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (23)
9887 050042 012737 177777 002362 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
9888 050050 012737 000001 005464 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
9889 050056 012737 004375 005466 MOV #2301,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
9890 050064 012737 016267 005470 MOV #EM2301,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRTABL.
9891 050072 012737 026440 005472 MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
9892
9893 ;+
9894 ; Reset the DUT to a known state, remove the status codes from the fifo.
9895 ; Clear TX and RX interrupt enable bits in the CSR.
9896 ; This subroutine reports error >>>> 2301 <<<<.
9897 050100 004737 030172 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9898 050104 103110 BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.
9899
9900 ;+
9901 ; Set internal loopback on all active lines.
9902 ; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
9903 ; 2 stop bits.
9904 ; Enable transmitters on all lines.
9905 050106 013705 002272 MOV ACTLNS,R5 ;PASS THE ACTIVE LINE BIT MAP.
9906 050112 012700 000200 MOV #200,R0 ;PASS THE LNCTRL CONTENTS.
9907 050116 004737 040250 JSR PC,WTWLCN ;INITIALISE THE LNCTRL REGISTERS.
9908 050122 012700 177670 MOV #177670,R0 ;PASS THE LPR CONTENTS.
9909 050126 004737 040324 JSR PC,WTWLPR ;INITIALISE THE LPR REGISTERS ON ALL LINES.
9910 050132 012704 000012 MOV #10,R4 ;PASS DELAY TIME OF 10 MILLI-SECONDS.
9911 050136 004737 030370 JSR PC,DELAY ;WAIT FOR LNCTR AND LPR REGS TO BE UPDATED.
9912 050142 012705 000377 MOV #MAPLNS,R5 ;PASS THE BIT MAP CORRESPONDING TO ALL LINES.
9913 050146 004737 036446 JSR PC,TXENBL ;ENABLE TRANSMITTERS ON ALL LINES.
9914
9915 ;+
9916 ; Test all active lines individually.
9917 ; Disable transmission on each active line.
9918 050152 012703 000001 MOV #1,R3 ;SET UP THE LINE BIT MAP FOR CHANNEL 0.
9919 050156 005004 CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
9920 050160 012737 004376 005466 2$: MOV #2302,ERRNBR ;SET THE ERROR NUMBER TO 2302.
9921 050166 030337 002272 BIT R3,ACTLNS ;CHECK IF THE LINE IS ACTIVE.
9922 050172 001447 BEQ 6$ ;SKIP TESTING THIS LINE IF IT IS INACTIVE.
9923
9924 ;+
9925 ; Clear the TX_ENABLE bit in TBUFAD2 register.
9926 ; Select the line under test.
9927 ; Verify it is clear, report error if set.
9928 050174 010305 MOV R3,R5 ;PASS THE BIT MAP OF THE LINE UNDER TEST.
9929 050176 004737 036352 JSR PC,TXDSBL ;DISABLE TRANSMISSION ON THE LINE UNDER TEST.
9930 050202 010477 132072 MOV R4,@CSRA ;SELECT THE LINE CURRENTLY UNDER TEST.

```

HARDWARE TEST

- TXENBI-

```

9931 050206 005777 132102      TST    @TXAD2A      ;VERIFY THE TX_ENABLE BIT IS SET.
9932 050212 100433              BMI    4$              ;GO REPORT ERROR IF TX_ENABLE BIT SET.
9933                          ;*
9934                          ; Write data word (ASCII <LF>) to TXCHR register.
9935                          ; Wait for a TX_ACTION to be returned, report error if a TX_ACTION
9936                          ; is found before time-out occurs.
9937                          ;-
9938 050214 012737 004377 005466  MOV    @2303.,ERRNBR ;SET ERROR NUMBER TO 2303.
9939 050222 012777 100012 132052  MOV    @100012,@TXCHA ;WRITE THE DATA WORD TO THE OUT'S TXCHAR REG.
9940 050230 012701 170002          MOV    @170002,R1      ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
9941 050234 013702 002300          MOV    CSRA,R2         ;PASS THE ADDRESS OF THE REGISTER TO TEST.
9942 050240 004737 037740          JSR     PC,WAIBIS        ;WAIT FOR TX_ACTION TO COME BACK.
9943 050244 103016              BCC     4$              ;GO REPORT ERROR IF NO TX-ACTION FOUND.
9944                          ;*
9945                          ; Wait for the data to appear in the fifo, report error if data found.
9946                          ;-
9947 050246 005237 005466          INC     ERRNBR          ;SET ERROR NUMBER TO 2304.
9948 050252 012701 070012          MOV    @70012,R1        ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
9949 050256 013702 002300          MOV    CSRA,R2         ;PASS THE ADDRESS OF THE REGISTER TO TEST.
9950 050262 004737 037740          JSP     PC,WAIBIS        ;WAIT FOR RX_DATA_AVAILABLE TO SET.
9951 050266 103405              BCS     4$              ;REPORT ERROR IF DATA RECEIVED IN THE FIFO.
9952 050270 005237 005466          INC     ERRNBR          ;SET ERROR NUMBER TO 2305.
9953 050274 017702 132002          MOV    @RBUFA,R2        ;READ THE DATA FROM THE FIFO.
9954 050300 100004              BPL     6$              ;SKIP ERROR REPORT IF DATA IS THERE.
9955
9956 050302 010401              4$:   MOV    R4,R1          ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
9957 050304 012702 016325          MOV    @EM2302,R2        ;PASS THE MESSAGE TO BE REPORTED.
9958
9959 050310              ERROR      ;"TX_ENABLE BIT BAD ON LINE: nn".
9959 050310 104460              ;          >>>> ERROR <<<<<.
9960                                  TRAP    C$ERROR
9961
9962                          ;*
9962                          ; Verify all active lines have been tested.
9963                          ;-
9963 050312 000241              6$:   CLC                  ;CLEAR THE CARRY BIT PRIOR TO ROTATION.
9964 050314 006103              ROL     R3                  ;SHIFT THE BIT MAP FOR THE NEXT LINE.
9965 050316 005204              INC     R4                  ;INCREMENT THE LINE NUMBER COUNTER.
9966 050320 020427 000010          CMP    R4,@NUMLNS        ;HAVE ALL THE LINES BEEN TESTED?.
9967 050324 002715              BLT     2$              ;NO; BRANCH TO TEST THE NEXT LINE.
9968
9969 050326 005037 002362          60$:  CLR     CTRLCF        ;INDICATE THAT WE ARE NOT WITHIN A TEST.
9970 050332          ENDTST
9970 050332
9970 050332 104401              L10060:
9970 050332              TRAP    C$ETST

```

HARDWARE TEST

- TXENBA -

SEQ 0258

```

9972 .SBTTL HARDWARE TEST - TXENBA-
9973 ;* *****
9974 ;* - TX_ENABLE (Active) Test -
9975 ;* This test verifies that when the TX_ENABLE bit is set in the appropriate
9976 ;* line register, transmission will take place on that line.
9977 ;* This test is performed in internal loopback, and on all active lines.
9978 ;*
9979 ;*
9980 ;* *****
9981 050334 BGNTST
9982 050334 T22::
9983 050334 000026 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
9984 050334 012737 000026 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (24)
9985 050334 012737 177777 002362 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
9986 050334 012737 000001 005464 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
9987 050334 012737 004541 005466 MOV #2401,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
9988 050334 012737 016363 005470 MOV #EM2401,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRBL.
9989 050334 012737 026440 005472 MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
9990 ;*
9991 ; Reset the DUT to a known state, remove the status codes from the fifo.
9992 ; Clear TX and RX interrupt enable bits in the CSR.
9993 ; This subroutine reports error >>>> 2401 <<<<.
9994 050400 004737 030172 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
9995 050404 103127 BCC 604 ;RESET FAILURE?, ABORT THIS TEST.
9996 ;*
9997 ; Set internal loopback on all active lines.
9998 ; Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
9999 ; 2 stop bits.
10000 ; Disable transmitters on all lines.
10001 ;*
10002 050406 013705 002272 MOV ACTLNS,R5 ;PASS THE ACTIVE LINE BIT MAP.
10003 050412 012700 000200 MOV #200,R0 ;PASS THE LNCTRL CONTENTS.
10004 050416 004737 040250 JSR PC,WTWLCN ;INITIALISE THE LNCTRL REGISTERS.
10005 050422 012700 177670 MOV #177670,R0 ;PASS THE LPR CONTENTS.
10006 050426 004737 040324 JSR PC,WTWLPR ;INITIALISE THE LPR REGISTERS ON ALL LINES.
10007 050432 012704 000012 MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI-SECONDS.
10008 050436 004737 030370 JSR PC,DELAY ;WAIT FOR LNCTR AND LPR REGS TO BE UPDATED.
10009 050442 012705 000377 MOV #MAPLNS,R5 ;PASS THE BIT MAP CORRESPONDING TO ALL LINES.
10010 050446 004737 036352 JSR PC,TXDSBL ;DISABLE TRANSMITTERS ON ALL LINES.
10011 ;*
10012 ; Test all active lines individually.
10013 ; Enable transmission on each active line.
10014 ;*
10015 050452 012703 000001 MOV #1,R3 ;SET UP THE LINE BIT MAP FOR CHANNEL 0.
10016 050456 005004 CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
10017 050460 012737 004542 005466 24: MOV #2402,ERRNBR ;SET THE ERROR NUMBER TO 2402.
10018 050466 030337 002272 BIT R3,ACTLNS ;CHECK IF THE LINE IS ACTIVE.
10019 050472 001463 BEQ 84 ;SKIP TESTING THIS LINE IF IT IS INACTIVE.
10020 ;*
10021 ; Select the line under test.
10022 ; Set the TX_ENABLE bit in TBUFFAD2 register.
10023 ; Verify it is set, report error if clear.
10024 ;*
10025 050474 010305 MOV R3,R5 ;PASS THE BIT MAP OF THE LINE UNDER TEST.
10026 050476 004737 036446 JSR PC,TXENBL ;ENABLE TRANSMISSION ON THE LINE UNDER TEST.
10027 050502 012705 000012 MOV #10.,R5 ;SET TXCHAR/LOOP COUNT TO 10.

```

HARDWARE TEST

- TXENBA -

```

10028 050506 010477 131566      MOV    R4,0CSRA      ;SELECT THE LINE CURRENTLY UNDER TEST.
10029 050512 005777 131576      TST     0TXAD2A      ;VERIFY THE TX_ENABLE BIT IS SET.
10030 050516 100045              BPL      6#           ;GO REPORT ERROR IF TX_ENABLE BIT CLEAR.
10031
10032      ;+
10033      ; Write data word (ASCII <LF>) to TXCHR register.
10034      ; Wait for a TX_ACTION to be returned, report error if no TX_ACTION
10035      ; found before time-out occurs.
10036 050520 012737 004543 005466 4# :  MOV    #2403.,ERRNBR    ;SET ERROR NUMBER TO 2403.
10037 050526 012777 100012 131546      MOV    #100012,0TXCHA ;WRITE THE DATA WORD TO THE DUT'S TXCHAR REG.
10038 050534 012701 170002              MOV    #170002,R1      ;TEST BIT 15, TIMEOUT OF 2 MILLI SECS.
10039 050540 013702 002300              MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
10040 050544 004737 037740              JSR     PC,WAIBIS    ;WAIT FOR TX_ACTION TO COME BACK.
10041 050550 103030              BCC      6#           ;GO REPORT ERROR IF NO TX-ACTION FOUND.
10042
10043      ;+
10044      ; Wait for the data to appear in the fifo, report error if time-out.
10045 050552 005237 005466      INC     ERRNBR      ;SET ERROR NUMBER TO 2404.
10046 050556 012701 070012      MOV    #70012,R1      ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
10047 050562 013702 002300              MOV    CSRA,R2      ;PASS THE ADDRESS OF THE REGISTER TO TEST.
10048 050566 004737 037740              JSR     PC,WAIBIS    ;WAIT FOR RX_DATA_AVAILABLE TO SET.
10049 050572 103017              BCC      6#           ;REPORT ERROR IF NO DATA RECEIVED IN THE FIFO.
10050 050574 005237 005466      INC     ERRNBR      ;SET ERROR NUMBER TO 2405.
10051 050600 017702 131476      MOV    0RBUFA,R2    ;READ THE DATA FROM THE FIFO.
10052 050604 100012              BPL      6#           ;GO REPORT ERROR IF THER IS'NT ANY DATA THERE.
10053 050606 005237 005466      INC     ERRNBR      ;SET ERROR NUMBER TO 2406.
10054 050612 000302              SWAB    R2           ;PUT THE LINE NUMBER IN THE LOW BYTE.
10055 050614 042702 177760      BIC     #177760,R2   ;CLEAR THE UNWANTED BITS.
10056 050620 020204              CMP     R2,R4        ;DID THE DATA COME FROM THE CORRECT LINE?.
10057 050622 001003              BNE     6#           ;NO, GO REPORT THE ERROR.
10058 050624 005305              DEC     R5          ;DECREMENT THE TXCHAR/LOOP COUNTER.
10059 050626 001334              BNE     4#           ;LOOP TO TX THE NEXT CHAR.
10060 050630 000404              BR      8#          ;GO TEST THE NEXT LINE.
10061
10062 050632 010401              6# :    MOV    R4,R1          ;PASS THE NUMBER OF CURRENT LINE UNDER TEST.
10063 050634 012702 016325      MOV    #EM2302,R2    ;PASS THE MESSAGE TO BE REPORTED.
10064
10065 050640              ERROR      ; "TX_ENABLE BIT BAD ON LINE: nn".
10066 050640 104460              ;          >>>>> ERROR <<<<<.
10067
10068      ;+
10069      ; Verify all active lines have been tested.
10070 050642 010305      8# :    MOV    R3,R5          ;PASS THE BIT MAP OF THE LINE UNDER TEST.
10071 050644 004737 036352      JSR     PC,TXDS8L    ;CLEAR THE TX_ENABLE BIT ON THIS LINE.
10072 050650 000241              CLC              ;CLEAR THE CARRY BIT PRIOR TO ROTATION.
10073 050652 006103              ROL     R3          ;SHIFT THE BIT MAP FOR THE NEXT LINE.
10074 050654 005204              INC     R4          ;INCREMENT THE LINE NUMBER COUNTER.
10075 050656 020427 000010      CMP     R4,#NUMLNS  ;HAVE ALL THE LINES BEEN TESTED?.
10076 050662 002676              BLT     2#           ;NO, BRANCH TO TEST THE NEXT LINE.
10077 050664 005037 002362      60# :   CLR     CTRLCF    ;INDICATE THAT WE ARE NOT WITHIN A TEST.
10078 050670              ENDTST
10079 050670
10080 050670 104401              L10061:
10081

```

TRAP C0ETST

HARDWARE TEST

- INTA -

```

10080 .SBTTL  HARDWARE TEST          - INTA -
10081 ;* *****
10082 ;*          - Interrupt Test -
10083 ;*      This test verifies that the Device Under Test (DUT) will generate
10084 ;*      reception and transmission interrupts correctly.  This test does
10085 ;*      not depend on the use of the serial line transmission or reception
10086 ;*      capabilities of the DUT.  The lines are put in internal loopback
10087 ;*      to minimize any external effects that could be caused on devices
10088 ;*      attached to the serial lines.
10089 ;*
10090 ;* *****
10091 ;*
10092 050672      BGNTST
10093 050672      T23::
10093 000027      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
10094 050672 012737 000027 002364      MOV  #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (26)
10095 050700 012737 177777 002362      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
10096 050706 012737 000001 005464      MOV  #1,ERRTYP      ;SET ERROR FATAL ERROR TYPE IN ERROR TABLE.
10097 050714 012737 003101 005466      MOV  #1601.,ERRNBR      ;SET FIRST ERROR REPORT NUMBER IN ERROR TABLE.
10098 050722 012737 016417 005470      MOV  #EM2601.,ERRMSG ;SET TEST ERROR MESSAGE IN ERROR TABLE.
10099 ;*
10100 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
10101 ; Clear TX and RX interrupt enable bits in the CSR.
10102 ; This subroutine reports errors from >>>> 2601 thru 2602 <<<<.
10103 ;*
10104 050730 004737 034530      JSR    PC,RESETT      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
10105 050734 103402          BCS     2$            ;SKIP AROUND ABORTING TEST IF NO ERROR FOUND.
10106 050736 000137 051676      JMP     60$        ;ABORT TEST IF FATAL ERROR FOUND DURING RESET.
10107 050742 012737 005053 005466 2$: MOV  #2603.,ERRNBR ;SET THE ERROR REPORT NUMBER TO 2603.
10108 ;*
10109 ; Enable transmitters on all lines.
10110 ;*
10111 050750 012705 000377      4$: MOV  #MAPLNS,R5      ;PASS ACTIVE LINE BIT MAP.
10112 050754 004737 036446      JSR    PC,TXENBL      ;ENABLE TRANSMISSION ON ALL LINES.
10113 ;*
10114 ; Test reception interrupts.
10115 ; Set up for RX and TX interrupts:
10116 ; RX interrupt service routine inputs a char and counts the interrupt.
10117 ; TX interrupt service routine counts TX interrupts.
10118 ;*
10119 050760 005037 002412      CLR     RXINTC      ;CLEAR THE RX INTERRUPT COUNTER.
10120 050764 005037 002414      CLR     RXINTF      ;CLEAR THE RX INTERRUPT FLAGS.
10121 050770 005037 002416      CLR     TXINTC      ;CLEAR THE TX INTERRUPT COUNTER.
10122 050774 012737 004012 002360      MOV  #BUF BAS,BUFPTR ;LOAD THE BUFFER PTR WITH THE BUFFER BASE ADR.
10123 051002 106427 000240      MTPS   #PRIOS      ;DISABLE DEVICE 1  RRUPTS.
10124 051006      SETVEC  RXVECA,#RXINPT,#PRIOS ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
10124 051006 012746 000240      MOV     #PRIOS,(SP)
10124 051012 012746 040734      MOV     #RXINPT,-(SP)
10124 051016 013746 002266      MOV     RXVECA,-(SP)
10124 051022 012746 000003      MOV     #3,(SP)
10124 051026 104437          TRAP    C$SVEC
10124 051030 062706 000010      ADD     #10,SP
10125 051034      SETVEC  TXVECA,#CACHTX,#PRIOS ;SET UP INTERRUPT VECTOR TO CATCH TX INT.
10125 051034 012746 000240      MOV     #PRIOS,-(SP)
10125 051040 012746 040402      MOV     #CACHTX,-(SP)
10125 051044 013746 002270      MOV     TXVECA,(SP)
10125 051050 012746 000003      MOV     #3,(SP)

```

- INTA -

```

MTPS      *PRI00      ;ALLOW INTERRUPTS.

```

```
TRAP      CSVEC
ADD       #10,SP
```

```

;*
;Enable reception interrupts.
;Delay 4 ms to allow time for the interrupts to take place.
;Disable reception interrupts.
;-
    JSR     PC,RXIE1      ;ENABLE THE RECEPTIO. 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$:   ERRDF   2603,EM2601,ER0503;          >>>>> ERROR #2603 <<<<<.
                                           TRAP      C$ERDF
                                           .WORD      2603
                                           .WORD      EM2601
                                           .WORD      ER0503
;*
; Verify that no TX interrupts have been generated so far in this test.
;
10$:  MOV     TXINTC,R2     ;LOAD # OF TX INTERRUPTS FOR ER0504 RTN.
    BEQ     12$          ;SKIP ERROR IF NO TX INTERRUPTS.
;REPORT "TX INTERRUPTS(S) RECEIVED WITH TX INTERRUPTS DISABLED."
    MOV     #EM0526,R1     ;SET UP MESSAGE ADR FOR INDIRECT PRINT.
    ERRDF   2604,EM2601,ER0504;          >>>>> ERROR #2604 <<<<<.
                                           TRAP      C$ERDF
                                           .WORD      2604
                                           .WORD      EM2601
                                           .WORD      ER0504
;*
; Clean out the interrupt vectors used in this test.

```


HARDWARE TEST

INTA -

```

10173
10174 051216 106427 000240      12$: MTPS  #PRI05      ;DISABLE DEVICE INTERRUPTS.
10175 051222      013700 002266      CLRVEC RXVECA      ;RETURN RX INT VECTOR TO UNUSED POOL.
                                MOV      RXVECA,R0
                                TRAP     C$CVEC
10176 051230      013700 002270      CLRVEC TXVECA      ;RETURN TX INT VECTOR TO UNUSED POOL.
                                MOV      TXVECA,R0
                                TRAP     C$CVEC
10177
10178      ;*
10179      ; Test transmission interrupts.
10180      ; Set up for RX and TX interrupts:
10181      ; RX interrupt service routine counts RX interrupts.
10182      ; TX interrupt service routine counts the interrupt and sets flags.
10183 051236 005037 002412      CLR      RXINTC      ;CLEAR THE RX INTERRUPT COUNTER.
10184 051242 005037 002416      CLR      TXINTC      ;CLEAR THE TX INTERRUPT COUNTER.
10185 051246 005037 002420      CLR      TXINTF      ;CLEAR THE RX INTERRUPT FLAGS.
10186 051252      012746 000240      SETVEC RXVECA,#CACHRX,#PRI05 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
                                MOV      #PRI05,-(SP)
                                MOV      #CACHRX,-(SP)
                                MOV      RXVECA,-(SP)
                                MOV      #3,-(SP)
                                TRAP     C$SVEC
                                ADD      #10,SP
10187 051300      012746 000240      SETVEC TXVECA,#TXINTR,#PRI05 ;SET UP INT VECTOR TO TX INT ROUTINE.
                                MOV      #PRI05,(SP)
                                MOV      #TXINTR,-(SP)
                                MOV      TXVECA,(SP)
                                MOV      #3,(SP)
                                TRAP     C$SVEC
                                ADD      #10,SP
10188 051326 106427 000000      MTPS  #PRI00      ;ALLOW INTERRUPTS.
10189
10190      ;*
10191      ; Verify that the TX_ACTION bit is clear.
10192 051332 012705 000022      MOV      #18.,R5      ;INITIALIZE THE LOOP COUNTER.
10193 051336 012701 000144      MOV      #100.,R1      ;SET 100 MS TIME-OUT.
10194 051342 012702 100000      MOV      #BIT15,R2      ;SELECT TX_ACTION BIT TO TEST.
10195 051346 013704 002300      MOV      CSRA,R4      ;PASS OUT CSR AS THE WORD TO TEST.
10196 051352 012703 100000      14$: MOV      #BIT15,R3      ;WAIT FOR TX_ACTION TO BE SET.
10197 051356 004737 032042      JSR      PC,MSLOOP      ;WAIT UP TO 100 MS FOR TX_ACTION SET.
10198 051362 103020      BCC      20$      ;IF TIME-OUT, CONSIDER TX_ACTION CLEAR.
10199 051364 005003      CLR      R3      ;NOW, WAIT FOR TX_ACTION CLEAR.
10200 051366 004737 032042      JSR      PC,MSLOOP      ;WAIT UP TO 100 MS FOR TX_ACTION CLEAR.
10201 051372 103005      BCC      16$      ;IF TIME-OUT, REPORT TX ACTION WON'T CLEAR.
10202 051374 005305      DEC      R5      ;DECREMENT THE TX_ACTION SET COUNTER.
10203 051376 001365      BNE      14$      ;LOOP IF NOT TOO MANY TX_ACTIONS FOUND.
10204      ;REPORT "TX_ACTION SET REPEATEDLY AFTER RESET, NO DATA SENT."
10205 051400 012701 017044      MOV      #EM2607,R1      ;SELECT ERROR MESSAGE.
10206 051404 000402      BR      18$      ;GO TO REPORT THE ERROR.
10207 051406 012701 017140      16$: MOV      #EM2608,R1      ;SELECT TX_ACTION STUCK SET MSG.
10208 051412      18$: ERRDF 2605,EM2606,ER0503;      >>>>> ERROR #2605 <<<<<.
                                TRAP     C$ERDF
                                .WORD    2605
                                .WORD    EM2606
                                .WORD    ER0503
10209 051422 000424      BR      24$      ;GO TO TEST WITH TX_ACTION SET.

```

HARDWARE TEST

- INTA -

```

10210
10211
10212
10213 051424 004737 036574
10214 051430 012704 000062
10215 051434 004737 030370
10216 051440 005737 002416
10217 051444 001413
10218 051446 012701 017044
10219 051452 005737 002420
10220 051456 100002
10221 051460 012701 017211
10222
10223 051464
      051464 104455
      051466 005056
      051470 017005
      051472 024644
10224
10225
10226
10227 051474 005037 002416
10228 051500 005037 002420
10229
10230
10231
10232 051504 012705 000377
10233 051510 012700 000200
10234 051514 004737 040250
10235 051520 012700 156430
10236 051524 004737 040324
10237
10238
10239
10240 051530 013701 002302
10241 051534 012702 100000
10242 051540 004737 040300
10243
10244
10245
10246 051544 012704 000372
10247 051550 004737 030370
10248
10249
10250
10251 051554 005737 002416
10252 051560 001007
10253
10254
10255
10256 051562 012701 017270
10257 051566 005777 130506
10258 051572 100407
10259 051574 012701 017362
10260
10261
10262

;+
; 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$: ERDF 2606,EM2606,ER0503; >>>> ERROR #2606 <<<<.
                                     TRAP C$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,WTWLCN ;DISABLE RECPTION AND DMA, ETC. ON DUT.
      MOV #156430,R0 ;SPECIFY 9600BPS,1STOP,NO PARITY,8BITS/CHAR.
      JSR PC,WTWLPR ;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,WTWLNS ;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 -

```

10263 051600 005737 002420      26$:   TST   TXINT+      ;CHECK THE TX INTERRUPT FLAGS.
10264 051604 100006              BPL     30$          ;SKIP ERROR IF TX_ACTION CLR FLAG IS CLEAR.
10265 051606 012701 017211      MOV     @EM2609,R1      ;SET UP TX INT WITH "TX_ACTION CLR" MSG.
10266                               ;+
10267                               ; Report "TRANSMIT INTERRUPT TEST ERROR:...."
10268                               ;+
10269 051612 28$:   ERDF    2607,EM2606,ER0503;          >>>> ERROR #2607 <<<<.
                                TRAP     C$ERDF
                                .WORD    2607
                                .WORD    EM2606
                                .WORD    ER0503
                                051612 104455
                                051614 005057
                                051616 017005
                                051620 024644
10270                               ;+
10271                               ; Verify that no TX interrupts have been generated so far in this test.
10272                               ;+
10273 051622 013702 002412      30$:   MOV     RXINTC,R2      ;LOAD # OF RX INTERRUPTS FOR ER0504 RTN.
10274 051626 001406              BEQ     32$          ;SKIP ERROR IF NO RX INTERRUPTS.
10275 051630 012701 012557      MOV     @EM0525,R1      ;SET UP MESSAGE ADR FOR INDIRECT PRINT.
10276                               ;REPORT "RX INTERRUPTS(S) RECEIVED WITH RX INTERRUPTS DISABLED."
10277 051634      ERDF    2608,EM2606,ER0504;          >>>> ERROR #2608 <<<<.
                                TRAP     C$ERDF
                                .WORD    2608
                                .WORD    EM2606
                                .WORD    ER0504
                                051634 104455
                                051636 005057
                                051640 017005
                                051642 024670
10278                               ;+
10279                               ; Disable interrupts and clean out the interrupt vectors used in this test.
10280                               ;+
10281 051644 005001      32$:   CLR     R1              ;CLEAR BOTH TRANSMITTER
10282 051646 004737 036542      JSR     PC,TXIE0        ; INTERRUPT ENABLE AND RECEIVER
10283 051652 004737 035546      JSR     PC,RXIE0        ; INTERRUPT ENABLE BITS IN THE DUT CSR.
10284 051656 106427 000240      MTPS    @PRI05         ;DISABLE DEVICE INTERRUPTS.
10285 051662      CLRVEC  RXVECA      ;RETURN RX INT VECTOR TO UNUSED POOL.
                                MOV     RXVECA,R0
                                TRAP     C$CVEC
                                051662 013700 002266
                                051666 104436
10286 051670      CLRVEC  TXVECA      ;RETURN TX INT VECTOR TO UNUSED POOL.
                                MOV     TXVECA,R0
                                TRAP     C$CVEC
                                051670 013700 002270
                                051674 104436
10287                               ;+
10288 051676 005037 002362      60$:   CLR     CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
10289                               ;+
10290 051702      ENDTST
                                L10062:
                                051702 104401          TRAP     C$ETST
10291

```

HARDWARE TEST

- BRLEVA -

```

10293 .SBTTL  HARDWARE TEST          - BRLEVA -
10294 ;* *****
10295 ;* - BR Level Test B -
10296 ;* This test verifies that the Device Under Test (DUT) will generate
10297 ;* reception and transmission interrupts at the correct BR level.
10298 ;* This test does not depend on the use of the serial line transmission
10299 ;* or reception capabilities of the DUT. The lines are put in internal
10300 ;* loopback to minimize any external effects that could be caused on
10301 ;* devices attached to the serial lines.
10302 ;*
10303 ;* *****
10304 ;*
10305 051704 BGNTST
10306 051704
10306 10306 000030 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
10307 051704 012737 000030 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (30)
10308 051712 012737 177777 002362 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
10309 051720 012737 000001 005464 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
10310 051726 012737 005671 005466 MOV #3001.,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
10311 051734 012737 017445 005470 MOV #EM3001,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERR_TBL.
10312 051742 005037 002650 CLR ERS_MRF ;INITIALIZE THE "REPORT ERROR SUMMARY" FLAGS.
10313 ;*
10314 ; Reset the DUT to a known state, do not remove the status codes from the fifo.
10315 ; Clear TX and RX interrupt enable bits in the CSR.
10316 ; This subroutine reports errors from >>>> 3001 thru 3002 <<<<.
10317 ;*
10318 051746 004737 034530 JSR PC,RESETT ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
10319 051752 103402 BCS 2$ ;SKIP AROUND ABORTING TEST IF NO ERROR FOUND.
10320 051754 000137 052544 JMP 60$ ;ABORT TEST IF FATAL ERROR FOUND DURING RESET.
10321 051760 012737 005673 005466 2$: MOV #3003.,ERRNBR ;SET THE ERROR REPORT NUMBER TO 3003.
10322 ;*
10323 ; Enable transmitters on all lines.
10324 ;*
10325 051766 012705 000377 4$: MOV #MAPLNS,R5 ;PASS ACTIVE LINE BIT MAP.
10326 051772 004737 036446 JSR PC,TXENBL ;ENABLE TRANSMISSION ON ALL LINES.
10327 ;*
10328 ; Generate a transmission interrupt request.
10329 ; Processor priority should be at 7 disabling ints.
10330 ;*
10331 051776 106427 000340 MTPS #PRI07 ;DISABLE ALL INTERRUPTS.
10332 052002 SETVEC TXVECA,#TXINTR,#PRI07 ;SET UP INTERRUPT VECTOR TO CATCH TX INT.
10332 052002 012746 000340 MOV #PRI07,-(SP)
10332 052006 012746 041216 MOV #TXINTR,-(SP)
10332 052012 013746 002270 MOV TXVECA,-(SP)
10332 052016 012746 000003 MOV #3,-(SP)
10332 052022 104437 TRAP C$SVEC
10332 052024 062706 000010 ADD #10,SP
10333 ;*
10334 ; Set up DUT for transmission interrupts:
10335 ; Set up internal loopback.
10336 ; Set up line parameters for transmission.
10337 ;*
10338 052030 012705 000377 MOV #MAPLNS,R5 ;PASS ACTIVE LINES BIT MASK.
10339 052034 012700 000200 MOV #200,R0 ;PASS INERT STATE, INTERNAL LOOPBACK.
10340 052040 004737 040250 JSR PC,WTW_LNC ;DISABLE RECEPTION AND DMA, ETC. ON DUT.
10341 052044 012700 156430 MOV #156430,R0 ;SPECIFY 9600BPS,1STOP,NO PARITY,8BITS/CHAR.
10342 052050 004737 040324 JSR PC,WTW_LPR ;WRITE INTO ALL LPR REGISTERS.

```

HARDWARE TEST

- BRLEVA -

```

10343
10344      ;+
10345      ; Send a null char to each line.
10346      ;-
10346 052054 013701 002302      MOV    TXCHA,R1      ;SET UP TXCHAR REGISTER ADDRESS.
10347 052060 012702 100000      MOV    #100000,R2      ;SET CHARACTER TO BE TRANSMITTED = NULL.
10348 052064 004737 040300      JSR     PC,WTWLS      ;SEND NULL CHAR TO EACH LINE.
10349
10350      ;+
10351      ; Delay 50 ms to allow time for the interrupt to be generated.
10352      ;-
10352 052070 012704 000062      MOV    #50.,R4      ;PASS 50 MS TIME TO THE DELAY ROUTINE.
10353 052074 004737 030370      JSR     PC,DELAY      ;DELAY 50 MILLI-SECONDS.
10354
10355      ;+
10356      ; Generate a reception interrupt request.
10357      ;-
10357 052100      SETVEC  RXVECA,#RXBRRT,#PRI07 ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
10357 052100 012746 000340      MOV    #PRI07,-(SP)
10357 052104 012746 040500      MOV    #RXBRRT,-(SP)
10357 052110 013746 002266      MOV    RXVECA,-(SP)
10357 052114 012746 000003      MOV    #3,-(SP)
10357 052120 104437      TRAP    C#SVEC
10357 052122 062706 000010      ADD     #10,SP
10358
10359      ;+
10360      ; Set up for the loop which tests the interrupt BR levels.
10361      ;-
10361 052126 012705 000340      MOV    #340,R5      ;SET UP THE PRIORITY LEVEL TO 7.
10362 052132 005003      CLR     R3      ;CLEAR THE RX PRIORITY STORE AND FLAGS.
10363 052134 005002      CLR     R2      ;CLEAR THE TX PRIORITY STORE AND FLAGS.
10364
10365      ;+
10366      ; Enable TX and RX interrupts.
10367      ; Processor priority should be at 7 disabling the interrupts.
10368      ;-
10368 052136 004737 035600      JSR     PC,RXIE1      ;ENABLE RECEIVER INTERRUPTS.
10369 052142 004737 036574      JSR     PC,TXIE1      ;ENABLE TRANSMITTER INTERRUPTS.
10370
10371      ;+
10372      ; Loop. lowering the processor priority until the DUT interrupts on RX and TX.
10373      ;-
10373 052146 005037 002416      6$:    CLR    TXINTC      ;CLEAR THE TX INTERRUPT COUNTER.
10374 052152 005037 002420      CLR    TXINTF      ;CLEAR THE TX INTERRUPT FLAGS.
10375 052156 005037 002412      CLR    RXINTC      ;CLEAR THE RX INTERRUPT COUNTER.
10376 052162 005037 002414      CLR    RXINTF      ;CLEAR THE RX INTERRUPT FLAGS.
10377 052166 106405      MTPS    R5      ;SET PROCESSOR PRIORITY TO THE SELECTED VALUE.
10378 052170 012704 000001      MOV    #1,R4      ;PASS 1 MS COUNT TO THE DELAY ROUTINE.
10379 052174 004737 030370      JSR     PC,DELAY      ;DELAY 1 MS TO ALLOW INTERRUPTS TO OCCUR.
10380
10381      ;+
10382      ; Determine if any RX DUT interrupts occurred.
10383      ; Log the processor priority for the RX interrupt if first RX int.
10384      ;-
10384 052200 005737 002412      TST     RXINTC      ;CHECK THE RECEIVE INTERRUPT COUNTER.
10385 052204 001412      BEQ     #$      ;SKIP THE PRIORITY LOG IF NO RX INT OCCURRED.
10386
10387      ;+
10388      ; If this is the first RX interrupt, log the priority.
10389      ;-
10389 052206 005703      TST     R3      ;CHECK THE RX PRIORITY STORE AND FLAGS.
10390 052210 001010      BNE     #$      ;GOTO TEST FOR TX INTS IF NOT THE FIRST RX INT.
10391 052212 010503      MOV    R5,R3      ;LOG THE PRESENT PRIORITY IN THE RX PRIO STORE.
10392 052214 052703 100000      BIS    #BIT15,R3      ;SET THE RX INT HAS OCCURRED FLAG.
10393 052220 013700 002414      MOV    RXINTF,R0      ;GET THE RX INTERRUPT ROUTINE FLAGS.

```

HARDWARE TEST

- BRLEVA -

```

10394 052224 042700 137777      BIC    #137777,R0      ;CLEAR ALL BUT THE TX INT ERROR FLAG.
10395 052230 050003              BIS    R0,R3          ;IF TX INT ERROR, SET BIT 14 OF THE PRIO FLAGS.
10396                               ;+
10397                               ; Determine if any TX DUT interrupts have occurred.
10398                               ; Log the present processor priority if this is the first TX interrupt.
10399                               ;-
10400 052232 005737 002416      8$:   TST    TXINTC          ;CHECK THE TRANSMIT INTERRUPT COUNTER.
10401 052236 001405              BEQ    10$             ;SKIP THE PRIORITY LOG IF NO TX INT OCCURRED.
10402                               ;+
10403                               ; If this is the first TX interrupt, log the priority.
10404                               ;-
10405 052240 005702              TST    R2              ;CHECK THE TX PRIORITY STORE AND FLAGS.
10406 052242 100403              BMI    10$             ;SKIP THE LOGGING IF NOT FIRST TX INTERRUPT.
10407 052244 010502              MOV    R5,R2          ;LOG THE PRESENT PRIORITY IN THE TX PRIO STORE.
10408 052246 052702 100000      BIS    #BIT15,R2       ;SET THE TX INT HAS OCCURRED FLAG.
10409                               ;+
10410                               ; Select next processor priority.
10411                               ; Test for both RX and TX interrupts having occurred, loop if not.
10412                               ;-
10413 052252 162705 000040      10$:  SUB    #40,R5          ;DECREMENT PRIORITY LEVEL BY ONE.
10414 052256 002402              BLT    12$             ;GOTO CHECK FOR ERRORS IF BELOW PRIORITY ZERO.
10415 052260 030203              BIT    R2,R3          ;AND PRIO FLAGS TOGETHER, ALTER NONE OF THEM.
10416 052262 100331              BPL    6$              ;LOOP IF RX AND TX INTS HAVEN'T BOTH OCCURRED.
10417                               ;+
10418                               ; Disable interrupts and clear interrupt vectors.
10419                               ;-
10420 052264 106427 000340      12$:  MTPS    #PRI07          ;DISABLE ALL INTERRUPTS.
10421 052270              CLRVEC  RXVECA              ;RETURN RX INT VECTOR TO UNUSED POOL.
10422 052276              CLRVEC  TXVECA              ;RETURN TX INT VECTOR TO UNUSED POOL.
10423 052276 013700 002266      MOV    RXVECA,R0      ;
10424 052274 104436              TRAP    C$CVEC        ;
10425 052276 013700 002270      MOV    TXVECA,R0      ;
10426 052302 104436              TRAP    C$CVEC        ;
10427                               ;+
10428                               ; Verify that RX and TX interrupts occurred,
10429                               ; at the proper BR level, and
10430                               ; in the proper order.
10431                               ; Determine if TX interrupt occurred.
10432                               ;-
10433 052304 005702              TST    R2              ;DETERMINE WHETHER TX INT OCCURRED OR NOT.
10434 052306 100414              BMI    16$             ;SKIP THESE ERRORS IF TX INT OCCURRED.
10435                               ;+
10436                               ; Determine reason that no TX int occurred.
10437                               ;-
10438 052310 012701 017270      MOV    #EM2610,R1        ;SELECT "NO TX INT FROM TX.ACTION" MESSAGE.
10439 052314 005777 127760      TST    #CSRA          ;CHECK THE TX.ACTION BIT OF THE DUT CSR.
10440 052320 100402              BMI    14$             ;SKIP TX.ACTION CLR MSG SELECTION IF IT IS SET.
10441 052322 012701 017362      MOV    #EM2611,R1        ;SELECT "TX.ACTION CLEAR AFTER CHARS SENT" MSG.
10442 052326              ;REPORT "INTERRUPT BR LEVEL TEST ERROR:"
10443 052326 104455      14$:  ERDF    3003,EM3001,ER0503;      >>>>> ERROR #3003 <<<<<.
10444 052330 005673              TRAP    C$ERDF        ;
10445 052332 017445              .WORD    3003         ;
10446 052334 024644              .WORD    EM3001        ;
10447 052336 000423              .WORD    ER0503        ;
10448              BR    18$              ;SKIP THE BR LEVEL CHECK, NO TX INT OCCURRED.
10449                               ;+
10450                               ; Verify that the TX interrupt was at the proper BR level.

```


HARDWARE TEST

- BRLEVA -

```

10443
10444 052340 010204      16$:  MOV    R2,R4      ;CALCULATE THE BR LEVEL
10445 052342 042704 177400 BIC    #177400,R4    ; THAT THE TRANSMIT
10446 052346 006204      ASR    R4      ; INTERRUPT WAS
10447 052350 006204      ASR    R4      ; REQUESTED AT, WHICH
10448 052352 006204      ASR    R4      ; IS ONE GREATER THAN
10449 052354 006204      ASR    R4      ; THE PROCESSOR PRIORITY
10450 052356 006204      ASR    R4      ; LEVEL AT WHICH THE
10451 052360 005204      INC    R4      ; TRANSMIT INTERRUPT OCCURRED.
10452 052362 113705 002275 MOVB   BRLEVL,R5    ;GET THE EXPECTED INTERRUPT BR LEVEL.
10453 052366 120405      CMPB   R4,R5    ;COMARE THE INTERRUPT BR LEVEL WITH EXPECTED.
10454 052370 001406      BEQ    18$      ;SKIP THE ERROR IF BR LEVEL IS CORRECT.
10455
;REPORT "TX INTERRUPT GENERATED AT WRONG BR LEVEL: ..."
10456 052372 012701 017572 MOV    #EM3003,R1    ;SELECT THE ERROR MESSAGE FOR THE ERROR CALL.
10457 052376      ERRDF   3004,EM3001,ER3001; >>>> ERROR #3004 <<<<.
                                TRAP    C#ERDF
                                .WORD   3004
                                .WORD   EM3001
                                .WORD   ER3001

10458
;+
; Determine if RX interrupt occurred.
10459
;-
10460
18$:  TST    R3      ;CHECK THE RX INT OCCURRED FLAG.
      BMI    22$      ;SKIP THESE ERRORS IF RX INT OCCURRED.
;+
; Determine reason that no RX int occurred.
10461 052406 005703      18$:
10462 052410 100415      BMI    22$
10463
;-
10464
;+
10465
; Determine reason that no RX int occurred.
10466 052412 012701 016447      MOV    #EM2602,R1    ;SELECT "NO RX INT FROM TX.ACTION" MSG.
10467 052416 032777 000200 127654 BIT    #BIT7,0CSRA    ;CHECK THE RX.DATA.AVAIL BIT OF THE DUT CSR.
10468 052424 001002      BNE    20$      ;SKIP RX.DATA.AVAIL CLR MSG IF BIT IS SET.
10469 052426 012701 017476      MOV    #EM3002,R1    ;SELECT "NO RX.DATA.AVAIL AFTER RESET" MSG.
10470
;REPORT "INTERRUPT BR LEVEL TEST ERROR:"
10471 052432      20$:  ERRDF   3005,EM3001,ER0503; >>>> ERROR #3005 <<<<.
                                TRAP    C#ERDF
                                .WORD   3005
                                .WORD   EM3001
                                .WORD   ER0503
10472 052442 000423      BR      24$      ;SKIP THE BR CHECK IF NO RX INT OCCURRED.
10473
;+
; Verify that the RX interrupt was at the proper BR level.
10474
;-
10475
22$:  MOV    R3,R4      ;CALCULATE THE BR LEVEL
      BIC    #177400,R4    ; THAT THE RECEIVE
10477 052446 042704 177400 ASR    R4      ; INTERRUPT WAS
10478 052452 006204      ASR    R4      ; REQUESTED AT, WHICH
10479 052454 006204      ASR    R4      ; IS ONE GREATER THAN
10480 052456 006204      ASR    R4      ; THE PROCESSOR PRIORITY
10481 052460 006204      ASR    R4      ; LEVEL AT WHICH THE
10482 052462 006204      ASR    R4      ; RECEIVE INTERRUPT OCCURRED.
10483 052464 005204      INC    R4
10484 052466 113705 002275 MOVB   BRLEVL,R5    ;GET THE EXPECTED INTERRUPT BR LEVEL.
10485 052472 120405      CMPB   R4,R5    ;COMARE THE INTERRUPT BR LEVEL WITH EXPECTED.
10486 052474 001406      BEQ    24$      ;SKIP THE ERROR IF BR LEVEL IS CORRECT.
10487
;REPORT "RX INTERRUPT GENERATED AT WRONG BR LEVEL: ..."
10488 052476 012701 017646 MOV    #EM3004,R1    ;SELECT ERROR MESSAGE FOR THE ERROR CALL.
10489 052502      ERRDF   3006,EM3001,ER3001; >>>> ERROR #3006 <<<<.
                                TRAP    C#ERDF
                                .WORD   3006
                                .WORD   3006
                                .WORD   3006

```

HARDWARE TEST

- BRLEVA -

```

052506 017445
052510 025114
10490
10491
10492
10493
10494 052512 032703 040000
10495 052516 001406
10496
10497 052520 012701 017722
10498 052524
      052524 104455
      052526 005677
      052530 017445
      052532 024644
10499
10500
10501
10502 052534 004737 036542
10503 052540 004737 035546
10504 052544 005037 002362
10505 052550 106427 000240
10506 052554
      052554
      052554 104401
10507

;+
; 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$:  JSR    PC,TXIE0      ;CLEAR TRANSMITTER INTERRUPTS.
      JSR    PC,RXIE0      ;CLEAR RECEIVER INTERRUPTS.
60$:  CLR    CTRLCF        ;INDICATE THAT WE ARE NOT WITHIN A TEST.
      MTPS   #PRI05        ;DISABLE ALL BUT LTC INTERRUPTS.
      ENDTST

                                L10063:
                                TRAP    C#ETST

```


HARDWARE TEST

- DIABMP -

```

10509 .SBTTL HARDWARE TEST - DIABMP -
10510 ;* *****
10511 ;* - Diagnostic field (BMP) Test -
10512 ;* This test verifies that a request to the DUT to report BMP status
10513 ;* codes is complied with, within the specified time.
10514 ;* All active lines are tested.
10515 ;* *****
10516 ;-- *****
10517 052556 BGNTST
10518 052556 T25::
10519 052556 012737 000031 002364 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
10520 052564 012737 177777 002362 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (31)
10521 052572 012777 000001 005464 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
10522 052600 012737 006035 005466 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
10523 052606 012737 020014 005470 MOV #3101,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
10524 052614 012737 026440 005472 MOV #EM3101,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRTABL.
10525 MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
10526 ;*
10527 ; Reset the DUT to a known state, remove the status codes from the fifo.
10528 ; Clear TX and RX interrupt enable bits in the CSR.
10529 ; This subroutine reports error >>>> 3101 <<<<.
10530 052622 004737 030172 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
10531 052626 103077 BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.
10532 ;*
10533 ; Test all active lines individually.
10534 ; Write the request code to the diagnostic field in the LPR register.
10535 ; Verify that a BMP code is returned within the correct time.
10536 ;--
10537 052630 013705 002272 MOV ACTLNS,R5 ;GET THE ACTIVE LINE BIT MAP.
10538 052634 005004 CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
10539 052636 013703 002300 MOV CSRA,R3 ;GET THE ADDRESS OF THE DUT'S CSR.
10540 052642 000241 2$: CLC ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
10541 052644 006005 ROR R5 ;SHIFT THE BIT MAP INTO THE CARRY BIT.
10542 052646 103064 BCC 8$ ;DO NOT TEST THE LINE IF IT IS INACTIVE.
10543 ;*
10544 ; Select the line under test.
10545 ; Write the BMP request code to the DIAG field in the LPR register.
10546 ;--
10547 052650 012737 006036 005466 MOV #3102,ERRNBR ;SET THE ERROR NUMBER TO 3102.
10548 052656 010413 MOV R4,(R3) ;SELECT THE LINE CURRENTLY UNDER TEST.
10549 052660 052777 000002 127416 BIS #2,@LPRA ;WRITE THE BMP REQUEST CODE TO THE LPR.
10550 ;*
10551 ; Wait for BMP request code to be cleared, report error if time-out
10552 ; occurs.
10553 ;--
10554 052666 012701 010764 MOV #10764,R1 ;TEST BIT 1, TIMEOUT OF 500 MILLI SECS.
10555 052672 013702 002304 MOV LPRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.
10556 052676 004737 037664 JSR PC,WAIBIC ;WAIT FOR REQUEST CODE TO CLEAR.
10557 052702 103042 BCC 6$ ;GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.
10558 ;*
10559 ; Wait for BMP code to appear in the fifo, report error if time-out
10560 ; occurs.
10561 ;--
10562 052704 005237 005466 INC ERRNBR ;SET ERROR NUMBER TO 3103.
10563 052710 012701 070012 MOV #70012,R1 ;TEST BIT 7, TIMEOUT OF 10 MILLI SECS.
10564 052714 013702 002300 MOV CSRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.

```

HARDWARE TEST

- DIABMP -

```

10565 052720 004737 037740      JSR      PC,WAIBIS      ;WAIT FOR RX_DATA_AVAILABLE TO SET.
10566 052724 103031      BCC      6$      ;GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.
10567                                ;*
10568                                ; Read the BMP code (if it is there) from the RBUF register.
10569                                ; Determine if it is a valid BMP code.
10570                                ; Verify the BMP code was received from the correct channel.
10571                                ; If the BMP code does not indicate DUT running ok, then save it on
10572                                ; the queue to be reported in a later test.
10573                                ;-
10574 052726 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 3104.
10575 052732 017702 127344      MOV      @RBUFA,R2      ;GET THE BMP CODE FROM THE FIFO.
10576 052736 100024      BPL      6$      ;GO REPORT ERROR IF NO BMP CODE FOUND.
10577 052740 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 3105.
10578 052744 012700 170301      MOV      @170301,R0      ;SET-UP A BMP CODE MASK.
10579 052750 040200      BIC      R2,R0      ;TRY TO CLEAR THE BMP MASK.
10580 052752 001016      BNE      6$      ;GO REPORT ERROR IF IT IS NOT A VALID BMP CODE.
10581 052754 005237 005466      INC      ERRNBR      ;SET THE ERROR NUMBER TO 3106.
10582 052760 010200      MOV      R2,R0      ;COPY THE BMP CODE.
10583 052762 000300      SWAB      R0      ;PUT THE LINE NUMBER IN THE LOW BYTE.
10584 052764 042700 177760      BIC      @177760,R0      ;CLEAR THE UNWANTED BITS.
10585 052770 120400      CMPB      R4,R0      ;DID THE BMP CODE COME FROM THE CORRECT LINE?.
10586 052772 001006      BNE      6$      ;NO; GO REPORT ERROR.
10587 052774 120227 000305      CMPB      R2,#305      ;IS THE BMP CODE A "GOOD ONE"?.
10588 053000 001407      BEQ      8$      ;YES; SKIP SAVING THE BMP CODE ON THE QUEUE.
10589 053002 004737 035624      JSR      PC,SAVBMP      ;SAVE THE BMP CODE ON THE QUEUE.
10590 053006 000404      BR      8$      ;GO SEE IF THERE ARE ANY MORE LINE TO TEST.
10591
10592 053010 010401      6$:      MOV      R4,R1      ;PASS THE LINE NUMBER TO BE REPORTED.
10593 053012 012702 020050      MOV      @EM3102,R2      ;PASS THE ERROR MESSAGE TO BE REPORTED.
10594                                ;"BMP REQUEST BIT BAD ON LINE:"
10595 053016      ERROR      ;          >>>> ERROR <<<<<
                                ;          TRAP      C$ERROR
10596                                ;*
10597                                ; Verify all active lines have been tested.
10598                                ;-
10599 053020 005204      8$:      INC      R4      ;INCREMENT THE LINE NUMBER COUNTER.
10600 053022 005705      TST      R5      ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
10601 053024 001306      BNE      2$      ;YES; BRANCH TO TEST THE NEXT LINE.
10602 053026 005037 002362      60$:    CLR      CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
10603 053032      ENDTST
                                ;
                                ;L10064:
                                ;          TRAP      C$ETST
053032 104401

```

HARDWARE TEST

- DIABMP -

```

10605
10606
10607
10608
10609
10610
10611
10612
10613
10614 053034
      053034
10615
10616 053034 000032
10617 053042 012737 000032 002364
10618 053050 012737 177777 002362
10619 053056 012737 000001 005464
10620 053064 012737 007641 005466
10621 053072 012737 020111 005470
10622
10623
10624
10625
10626
10627 053100 004737 030172
10628 053104 103141
10629
10630 053106 004737 031342
10631
10632
10633
10634
10635
10636
10637 053112 013705 002272
10638 053116 012700 000204
10639 053122 004737 040250
10640 053126 012700 177670
10641 053132 004737 040324
10642 053136 004737 036446
10643
10644
10645
10646 053142 013705 002272
10647 053146 005001
10648 053150 012737 007642 005466 21:
10649 053156 006241
10650 053160 006005
10651 053162 103106
10652 053164 004737 032722
10653 053170 103107
10654
10655
10656
10657
10658
10659
10660

.SBTTL  HARDWARE TEST          - DMASTA -
;+ *****
;*          - DMA Start Bit Test -
;*      This test verifies that the DMA_START bit in the DUT's Line control
;*      registers will initiate DMA transmission on the selected line.
;*      This test is performed in internal loopback, on all active lines.
;*
;-- *****
      BGNTST
;+
;+ T26::
;+ INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      TNUM == TNUM + 1
;+ SET UP THE TEST NUMBER. (40)
      MOV     #TNUM,TSTNUM
;+ INDICATE THAT WE ARE IN A TEST.
      MOV     #-1,CTRLCF
;+ SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV     #1,ERRTYP
;+ SET THE FIRST ERROR NUMBER IN ERROR TABLE.
      MOV     #4001,ERRNBR
;+ SET ERROR MESSAGE ADDRESS IN ERRMBL.
      MOV     #EM4001,ERRMSG
;+ SELECT THE CORRECT ERROR REPORTING ROUTINE.
      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 >>>> 4001 <<<<.
;+
      JSR     PC,CLNRST      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCC     501           ;RESET FAILURE?, ABORT THIS TEST.
;+
      JSR     PC,INDATP      ;INITIALISE THE 256 BYTE DATA PATTERN.
;+
;+ Set internal loopback,enable receiver functions on all active lines.
;+ Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
;+ 2 stop bits.
;+ Enable transmitters on all active lines.
;+
      MOV     ACTLNS,R5      ;PASS THE ACTIVE LINE BIT MAP.
      MOV     #204,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.
      JSR     PC,TXENBL      ;ENABLE TRANSMITTERS ON ALL LINES.
;+
;+ Set-up outer loop to test the DMA_START bit on all active lines.
;+
      MOV     ACTLNS,R5      ;GET THE ACTIVE LINE BIT MAP.
      CLR     R1             ;CLEAR THE LINE NUMBER COUNTER.
      MOV     #4002,ERRNBR   ;SET THE ERROR NUMBER TO 4002.
      CLC                     ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
      ROR     R5             ;SHIFT THE BIT MAP INTO THE CARRY BIT.
      BCC     141           ;DO NOT TEST THE LINE IF IT IS INACTIVE.
      JSR     PC,PUFIFO      ;PURGE THE FIFO.
      BCC     501           ;GO REPORT ERROR IF FIFO WILL NOT CLEAR.
;+
;+ Perform DMA_START bit testing on each line individually.
;+ Test each DMA_START bit before TX'ing data pattern, report error if set
;+ Set DMA_START bit on LUT, verify it is set, report error if clear.
;+ Wait for DMA to complete.
;+ Verify DMA_START bit is clear, report error if set.
;+ Verify correct number of chars were received, report error if < expected.

```

HARDWARE TEST

DMASTA -

```

10661
10662 053172 005237 005466      ; INC ERRNBR ;SET ERROR NUMBER TO 4003.
10663 053176 012702 004012      ; MOV #0UFBAS,R2 ;PASS THE START OF THE DATA PATTERN TO TX.
10664 053202 012703 000144      ; MOV #100.,R3 ;PASS THE LENGTH OF THE DATA PATTERN.
10665 053206 004737 030624      ; JSR PC,D0DMA ;TRANSMIT THE DATA PATTERN.
10666 053212 103067              ; BCC 12$ ;GO REPORT ERROR IF DMA_START BIT SET.
10667
10668      ; Test the state of the DMA_START bit on the line under test.
10669      ; Report error if DMA_START bit is clear.
10670
10671 053214 005237 005466      ; INC ERRNBR ;INCREMENT ERROR NUMBER TO 4004.
10672 053220 010177 127054      ; MOV R1,0CSRA ;SELECT THE LINE CURRENTLY UNDER TEST.
10673 053224 105777 127064      ; TSTB @TXAD2A ;TEST THE STATE OF THE DMA_START BIT.
10674 053230 100060              ; BPL 12$ ;GO REPORT ERROR IF BIT IS CLEAR.
10675
10676      ; Wait for DMA transmission to complete.
10677
10678 053232 005237 005466      4$: INC ERRNBR ;INCREMENT ERROR NUMBER TO 4005.
10679 053236 010103              ; MOV R1,R3 ;SAVE THE LINE NUMBER.
10680 053240 012701 170226      ; MOV #170226,R1 ;TEST BIT 15, TIMEOUT OF 150 MILLI SECS.
10681 053244 013702 002300      ; MOV CSRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.
10682 053250 004737 037740      ; JSR PC,WAIBIS ;WAIT FOR DMA TO COMPLETE.
10683 053254 103045              ; BCC 10$ ;GO REPORT ERROR IF TIMEOUT OCCURRED.
10684 053256 012704 000005      ; MOV #5,R4 ;PASS DELAY OF 5 MILLI SECS.
10685 053262 004737 030370      ; JSR PC,DELAY ;WAIT FOR CHAR TO BE RECEIVED AND PROCESSED.
10686 053266 010301              ; MOV R3,R1 ;RESTORE THE CURRENT LINE NUMBER.
10687
10688      ; Test the state of the DMA_START bit on the line under test.
10689      ; Report error if DMA_START bit is set.
10690
10691 053270 005237 005466      ; INC ERRNBR ;INCREMENT ERROR NUMBER TO 4006.
10692 053274 010177 127000      ; MOV R1,0CSRA ;SELECT THE LINE CURRENTLY UNDER TEST.
10693 053300 105777 127010      ; TSTB @TXAD2A ;TEST THE STATE OF THE DMA_START BIT.
10694 053304 100432              ; BMI 12$ ;GO REPORT ERROR IF BIT IS STILL SET.
10695
10696      ; Verify the number of chars received = number of chars expected.
10697      ; Report error if count is incorrect.
10698      ; If more than 128 BMP codes are found then report error and exit test.
10699
10700 053306 005003              ; CLR R3 ;CLEAR THE READ COUNTER.
10701 053310 012704 000200      ; MOV #128.,R4 ;SET UP MAX BMP CODE READ COUNT.
10702 053314 012737 007647      005466 6$: MOV #4007.,ERRNBR ;SET ERROR NUMBER TO 4007.
10703 053322 017702 126754      ; MOV @RBUFA,R2 ;READ THE CHARACTER FROM THE FIFO.
10704 053326 100021              ; BPL 12$ ;GO REPORT ERROR IF FIFO EMPTY TOO SOON.
10705 053330 012700 170301      ; MOV #170301,R0 ;SET-UP BIT MASK OF A BMP CODE.
10706 053334 040200              ; BIC R2,R0 ;TRY TO CLEAR THE BMP CODE MASK.
10707 053336 001007              ; BNE 8$ ;BRANCH IF NOT A BMP CODE.
10708 053340 005237 005466      ; INC ERRNBR ;INCREMENT ERROR NUMBER TO 4008.
10709 053344 004737 035624      ; JSR PC,SAVBMP ;SAVE THE BMP CODE ON THE QUEUE.
10710 053350 005304              ; DEC R4 ;DECREMENT MAX BMP CODE READ COUNT.
10711 053352 001416              ; BEQ 50$ ;GO REPORT ERROR IF TOO MANY BMP CODES FOUND.
10712 053354 000757              ; BR 6$ ;DO NOT COUNT THE BMP CODE AS A VALID CHAR.
10713 053356 005203      8$: INC R3 ;COUNT THIS CHARACTER.
10714 053360 020327 000144      ; CMP R3,#100. ;HAVE WE RECIEVED 100 CHARACTERS?.
10715 053364 002753              ; BLT 6$ ;LOOP UNTIL 100 (NON-BMP) CHARS ARE READ.
10716 053366 000404              ; BR 14$ ;SKIP AROUND THE ERROR REPORT.
10717

```

HARDWARE TEST

- DMASTA -

```

10718
10719
10720
10721 053370 010301      10$:  MOV    R3,R1      ;RESTORE THE CURRENT LINE NUMBER.
10722
10723 053372 012702 020134 12$:  MOV    @EM4002,R2    ;PASS THE ERROR MESSAGE TO BE REPORTED.
10724
10725 053376          ERROR          ; "DMA_START BIT BAD ON LINE nn".
10726 053376 104460          ;          >>>>> ERROR <<<<<.
10727 053400 005201      14$:  INC     R1          ;INCREMENT THE LINE NUMBER COUNTER.
10728 053402 005705          TST     R5          ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
10729 053404 001261          BNE     2$          ;YES; BRANCH TO TEST THE NEXT LINE.
10730 053406 000402          BR      60$         ;NO; EXIT THIS TEST.
10731
10732 053410 004737 036106 50$:  JSR     PC,TSABRT    ;REPORT TEST ABORTED. NON-TEST RELATED ERROR.
10733 053414 005037 002362 60$:  CLR     CTRLCF    ;INDICATE THAT WE ARE NOT WITHIN A TEST.
10734
10735 053420          ENDTST
      053420
      053420 104401
                                L10065:
                                TRAP    C$ETST

```

HARDWARE TEST

DMABRT -

```

10737 .SBTTL  HARDWARE TEST          - DMABRT -
10738 ;* *****
10739 ;*          - DMA Abort/Restart Test -
10740 ;*      This test verifies that each DMA_ABORT bit will correctly halt
10741 ;*      a DMA transmission, and return a TX_ACTION.
10742 ;*      It will also verify that the aborted DMA transmission can be resumed,
10743 ;*      and that a TX_ACTION is returned upon completion.
10744 ;*      This test is performed in internal loopback, on all active lines.
10745 ;*
10746 ;* *****
10747 053422 053422  BGNTST
10748
10749 053422 012737 000033 002364      T27::
10750 053430 012737 177777 002362      MOV    #TNUM,TSTNUM      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
10751 053436 012737 000001 005464      MOV    #-1,CTRLCF      ;SET UP THE TEST NUMBER. (41)
10752 053444 012737 010005 005466      MOV    #1,ERRTYP      ;INDICATE THAT WE ARE IN A TEST.
10753 053452 012737 020170 005470      MOV    #4101,ERRNBR      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
10754 053460 012737 026440 005472      MOV    #4101,ERRMSG      ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
10755 ;*                               ;SET ERROR MESSAGE ADDRESS IN ERRBL.
10756 ;*                               ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
10757 ;*
10758 ;*      Reset the DUT to a known state, remove the status codes from the fifo.
10759 ;*      Clear TX and RX interrupt enable bits in the CSR.
10760 ;*      This subroutine reports error >>>> 4101 <<<<.
10761 053466 004737 030172      JSR    PC,CLNRST      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
10762 053472 103160      BCC    60$      ;RESET FAILURE?, ABORT THIS TEST.
10763 053474 004737 031342      JSR    PC,INDATP      ;INITIALISE 256 BYTE DATA PATTERN.
10764 ;*
10765 ;*      Set internal loopback,enable receiver functions on all active lines.
10766 ;*      Set LPR on all lines to 38.4k baud, 8 bits per character, odd parity,
10767 ;*      2 stop bits.
10768 ;*      Enable transmitters on all active lines.
10769 ;*
10770 053500 013705 002272      MOV    ACTLNS,R5      ;PASS THE ACTIVE LINE BIT MAP.
10771 053504 012700 000204      MOV    #204,R0      ;PASS THE LNCTRL CONTENTS.
10772 053510 004737 040250      JSR    PC,WTWLNLC      ;INITIALISE THE LNCTRL REGISTERS.
10773 053514 012700 177670      MOV    #177670,R0      ;PASS THE LPR CONTENTS.
10774 053520 004737 040324      JSR    PC,WTWLPR      ;INITIALISE THE LPR REGISTERS ON ALL LINES.
10775 053524 004737 036446      JSR    PC,TXENBL      ;ENABLE TRANSMITTERS ON ALL LINES.
10776 ;*
10777 ;*      Perform DMA_ABORT bit testing on each individual (active) line.
10778 ;*
10779 053530 013705 002272      MOV    ACTLNS,R5      ;GET THE ACTIVE LINE BIT MAP.
10780 053534 005001      CLR    R1      ;CLEAR THE LINE NUMBER COUNTER.
10781 053536 012737 010006 005466 2$:  MOV    #4102,ERRNBR      ;SET THE ERROR NUMBER TO 4102.
10782 053544 000241      CLC      ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
10783 053546 006005      ROR    R5      ;SHIFT THE BIT MAP INTO THE CARRY BIT.
10784 053550 103123      BCC    10$      ;DO NOT TEST THE LINE IF IT IS INACTIVE.
10785 053552 004737 032722      JSR    PC,PUFIFO      ;PURGE THE FIFO.
10786 053556 103124      BCC    50$      ;GO REPORT ERROR IF FIFO WILL NOT CLEAR.
10787 ;*
10788 ;*      Check the DMA_ABORT bit before enabling DMA, report error if set.
10789 ;*
10790 053560 005237 005466      INC    ERRNBR      ;INCREMENT ERROR NUMBER TO 4103.
10791 053564 010177 126510      MOV    R1,@CSRA      ;SELECT THE LINE CURRENTLY UNDER TEST.
10792 053570 032777 000001 126512      BIT    @BIT0,@LNCTRA      ;TEST THE STATE OF THE DMA ABORT BIT.

```


HARDWARE TEST

- DMABRT -

```

10793 053576 001105
10794
10795
10796
10797
10798 053600 005237 005466
10799 053604 012702 004012
10800 053610 012703 000400
10801 053614 004737 030624
10802 053620 103103
10803
10804
10805
10806 053622 010177 126452
10807 053626 012704 000062
10808 053632 004737 030370
10809 053636 052777 000001 126444
10810
10811
10812
10813 053644 005237 005466
10814 053650 010103
10815 053652 012701 170012
10816 053656 013702 002300
10817 053662 004737 037740
10818 053666 103050
10819 053670 010301
10820
10821
10822
10823 053672 005237 005466
10824 053676 012702 020247
10825
10826 053702 010177 126372
10827 053706 105777 126402
10828 053712 100441
10829
10830
10831
10832 053714 042777 000001 126366
10833 053722 052777 000200 126364
10834
10835
10836
10837 053730 005237 005466
10838 053734 010103
10839 053736 012701 170536
10840 053742 013702 002300
10841 053746 004737 037740
10842 053752 103016
10843 053754 012704 000002
10844 053760 004737 030370
10845 053764 010301
10846
10847
10848
10849

```

```

BNE      6$              ;GO REPORT ERROR IF BIT IS SET.
;+
; Enable DMA TX on selected line, wait for DMA to TX approx 1/4 of data.
; Abort the DMA transmission. Wait for TX_ACTION to be returned.
;-
      INC      ERRNBR      ;SET ERROR NUMBER TO 4104.
      MOV      @BUFBAS,R2  ;PASS THE START OF THE DATA PATTERN TO TX.
      MOV      @256.,R3    ;PASS THE LENGTH OF THE DATA PATTERN.
      JSR      PC,DODMA    ;TRANSMIT THE DATA PATTERN.
      BCC      50$        ;GO REPORT ERROR IF THERE ARE TX PROBLEMS.
;+
; Wait for DMA to transmit 1/4 of the data before aborting.
;-
      MOV      R1,@CSRA    ;SELECT THE LINE CURRENTLY UNDER TEST.
      MOV      @50.,R4     ;PASS THE DELAY TIME OF 40 MILLI SECONDS.
      JSR      PC,DELAY    ;WAIT FOR APPROX 1/4 OF DATA TO BE TX'D.
      BIS      @BIT0,@LNCTRA ;ABORT THE DMA TRANSMISSION.
;+
; Wait for TX_ACTION to be returned, report error if time-out occurs.
;-
      INC      ERRNBR      ;INCREMENT ERROR NUMBER TO 4105.
      MOV      R1,R3       ;SAVE THE LINE NUMBER.
      MOV      @170012,R1  ;TEST BIT 15, TIMEOUT OF 10 MILLI SECS.
      MOV      CSRA,R2     ;PASS THE ADDRESS OF THE REGISTER TO TEST.
      JSR      PC,WAIBIS   ;WAIT FOR DMA TO COMPLETE.
      BCC      4$          ;GO REPORT ERROR IF TIMEOUT OCCURRED.
      MOV      R3,R1       ;RESTORE THE CURRENT LINE NUMBER.
;+
; Verify DMA_START bit clear, report error if set.
;-
      INC      ERRNBR      ;INCREMENT ERROR NUMBER TO 4106.
      MOV      @EM4103,R2  ;SELECT MESSAGE TO BE REPORTED.
; "DMA_START BIT FOUND SET AFTER DMA ABORTED".
      MOV      R1,@CSRA    ;SELECT THE LINE CURRENTLY UNDER TEST.
      TSTB     @TXAD2A     ;TEST THE STATE OF THE DMA_START BIT.
      BMI      8$          ;GO REPORT ERROR IF IT IS SET.
;+
; Resume DMA transmission by clearing DMA_ABORT and setting DMA_START.
;-
      BIC      @BIT0,@LNCTRA ;CLEAR THE DMA_ABORT BIT.
      BIS      @BIT7,@TXAD2A ;SET THE DMA_START BIT.
;+
; Wait for DMA transmission to complete.
;-
      INC      ERRNBR      ;INCREMENT ERROR NUMBER TO 4107.
      MOV      R1,R3       ;SAVE THE LINE NUMBER.
      MOV      @170536,R1  ;TEST BIT 15, TIMEOUT OF 350 MILLI SECS.
      MOV      CSRA,R2     ;PASS THE ADDRESS OF THE REGISTER TO TEST.
      JSR      PC,WAIBIS   ;WAIT FOR DMA TO COMPLETE.
      BCC      4$          ;GO REPORT ERROR IF TIMEOUT OCCURRED.
      MOV      @2,R4       ;PASS TIME-OUT OF 2 MILLI SECS.
      JSR      PC,DELAY    ;WAIT FOR CHAR TO BE RECEIVED AND PROCESSED.
      MOV      R3,R1       ;RESTORE THE CURRENT LINE NUMBER.
;+
; Test the state of the DMA_ABORT bit on the line under test.
; Report error if DMA_ABORT bit is set.
;-

```

HARDWARE TEST

- DMABRT -

```

10850 053766 005237 005466      INC  ERRNBR      ;INCREMENT ERROR NUMBER TO 4108.
10851 053772 010177 126302      MOV  R1,0CSRA    ;SELECT THE LINE CURRENTLY UNDER TEST.
10852 053776 032777 000001 126304  BIT  @BIT0,@LNCTRA ;TEST THE STATE OF THE DMA_ABORT BIT.
10853 054004 001002              BNE  6$          ;GO REPORT ERROR IF BIT IS SET.
10854 054006 000404              BR   10$          ;BRANCH TO CHECK FOR ANY MORE LINES TO TEST.
10855
10856                          ;+
10857                          ; Report error, skip further testing on this line.
10858 054010 010301              4$:  MOV  R3,R1          ;RESTORE THE CURRENT LINE NUMBER.
10859
10860 054012 012702 020213      6$:  MOV  @EM4102,R2      ;PASS THE ERROR MESSAGE TO BE REPORTED.
10861
10862 054016              8$:  ERROR                      ; "DMA_ABORT BIT BAD ON LINE nn".
10863 054016 104460              ;                      >>>>> ERROR <<<<<.
10864                          ;+                                TRAP  C$ERROR
10865                          ; Verify all active lines have been tested.
10866 054020 005201      10$:  INC  R1          ;INCREMENT THE LINE NUMBER COUNTER.
10867 054022 005705      TST  R5          ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
10868 054024 001244      BNE  2$          ;YES; BRANCH TO TEST THE NEXT LINE.
10869 054026 000402      BK   60$        ;NO; EXIT THIS TEST.
10870
10871 054030 004737 036106      50$:  JSR  PC,TSABRT      ;REPORT TEST ABORTED. NON-TEST RELATED ERROR.
10872 054034 005037 002362      60$:  CLR  CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
10873
10874 054040              ENDTST
10875 054040
10876 054040 104401              L10066:
                                TRAP  C$ETST

```


HARDWARE TEST

- IAUTOI -

```

10876 .SBTTL  HARDWARE TEST          - IAUTOI -
10877 ;*****
10878 ;
10879 ;
10880 ;
10881 ;   This test verifies that the DUT's IAUTO function behaves correctly
10882 ;   when inactive, ie. IAUTO bit clear.
10883 ;   All active lines are tested individually by filling the FIFO
10884 ;   then reading the received data checking for the presence of
10885 ;   XOFF(ASCII DC3) or XON (ASCII DC1) characters.
10886 ;   If any are found then appropriate errors are reported.
10887 ;   Any BMP codes that are found will be placed on the BMP code queue,
10888 ;   to be reported later.
10889 ;   The characters are transmitted on all active lines, in internal
10890 ;   loopback mode.
10891 ;
10892 ;--*****
10893
10893 054042      BGNTST
10893 054042
10894          000034          TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
10895 054042 012737 000034 002364      MOV    #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (51)
10896 054050 012737 177777 002362      MOV    #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
10897 054056 012737 000001 005464      MOV    #1,ERRTYP      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
10898 054064 012737 011755 005466      MOV    #5101,ERRNBR      ;SET ERROR NUMBER TO 5101.
10899 054072 012737 021205 005470      MOV    #EM5101,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
10900 054100 012737 026440 005472      MOV    #ER9101,ERRBLK      ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
10901
10902 ;
10903 ; Reset the DUT to a known state, remove the status codes from the fifo.
10904 ; Clear TX and RX interrupt enable bits in the CSR.
10905 ; This subroutine reports error >>>> 5101 <<<<.
10906 054106 004737 030172      JSR    PC,CLNRST      ;RESET THE DHV11 M, REPORT ANY ERRORS FOUND.
10907 054112 103146      BCC    60$      ;EXIT TEST IF FATAL ERROR FOUND.
10908
10909 ;
10910 ; Initialize the 256 byte data pattern.
10911 ; Ensure the data pattern is free from XON's or XOFF's to prevent errors.
10912 ; Note: the first two characters and the last two characters will be the same.
10913
10914 054114 004737 031372      JSR    PC,INDTPX      ;INITIALISE DATA PATTERN.
10915
10916 ;
10917 ; Set internal loopback, disable IAUTO, enable receiver on the selected line.
10918 ; Set LPR to 38.4k baud, 8 bits per character, odd parity, 2 stop bits.
10919 054120 013705 002272      MOV    ACTLNS,R5      ;PASS THE ACTIVE LINE BIT MAP.
10920 054124 012700 000204      MOV    #204,R0      ;PASS INT'L LOPBCK, ENABLE RX, DISABLE IAUTO.
10921 054130 004737 040250      JSR    PC,WTWLNC      ;INITIALISE THE LINE CONTROL REGISTER.
10922 054134 012700 177670      MOV    #177670,R0      ;PASS THE LPR CONTENTS.
10923 054140 004737 040324      JSR    PC,WTWLPR      ; SET THE LPR CONTENTS TO 38.4K BAUD.
10924 054144 012704 000012      MOV    #10,R4      ;PASS DELAY TIME OF 10 MILLI SECONDS.
10925 054150 004737 030370      JSR    PC,DELAY      ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
10926
10927 ;
10928 ; Set up loop for all active lines.
10929 ; Test the state of the IAUTO bit prior to transmitting the data pattern.
10930 ; If the bit is set, then report the error and skip transmitting
10931 ; the data pattern on the selected line.

```

HARDWARE TEST

- IAUTOI -

```

10932      ; Transmit a 256 character data pattern using DMA, on a single channel
10933      ; Empty the fifo, and verify no XOFF or XON chars were found.
10934      ; -
10935 054154 005001      CLR      R1      ;CLEAR THE LINE NUMBER COUNTER.
10936 054156 005037 054426  CLR      55$      ;CLEAR STORAGE FOR LINE NUMBER.
10937 054162 012737 011756 005466 2$:  MOV      $5102.,ERRNBR ;SET THE ERROR NUMBER TO 5102.
10938 054170 004737 032722      JSR      PC,PUFIFO ;PURGE THE FIFO.
10939 054174 103111      BCC      50$      ;GO REPORT ERROR IF FIFO DID NOT PURGE.
10940 054176 000241      CLC      ;CLEAR CARRY PRIOR TO ROTATING BIT MAP.
10941 054200 006005      ROR      R5      ;ROTATE THE BIT MAP INTO THE CARRY BIT.
10942 054202 103077      BCC      12$      ;BRANCH IF LINE IS INACTIVE.
10943      ;+
10944      ; Test the IAUTO bit on the selected active line.
10945      ; Report error if it is set.
10946      ; Do not transmit the data pattern on the selected line.
10947      ; -
10948 054204 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5103.
10949 054210 010177 126064      MOV      R1,0CSRA ;SELECT LINE TO TEST.
10950 054214 032777 000002 126066  BIT      $BIT1,0LNCTRA ;TEST THE STATE OF THE IAUTO BIT ON THIS LINE.
10951 054222 001404      BEQ      4$      ;SKIP ERROR IF IAUTO BIT CLEAR.
10952 054224 012702 021233      MOV      $EM5102,R2 ;PASS THE CORRECT ERROR MESSAGE.
10953 054230      ERROR      ;
10954 054232 104460      ;>>>> ERROR <<<<. TRAP      C$ERROR
10955      BR      12$      ;SKIP TRANSMITTING DATA PATTERN.
10956      ;+
10957      ; Transmit data pattern of 256 chars.
10958      ; -
10959 054234 005237 005466 4$:  INC      ERRNBR      ;SET ERROR NUMBER TO 5104.
10960 054240 012702 004012      MOV      $0UFBAS,R2 ;PASS THE START OF THE DATA PATTERN TO TX.
10961 054244 012703 000400      MOV      $256.,R3 ;PASS THE LENGTH OF THE DATA PATTERN.
10962 054250 004737 030624      JSR      PC,DODMA ;TRANSMIT THE DATA PATTERN.
10963 054254 103061      BCC      50$      ;ABORT THE TEST IF ERROR FOUND DURING DMA TX.
10964      ;+
10965      ; Wait for DMA to complete, then wait for the last character plus XOFF
10966      ; to arrive in the fifo.
10967      ; -
10968 054256 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5105.
10969 054262 012701 170536      MOV      $170536,R1 ;PASS TIME-OUT VALUE OF 350 MILLI SECS.
10970 054266 013702 002300      MOV      CSRA,R2 ;PASS THE ADDRESS OF THE CSR.
10971 054272 004737 037740      JSR      PC,WAIBIS ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
10972 054276 103050      BCC      50$      ;IF NO TX_ACTION WAS RECEIVED, ABORT THE TEST.
10973 054300 012704 000012      MOV      $10.,R4 ;PASS DELAY OF 10 MILLI SECS.
10974 054304 004737 030370      JSR      PC,DELAY ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.
10975      ;+
10976      ; Read 256 chars from the fifo. Report error if any XOFF's or XON's
10977      ; are found.
10978      ; -
10979 054310 005237 005466      INC      ERRNBR      ;INCREMENT ERROR NUMBER TO 5106.
10980 054314 012701 000400      MOV      $256.,R1 ;INITIALISE THE READ COUNTER.
10981 054320 017702 125756 6$:  MOV      $RBUFA,R2 ;READ CHAR FROM THE FIFO.
10982 054324 100035      BPL      50$      ;GO REPORT ERROR IF FIFO EMPTY.
10983      ;+
10984      ; Check for BMP code in the fifo. Save any found on the queue.
10985      ; -
10986
10987

```

HARDWARE TEST

- IAUTOI -

```

10988 054326 012700 170301      MOV    #170301,R0      ;SET UP BMP BIT MASK.
10989 054332 040200              BIC     R2,R0          ;TRY TO CLEAR ALL THE BMP BITS.
10990 054334 001002              BNE     8$              ;SKIP BMPSAV IF NOT A BMP CODE.
10991 054336 004737 035624      JSR     PC,SAVBMP      ;SAVE THE BMP CODE ON THE QUEUE.
10992                               ;+
10993                               ; Check for XOFF and XON characters.
10994                               ;-
10995 054342 120227 000023      8$:    CMPB   R2,#23          ;IS IT AN XOFF CHARACTER?.
10996 054346 001406              BEQ     10$          ;YES; GO REPORT ERROR.
10997 054350 120227 000021      CMPB   R2,#21          ;NO; IS IT AN XON CHARACTER?.
10998 054354 001403              BEQ     10$          ;YES; GO REPORT ERROR.
10999 054356 005301              DEC     R1            ;DECREMENT THE READ COUNT.
11000 054360 001357              BNE     6$              ;LOOP TO READ THE NEXT CHAR.
11001 054362 000407              BR      12$          ;GO CHECK FOR ANY UNTESTED ACTIVE LINES.
11002
11003 054364 005237 005466      10$:    INC     ERRNBR          ;SET ERROR NUMBER TO 5107.
11004 054370 013701 054426      MOV     55$,R1          ;PASS THE LINE NUMBER TO BE REPORTED.
11005 054374 012702 021271      MOV     #EM5103,R2      ;PASS THE ERROR MESSAGE TO BE REPORTED.
11006 054400              ERROR              ;
11007              >>>>> ERROR <<<<<.
11008              TRAP     C$ERROR
11009              ;+
11010              ; Check if all active lines have been tested.
11011              ;-
11012 054402 005237 054426      12$:    INC     55$          ;INCREMENT LINE NUMBER.
11013 054406 013701 054426      MOV     55$,R1          ;GET NUMBER OF THE NEXT LINE TO TEST.
11014 054412 005705              TST     R5            ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
11015 054414 001262              BNE     2$              ;LOOP TO CHECK NEXT LINE.
11016 054416 000404              BR      60$          ;EXIT TEST.
11017
11018 054420 004737 036106      50$:    JSR     PC,TSABRT      ;REPORT TEST ABORTED. NON-TEST RELATED ERROR.
11019 054424 000401              BR      60$          ;EXIT THIS TEST.
11020 054426 000000              55$:    .WORD   C            ;STORAGE FOR LINE NUMBER.
11021 054430 005037 002362      60$:    CLR     CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
11022 054434              ENDTST
11023 054434
11024 054434 104401

```

L10067: TRAP C\$ETST

HARDWARE TEST

- IAUTOA -

```

11023
11024
11025
11026
11027
11028
11029
11030
11031
11032
11033
11034
11035
11036
11037
11038
11039 054436
      054436
11040      000035
11041 054436 012737 000035 002364
11042 054444 012737 177777 002362
11043 054452 012737 000001 005464
11044 054460 012737 012121 005466
11045 054466 012737 021321 005470
11046 054474 012737 026440 005472
11047
11048
11049
11050
11051
11052 054502 004737 030172
11053 054506 103161
11054
11055
11056
11057
11058
11059
11060 054510 004737 031372
11061
11062
11063
11064
11065 054514 013705 002272
11066 054520 012700 000206
11067 054524 004737 040250
11068 054530 012700 177670
11069 054534 004737 040324
11070 054540 012704 000012
11071 054544 004737 030370
11072
11073
11074
11075
11076
11077
11078

```

```

.SBTTL HARDWARE TEST - IAUTOA -
;*****
; - IAUTO BIT ACTIVE TEST -
;
; This test verifies that the DUT's IAUTO function behaves correctly
; when active, ie IAUTO asserted high.
; All active lines are tested individually by filling the FIFO, and
; checking for the presence of at least one XOFF(ASCII DC3) character
; and one XON (ASCII DC1) character.
; Any BMP codes that are found will be placed on the BMP code queue,
; to be reported later.
; The characters are transmitted on all active lines, in internal
; loopback mode.
;
;-----*****
BGNTST
                                T29::
TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV @TNUM,TSTNUM ;SET UP THE TEST NUMBER. (52)
MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV #5201,ERRNBR ;SET ERROR NUMBER TO 5201.
MOV #EM5201,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
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 >>>> 5201 <<<<.
;
JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
BCC 60$ ;EXIT TEST IF FATAL ERROR FOUND.
;
; Initialize the 256 byte data pattern.
; Ensure the data pattern is free from XON's or XOFF's to prevent errors.
; Note: the first two characters and the last two characters will be the same.
;
JSR PC,INDTPX ;INITIALISE DATA PATTERN.
;
; Set internal loopback, enable IAUTO and receiver on the selected line.
; Set LPR to 38.4k baud, 8 bits per character, odd parity, 2 stop bits.
;
MOV ACTLNS,R5 ;PASS THE ACTIVE LINE BIT MAP.
MOV #206,R0 ;PASS INTERNAL LOPBCK, ENABLE RX AND IAUTO.
JSR PC,WTWLNCR ;INITIALISE THE LINE CONTROL REGISTER.
MOV #177670,R0 ;PASS THE LPR CONTENTS.
JSR PC,WTWLPR ;SET THE LPR CONTENTS TO 38.4K BAUD.
MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI SECONDS.
JSR PC,DELAY ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
;
; Set up loop for all active lines.
; Test the state of the OAUTO bit prior to transmitting the data pattern.
; If the bit is clear, then report the error and skip transmitting
; the data pattern on the selected line.
; Transmit a 224 character data pattern using DMA, on a single channel

```

HARDWARE TEST

- IAUTOA -

```

11079      ; Empty the fifo, and count the XOFF and an XON chars found.
11080      ; -
11081 054550 005001      CLR      R1      ;CLEAR THE LINE NUMBER COUNTER.
11082 054552 005037 055050 CLR      55$    ;CLEAR STORAGE FOR LINE NUMBER.
11083 054556 012737 012122 005466 2$: MOV      #5202.,ERRNBR ;SET THE ERROR NUMBER TO 5202.
11084 054564 004737 032722 JSR      PC,PUFIFO ;PURGE THE FIFO.
11085 054570 103124      BCC      50$    ;GO REPORT ERROR IF FIFO DID NOT PURGE.
11086 054572 000241      CLC          ;CLEAR CARRY PRIOR TO ROTATING BIT MAP.
11087 054574 006005      ROR      R5      ;ROTATE THE BIT MAP INTO THE CARRY BIT.
11088 054576 103112      BCC      16$    ;BRANCH IF LINE IS INACTIVE.
11089      ;+
11090      ; Test the IAUTO bit on the selected active line.
11091      ; Report error if it is clear.
11092      ; Do not transmit the data pattern on the selected line.
11093      ; -
11094 054600 005237 005466      INC      ERRNBR ;SET ERROR NUMBER TO 5203.
11095 054604 010177 125470      MOV      R1,0CSRA ;SELECT LINE TO TEST.
11096 054610 032777 000002 125472 BIT      #BIT1,0LNCTRA ;TEST THE STATE OF THE IAUTO BIT ON THIS LINE.
11097 054616 001004      BNE      4$      ;SKIP ERROR IF IAUTO BIT SET.
11098 054620 012702 021345      MOV      #EM5202,R2 ;PASS THE CORRECT ERROR MESSAGE.
11099      ; "IAUTO BIT FOUND CLEAR ON LINE nn"
11100 054624      ERROR      ;
11101 054624 104460      ;>>>> ERROR <<<<<. TRAP      C$ERROR
11101 054626 000476      BR      16$    ;SKIP TRANSMITTING DATA PATTERN.
11102
11103      ;+
11104      ; Transmit data pattern to fill the fifo, 223 chars + 32 xoff's + xon.
11105      ; -
11106 054630 005237 005466 4$: INC      ERRNBR ;SET ERROR NUMBER TO 5204.
11107 054634 012702 004012      MOV      #0UFBAS,R2 ;PASS THE START OF THE DATA PATTERN TO TX.
11108 054640 012703 000337      MOV      #223.,R3 ;PASS THE LENGTH OF THE DATA PATTERN.
11109 054644 004737 030624      JSR      PC,DODMA ;TRANSMIT THE DATA PATTERN.
11110 054650 103074      BCC      50$    ;ABORT THE TEST IF ERROR FOUND DURING DMA TX.
11111
11112      ;+
11113      ; Wait for DMA to complete, then wait for the last character plus XOFF
11114      ; to arrive in the fifo.
11115      ; -
11116 054652 005237 005466      INC      ERRNBR ;SET ERROR NUMBER TO 5205.
11117 054656 012701 170536      MOV      #170536,R1 ;PASS TIME-OUT VALUE OF 350 MILLI SECS.
11118 054662 013702 002300      MOV      CSRA,R2 ;PASS THE ADDRESS OF THE CSR.
11119 054666 004737 037740      JSR      PC,WAIBIS ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
11120 054672 103063      BCC      50$    ;IF NO TX_ACTION WAS RECEIVED, ABORT THE TEST.
11121
11122      ;+
11123      ; Read 256 chars from the fifo, count any XOFF or XON chars found.
11124      ; -
11125 054674 005003      CLR      R3      ;CLEAR XOFF AND XON INDICATOR.
11126 054676 005237 005466      INC      ERRNBR ;INCREMENT ERROR NUMBER TO 5206.
11127 054702 012701 000400      MOV      #256.,R1 ;INITIALISE THE READ COUNTER.
11128 054706 017702 125370 6$: MOV      @RBUFA,R2 ;READ CHAR FROM THE FIFO.
11129 054712 100053      BPL      50$    ;GO REPORT ERROR IF FIFO EMPTY.
11130      ;+
11131      ; Check for BMP code in the fifo. Save any found on the queue.
11132      ;
11133 054714 012700 170301      MOV      #170301,R0 ;SET UP BMP BIT MASK.
11134 054720 040200      BIC      R2,R0 ;TRY TO CLEAR ALL THE BMP BITS.

```

HARDWARE TEST

- IAUTOA -

```

11135 054722 001002      BNE      8$      ;SKIP BMPSAV IF NOT A BMP CODE.
11136 054724 004737 035624 JSR      PC,SAVBMP ;SAVE THE BMP CODE ON THE QUEUE.
11137
11138      ; Check for XOFF and XON characters.
11139      ;
11140 054730 120227 000023 8$:      CMPB    R2,#23      ;IS IT AN XOFF CHARACTER?.
11141 054734 001002      BNE      10$      ;NO, BRANCH TO SEE IF IT IS AN XON.
11142 054736 052703 000001      BIS      #BIT0,R3 ;INDICATE THE XOFF CHAR.
11143 054742 120227 000021 10$:     CMPB    R2,#21      ;IS IT AN XON CHARACTER?.
11144 054746 001002      BNE      12$      ;NO, SKIP THE NEXT INSTRUCTION.
11145 054750 052703 000002      BIS      #BIT1,R3 ;INDICATE THE XON CHAR.
11146 054754 005301 12$:     DEC      R1      ;DECREMENT THE READ COUNT. 256 READ?
11147 054756 001410      BEQ      14$      ;YES, EXIT THE LOOP.
11148 054760 020127 000200      CMP      R1,#128. ;NO. HAVE WE READ EXACTLY 128 CHARS? $$$
11149 054764 001350      BNE      6$      ;NO, LOOP TO READ MORE.
11150 054766 012704 000012      MOV      #10.,R4 ;YES. DELAY 10MS TO WAIT FOR THE XON CHAR
11151 054772 004737 030370      JSR      PC,DELAY ; (WHICH IS LAST CHAR) TO GET INTO FIFO.
11152 054776 000743      BR       6$      ;LOOP TO READ MORE CHARS.
11153
11154      ; Verify then at least 1 XOFF and 1 XON was found in the fifo.
11155      ; Report error if none were found.
11156      ;
11157 055000 020327 000003 14$:     CMP      R3,#3      ;DID WE GET AT LEAST 1 XON AND 1 XOFF?
11158 055004 001407      BEQ      16$      ;YES, SKIP ERROR REPORT.
11159 055006 005237 005466      INC      ERRNBR ;NO, SET ERROR NUMBER TO 5207.
11160 055012 013701 055050      MOV      55$,R1 ;PASS THE LINE NUMBER TO BE REPORTED.
11161 055016 012702 021271      MOV      #EM5103,R2 ;PASS THE ERROR MESSAGE TO BE REPORTED.
11162
11163 055022      ERROR      ; "IAUTO BIT BAD ON LINE nn".
11164 055022 104460      ; >>>>> ERROR <<<<<.
11165      TRAP      C$ERROR
11166      ;
11167      ; Check if all active lines have been tested.
11168      ;
11169 055024 005237 055050 16$:     JPC      55$      ;INCREMENT LINE NUMBER.
11170 055030 013701 055050      MOV      55$,R1 ;GET NUMBER OF THE NEXT LINE TO TEST.
11171 055034 005705      TST      R5      ;ARE THERE ANY MORE ACTIVE LINES TO TEST?.
11172 055036 001247      BNE      2$      ;LOOP TO CHECK NEXT LINE.
11173 055040 000404      BR       60$      ;EXIT TEST.
11174
11175 055042 004737 036106 50$:     JSR      PC,TSABRT ;REPORT TEST ABORTED. NON-TEST RELATED ERROR.
11176 055046 000401      BR       60$      ;EXIT THIS TEST.
11177 055050 000000 55$:     .WORD    0      ;STORAGE FOR LINE NUMBER.
11178 055052 005037 002362 60$:     CLR      CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.
11179
11180      ENDTST
11181
11182 055056      L10070:
11183 055056      TRAP      C$ETST
11184 055056 104401

```


HARDWARE TEST

- FIFDAT -

```

11180
11181
11182
11183
11184
11185
11186
11187
11188
11189
11190
11191
11192
11193
11194
11195
11196 055060
      055060
11197
11198 055060 000036
11199 055066 012737 000036 002364
11200 055074 012737 177777 002362
11201 055102 012737 000001 005464
11202 055110 012737 012265 005466
11203 055110 012737 021403 005470
11204
11205
11206
11207
11208 055116 004737 030172
11209 055122 103107
11210
11211
11212
11213
11214 055124 004737 030772
11215 055130 103104
11216 055132 004737 C31342
11217
11218
11219
11220
11221
11222
11223
11224
11225 055136 012700 000204
11226 055142 004737 040250
11227 055146 012700 177670
11228 055152 004737 040324
11229 055156 012704 000012
11230 055162 004737 030370
11231 055166 012702 0040'2
11232 055172 012703 000400
11233 055176 005237 005466
11234 055202 004737 030624
11235 055206 103053

```

```

.SBTTL HARDWARE TEST          - FIFDAT -
;*****
;                                - FIFO VALID DATA TEST -
;
; This test verifies that the DUT is capable of holding 256 valid
; characters in its fifo.
; The characters are transmitted on the first available active line, in
; internal loopback mode.
; The data found in the fifo is compared with the expected data, and any
; discrepancies are reported.
; Any BMP code found will invalidate the test and cause it to be aborted.
; However the BMP code will be placed on the BMP code queue, to be
; reported later.
;*****
;--*****

      BGNTST

      T30::
      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV    #TNUM,TSTNUM      ;SET UP THE TEST NUMBER.          (53)
      MOV    #-1,CTRLCF        ;INDICATE THAT WE ARE IN A TEST.
      MOV    #1,ERRTP          ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV    #5301,ERRNBR      ;SET ERROR NUMBER TO 5301.
      MOV    #EMS301,ERRMSG     ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
;
; 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 >>>> 5301 <<<<.
;
      JSR    PC,CLNRST         ;RESET THE DHV11-M. REPORT ANY ERRORS FOUND.
      BCC    60$              ;EXIT TEST IF FATAL ERROR FOUND.
;
; Find an active line on which to perform the test.
; Initialise 256 byte data pattern.
;
      JSR    PC,FINACT         ;FIND AN ACTIVE LINE.
      BCC    60$              ;EXIT IF NO ACTIVE LINES FOUND.
      JSR    PC,INDATP         ;INITIALISE THE DATA PATTERN.
;
; Transmit a 265 character data pattern using DMA, on a single channel
; at 38.4k baud, 8 bits per character, odd parity, 2 stop bits.
;
;
; Set internal loopback on the selected line.
; Transmit the data pattern on the first available active line.
;
      MOV    #204,R0           ;PASS PARAMETER FOR INTERNAL LOPBCK,ENABLE RX.
      JSR    PC,WTWLNCR        ;INITIALISE THE LINE CONTROL REGISTER.
      MOV    #177670,R0        ;PASS THE LPR CONTENTS.
      JSR    PC,WTWLPR         ;SET THE LPR CONTENTS TO 38.4K BAUD.
      MOV    #10.,R4           ;PASS DELAY TIME OF 10 MILLI SECONDS.
      JSR    PC,DELAY          ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
      MOV    #BUFBAS,R2        ;PASS THE START OF THE DATA PATTERN TO TX.
      MOV    #BUFHID-BUFBAS,R3 ;PASS THE LENGTH OF THE DATA PATTERN.
      INC    ERRNBR            ;SET ERROR NUMBER TO 5302.
      JSR    PC,DODMA          ;TRANSMIT THE DATA PATTERN.
      BCC    50$              ;ABORT TEST IF ERROR FOUND DURING DMA TX.

```

HARDWARE TEST

- FIFDAT -

```

11236
11237
11238
11239
11240 055210 005237 005466
11241 055214 010103
11242 055216 012701 170536
11243 055222 013702 002300
11244 055226 004737 037740
11245 055232 103041
11246 055234 012704 000005
11247 055240 004737 030370
11248
11249
11250
11251
11252 055244 006303
11253 055246 005004
11254 055250 013705 002302
11255 055254 012737 012270 005466 2:
11256 055262 011502
11257 055264 100024
11258
11259
11260
11261
11262
11263 055266 005237 005466
11264 055272 004737 027256
11265 055276 103002
11266 055300
11267 055302 104460
11268 000417
11269 055304 005237 005466
11270 055310 120402
11271 055312 001406
11272 055314 012737 025420 005472
11273 055322 012701 021430
11274
11275 055326
11276 055326 104460
11277 055332 105204
11278 055334 001350
11279 000402
11280 055336 004737 036106
11281 055342 005037 002362
11282
11283 055346
11284 055346
11285 055346 104401

```

```

;+
; Wait for DMA to complete, then wait for the last character to arrive in
; the fifo.
;-
INC ERRNBR ;SET ERROR NUMBER TO 5303.
MOV R1,R3 ;SAVE THE NUMBER OF THE SELECTED ACTIVE LINE.
MOV #170536,R1 ;PASS TIME-OUT VALUE OF 350 MILLI SECS.
MOV CSRA,R2 ;PASS THE ADDRESS OF THE CSR.
JSR PC,WAIBIS ;WAIT FOR DMA TO COMPLETE, TX ACTION SET.
BCC 50: ;BRANCH IF FIFO EMPTY, ABORT THE TEST.
MOV #5,R4 ;PASS DELAY OF 5 MILLI SECS.
JSR PC,DELAY ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.

;+
; Read the FIFO checking for data corruption, report any errors found.
; Abort the test if a BMP code was found in the fifo.
;-
ASL R3 ;MULTIPLY BY 2.
CLR R4 ;INITIALISE THE EXPECTED DATA.
MOV RBUFA,R5 ;GET THE ADDRESS OF THE RECEIVER BUFFER REG.
MOV #5304,ERRNBR ;SET UP ERROR NUMBER EACH TIME AROUND LOOP.
MOV (R5),R2 ;GET THE ACTUAL DATA FROM THE FIFO.
BPL 50: ;ABORT THE TEST IF THE FIFO IS EMPTY.

;+
; Check if the read character is a BMP code.
; If it is a BMP code save it on the queue to be reported later, and
; abort the test.
;-
INC ERRNBR ;SET ERROR NUMBER TO 5305.
JSR PC,CHKBMP ;CHECK IF CHARACTER IS A BMP CODE.
BCC 4: ;BRANCH IF NOT A BMP CODE.
ERROR ;
; >>>> ERROR 5305 <<<<.
; TRAP C:ERROR
BR 60: ;ABORT THIS TEST.

4: INC ERRNBR ;SET ERROR NUMBER TO 5306.
CMPB R4,R2 ;COMPARE THE EXPECTED WITH THE ACTUAL DATA.
BEQ 8: ;SKIP ERROR REPORT IF DATA IS OK.
MOV #ER9002,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
MOV #EM5302,R1 ;PASS THE MESSAGE TO BE REPORTED.
;REPORT THE ERROR "FIFO BAD, DATA FIELD CORRUPTED"
6: ERROR ;
; >>>> ERROR 5306 <<<<.
; TRAP C:ERROR
8: INCB R4 ;INCREMENT THE EXPECTED DATA.
BNE 2: ;LOOP IF NOT DONE.
BR 60: ;EXIT

50: JSR PC,TSABRT ;ABORT THE TEST, REASON SHOWN BY ERROR NUMBER.
60: CLR CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.

ENDTST

```

L10071: TRAP C:ETST

HARDWARE TEST

- FI3QLI -

```

11285 .SBTTL  HARDWARE TEST          - FI3QLI -
11286 ;*****
11287 ;
11288 ;
11289 ;
11290 ;
11291 ;
11292 ;
11293 ;
11294 ;
11295 ;
11296 ;
11297 ;
11298 ;
11299 ;
11300 ;
11301 055350      BGNTST
11302 055350
11303 055350 000037      TNUM ** TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
11304 055356 012737 000037 002364      MOV  #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (54)
11305 055364 012737 177777 002362      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
11306 055372 012737 000001 005464      MOV  #1,ERRTP      ;SET FATAL ERROR TYPE IN ERROR TABLE.
11307 055400 012737 012431 005466      MOV  #5401,ERRNBR      ;SET ERROR NUMBER TO 5401.
11308 055406 012737 021560 005470      MOV  #EM5401,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
11309 055406 012737 024644 005472      MOV  #ER0503,ERRBLK      ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
11310 ;
11311 ;
11312 ;
11313 ;
11314 055414 004737 030172      JSR    PC,CLNRST      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
11315 055420 103111      BCC    60$      ;EXIT TEST IF FATAL ERROR FOUND.
11316 ;
11317 ;
11318 ;
11319 055422 004737 030772      JSR    PC,FINACT      ;FIND THE NUMBER OF THE FIRST ACTIVE LINE.
11320 055426 103106      BCC    60$      ;EXIT IF NO LINES ARE AVAILABLE.
11321 ;
11322 ;
11323 ;
11324 ;
11325 ;
11326 ;
11327 055430 004737 031372      JSR    PC,INDTPX      ;INITIALISE THE DATA PATTERN.
11328 ;
11329 ;
11330 ;
11331 ;
11332 ;
11333 ;
11334 ;
11335 ;
11336 055434 012700 000206      MOV  #206,R0      ;PASS INTERNAL LOPBCK, ENABLE RX AND IAUTO.
11337 055440 004737 040250      JSR    PC,WTWLNC      ;INITIALISE THE LINE CONTROL REGISTER.
11338 055444 012700 177670      MOV  #177670,R0      ;PASS THE LPR CONTENTS.
11339 055450 004737 040324      JSR    PC,WTWLPR      ;SET THE LPR CONTENTS TO 38.4K BAUD.
11340 055454 012704 000012      MOV  #10.,R4      ;PASS DELAY TIME OF 10 MILLI SECONDS.

```

This test verifies that the DUT's fifo 3/4 level alarm system remains inactive while it contains 191 characters or less. The test looks for an XOFF (ASCII DC3) character in the fifo. If any XOFF's are found an error will be reported and the test aborted. Any BMP code found will invalidate the test and cause it to be aborted. However the BMP code will be placed on the BMP code queue, to be reported later. The characters are transmitted on the first available active line, in internal loopback mode.

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 >>>> 5401 <<<<.

Find an active line on which to perform the test.

Initialize the 256 byte data pattern. Ensure the data pattern is free from XON's or XOFF's to prevent errors. Note: the first two characters and the last two characters will be the same.

Transmit a 191 character data pattern using DMA, on a single channel at 38.4k baud, 8 bits per character, odd parity, 2 stop bits.

Set internal loopback, enable IAUTO and RX on the selected line. Transmit the data pattern on the first available active line.

HARDWARE TEST

- FI3QLI -

```

11341 055460 004737 030370      JSR    PC,DELAY      ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
11342 055464 012702 004012      MOV     @BUFBAS,R2    ;PASS THE START OF THE DATA PATTERN TO TX.
11343 055470 012703 000277      MOV     @191.,R3     ;PASS THE LENGTH OF THE DATA PATTERN.
11344 055474 004737 030624      JSR     PC,DODMA     ;TRANSMIT THE DATA PATTERN.
11345 055500 103057              BCC     50$           ;IF ERROR FOUND DURING DMA THEN ABORT TEST.
11346
11347
11348
11349
11350
11351 055502 005237 005466      INC     ERRNBR        ;SET ERROR NUMBER TO 5402.
11352 055506 012701 170454      MOV     @170454,R1    ;PASS TIME-OUT VALUE OF 300 MILLI SECS.
11353 055512 013702 002300      MOV     CSRA,R2       ;PASS THE ADDRESS OF THE CSR.
11354 055516 004737 037740      JSR     PC,WAIBIS     ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
11355 055522 103046              BCC     50$           ;IF FIFO EMPTY, REPORT ERROR, ABORT THE TEST.
11356 055524 012704 000005      MOV     @5,R4         ;PASS DELAY OF 5 MILLI SECS.
11357 055530 004737 030370      JSR     PC,DELAY     ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.
11358
11359
11360
11361
11362
11363
11364
11365
11366 055534 005004              ;
11367 055536 013705 002302      CLR     R4            ;CLEAR THE CHARACTER COUNT.
11368 055542 012737 012267 005466 2$: MOV     RBUFA,R5      ;GET THE ADDRESS OF THE RECEIVER BUFFER REG.
11369 055550 011502              MOV     @5303.,ERRNBR ;SET ERROR NUMBER TO 5403.
11370 055552 100032              MOV     (R5),R2       ;GET THE ACTUAL DATA FROM THE FIFO.
11371 055554 005204              BPL     50$           ;FIFO EMPTY, ABORT TEST.
11372
11373
11374
11375
11376
11377 055556 005237 005466      INC     ERRNBR        ;SET ERROR NUMBER TO 5404.
11378 055562 004737 027256      JSR     PC,CHKBMP     ;CHECK IF CHARACTER IS A BMP CODE.
11379 055566 103001              BCC     4$           ;BRANCH IF NOT A BMP CODE.
11380
11381 055570 000421              ;REPORT ERROR "BMP CODE FOUND IN FIFO, TEST INVALIDATED".
11382
11383
11384
11385
11386 055572 005237 005466      BR      8$           ;REPORT THE ERROR AND ABORT THE TEST.
11387 055576 122702 000023      ;
11388 055602 001003              ;
11389 055604 012701 021617      ;
11390
11391 055610 000411              ;
11392
11393 055612 005237 005466      ;
11394 055616 020427 000277      ;
11395 055622 001347              ;
11396 055624 011502              ;
11397 055626 100006              ;

```

SEQ 0288

HARDWARE TEST

- FI3QLI -

11398 055630 012701 021617

11399

11400

11401 055634

055634 104460

11402 055636 000402

11403

11404

11405 055640 004737 036106

11406 055644 005037 002362

11407

11408 055650

055650

055650 104401

```
MOV    #EM5402,R1      ;PASS THE MESSAGE TO BE REPORTED.
;REPORT THE ERROR "FIFO BAD, ALARM SIGNAL DEFECTIVL".
```

83:

ERROR

```

;      >>>> ERRORS 5304 THRU 5306 <<<<.
                                TRAP      C:ERROR

```

BR 60\$

```
;EXIT THE TEST.
```

504:

JSR PC.TSABRT

;>>>> ERRORS 5402 AND 5403 <<<<.

60\$:

CLR CTRLCF

```

;REPORT TEST ABORTED. NON-TEST RELATED ERROR.
;INDICATE THAT WE ARE NOT WITHIN A TEST.

```

ENDTST

L10072:

TRAP CSETST

HARDWARE TEST

- FI3QLA -

```

11410 .SBTTL  HARDWARE TEST          - FI3QLA -
11411 ;*****
11412 ;
11413 ;
11414 ;
11415 ;
11416 ;
11417 ;
11418 ;
11419 ;
11420 ;
11421 ;
11422 ;
11423 ;
11424 ;
11425 ;
11426 ;
11427 055652      BGNTST
      055652
11428      000040      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
11429 055652 012737 000040 002364      MOV  #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (55)
11430 055660 012737 177777 002362      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
11431 055666 012737 000001 005464      MOV  #1,ERRTYP      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
11432 055674 012737 012575 005466      MOV  #5501,ERRNBR      ;SET ERROR NUMBER TO 5501.
11433 055702 012737 021660 005470      MOV  #EM5501,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
11434 ;
11435 ;
11436 ;
11437 ;
11438 ;
11439 055710 004737 030172      JSR    PC,CLNRST      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
11440 055714 103402      BCS    .+6      ;SKIP EXIT OF TEST IF NO FATAL ERROR FOUND.
11441 055716 000137 056336      JMP    60$      ;EXIT TEST FATAL ERROR FOUND.
11442 ;
11443 ;
11444 ;
11445 055722 004737 030772      JSR    PC,FINACT      ;FIND AN ACTIVE LINE.
11446 055726 103402      BCS    .+6      ;SKIP EXIT OF TEST IF ACTIVE LINE FOUND.
11447 055730 000137 056336      JMP    60$      ;EXIT TEST.
11448 ;
11449 ;
11450 ;
11451 ;
11452 ;
11453 055734 004737 031372      JSR    PC,INDTPX      ;INITIALISE DATA PATTERN.
11454 ;
11455 ;
11456 ;
11457 ;
11458 ;
11459 ;
11460 ;
11461 ;
11462 055740 005237 005466      2$: INC  ERRNBR      ;SET ERROR NUMBER TO 5502.
11463 055744 012700 000206      MOV  #206,R0      ;PASS INTERNAL LOPBCK, ENABLE RX AND IAUTO.
11464 055750 004737 040250      JSR    PC,WTWLCN      ;INITIALISE THE LINE CONTROL REGISTER.
11465 055754 012700 177670      MOV  #177670,R0      ;PASS THE LPR CONTENTS.

```

HARDWARE TEST

- FI3QLA -

```

11466 055760 004737 040324      JSR    PC,WTWLPB      ;SET THE LPR CONTENTS TO 38.4K BAUD.
11467 055764 012704 000012      MOV    #10.,R4        ;PASS DELAY TIME OF 10 MILLI SECONDS.
11468 055770 004737 030370      JSR    PC,DELAY       ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
11469 055774 010105              MOV    R1,R5           ;COPY THE LINE NUMBER.
11470 055776 012702 004012      MOV    #0UFBAS,R2    ;PASS THE START OF THE DATA PATTERN TO TX.
11471 056002 012703 000277      MOV    #191.,R3      ;PASS THE LENGTH OF THE DATA PATTERN.
11472 056006 004737 030624      JSR    PC,DODMA       ;TRANSMIT THE DATA PATTERN.
11473 056012 103147              BCC     50$             ;ABORT THE TEST IF ERROR FOUND DURING DMA TX.
11474
11475
11476
11477
11478
11479 056014 005237 005466      ;+
11480 056020 012701 170454      ; Wait for DMA to complete, then wait for the last character to arrive in
11481 056024 013702 002300      ; the fifo.
11482 056030 004737 037740      ;+
11483 056034 103136              ; -
11484 056036 012704 000005      INC     ERRNBR        ;SET ERROR NUMBER TO 5503.
11485 056042 004737 030370      MOV    #170454,R1    ;PASS TIME-OUT VALUE OF 300 MILLI SECS.
11486
11487      MOV    CSRA,R2    ;PASS THE ADDRESS OF THE CSR.
11488
11489 056046 005237 005466      JSR    PC,WAIBIS      ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
11490 056052 010501              BCC     50$             ;IF NO TX_ACTION WAS RECEIVED, ABORT THE TEST.
11491 056054 012702 004012      MOV    #5,R4         ;PASS DELAY OF 5 MILLI SECS.
11492 056060 012703 000001      JSR    PC,DELAY       ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.
11493 056064 004737 030624
11494 056070 103120
11495
11496
11497
11498
11499
11500 056072 005237 005466      ;+
11501 056076 012701 170012      ; Transmit a null character which will cause an XOFF to be generated.
11502 056102 013702 002300      ; -
11503 056106 004737 037740      INC     ERRNBR        ;SET ERROR NUMBER TO 5504.
11504 056112 103107              MOV    R5,R1           ;PASS THE LINE NUMBER.
11505 056114 012704 000005      MOV    #0UFBAS,R2    ;PASS THE START OF THE DATA PATTERN TO TX.
11506 056120 004737 030370      MOV    #1,R3         ;PASS THE NUMBER OF
11507
11508
11509
11510 056124 012702 004012      JSR    PC,DODMA       ;TX A NULL CHARACTER TO CAUSE AN XOFF.
11511 056130 105022              BCC     50$             ;ABORT THE TEST IF ERROR FOUND DURING DMA TX.
11512 056132 020227 004412
11513 056136 103774
11514
11515
11516
11517
11518
11519 056140 005237 005466      ;+
11520 056144 010501              ; Wait for the XOFF to be received before TX the next 42 characters
11521 056146 012702 004012      ; which will cause a further 21 XOFF's to be generated.
11522 056152 012703 000037      ; -
11523
11524
11525
11526
11527
11528
11529
11530
11531
11532
11533
11534
11535
11536
11537
11538
11539
11540
11541
11542
11543
11544
11545
11546
11547
11548
11549
11550
11551
11552
11553
11554
11555
11556
11557
11558
11559
11560
11561
11562
11563
11564
11565
11566
11567
11568
11569
11570
11571
11572
11573
11574
11575
11576
11577
11578
11579
11580
11581
11582
11583
11584
11585
11586
11587
11588
11589
11590
11591
11592
11593
11594
11595
11596
11597
11598
11599
11600
11601
11602
11603
11604
11605
11606
11607
11608
11609
11610
11611
11612
11613
11614
11615
11616
11617
11618
11619
11620
11621
11622
11623
11624
11625
11626
11627
11628
11629
11630
11631
11632
11633
11634
11635
11636
11637
11638
11639
11640
11641
11642
11643
11644
11645
11646
11647
11648
11649
11650
11651
11652
11653
11654
11655
11656
11657
11658
11659
11660
11661
11662
11663
11664
11665
11666
11667
11668
11669
11670
11671
11672
11673
11674
11675
11676
11677
11678
11679
11680
11681
11682
11683
11684
11685
11686
11687
11688
11689
11690
11691
11692
11693
11694
11695
11696
11697
11698
11699
11700
11701
11702
11703
11704
11705
11706
11707
11708
11709
11710
11711
11712
11713
11714
11715
11716
11717
11718
11719
11720
11721
11722
11723
11724
11725
11726
11727
11728
11729
11730
11731
11732
11733
11734
11735
11736
11737
11738
11739
11740
11741
11742
11743
11744
11745
11746
11747
11748
11749
11750
11751
11752
11753
11754
11755
11756
11757
11758
11759
11760
11761
11762
11763
11764
11765
11766
11767
11768
11769
11770
11771
11772
11773
11774
11775
11776
11777
11778
11779
11780
11781
11782
11783
11784
11785
11786
11787
11788
11789
11790
11791
11792
11793
11794
11795
11796
11797
11798
11799
11800
11801
11802
11803
11804
11805
11806
11807
11808
11809
11810
11811
11812
11813
11814
11815
11816
11817
11818
11819
11820
11821
11822
11823
11824
11825
11826
11827
11828
11829
11830
11831
11832
11833
11834
11835
11836
11837
11838
11839
11840
11841
11842
11843
11844
11845
11846
11847
11848
11849
11850
11851
11852
11853
11854
11855
11856
11857
11858
11859
11860
11861
11862
11863
11864
11865
11866
11867
11868
11869
11870
11871
11872
11873
11874
11875
11876
11877
11878
11879
11880
11881
11882
11883
11884
11885
11886
11887
11888
11889
11890
11891
11892
11893
11894
11895
11896
11897
11898
11899
11900
11901
11902
11903
11904
11905
11906
11907
11908
11909
11910
11911
11912
11913
11914
11915
11916
11917
11918
11919
11920
11921
11922
11923
11924
11925
11926
11927
11928
11929
11930
11931
11932
11933
11934
11935
11936
11937
11938
11939
11940
11941
11942
11943
11944
11945
11946
11947
11948
11949
11950
11951
11952
11953
11954
11955
11956
11957
11958
11959
11960
11961
11962
11963
11964
11965
11966
11967
11968
11969
11970
11971
11972
11973
11974
11975
11976
11977
11978
11979
11980
11981
11982
11983
11984
11985
11986
11987
11988
11989
11990
11991
11992
11993
11994
11995
11996
11997
11998
11999

```

HARDWARE TEST

- FI3QLA -

```

11523 056156 004737 030624      JSR    PC,DODMA      ;TRANSMIT THE DATA PATTERN.
11524 056162 103063              BCC    50$             ;ABORT THE TEST IF ERROR FOUND DURING DMA TX.
11525                          ;+
11526                          ; Wait for the XOFF's and the null characters to be received.
11527                          ; -
11528 056164 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5507.
11529 056170 012701 170454      MOV    #170454,R1    ;PASS TIME-OUT VALUE OF 300 MILLI SECS.
11530 056174 013702 002300      MOV    CSRA,R2      ;PASS THE ADDRESS OF THE CSR.
11531 056200 004737 037740      JSR    PC,WAIBIS    ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
11532 056204 103052              BCC    50$             ;IF NO TX_ACTION WAS RECEIVED, ABORT THE TEST.
11533 056206 012704 000005      MOV    #5,R4         ;PASS DELAY OF 5 MILLI SECS.
11534 056212 004737 030370      JSR    PC,DELAY     ;WAIT FOR XOFF TO GET INTO THE FIFO.
11535                          ;+
11536                          ; Read the fifo until empty, counting the number of XOFF characters
11537                          ; that are found.
11538                          ; -
11539 056216 005004              CLR    R4             ;CLEAR CHARACTER COUNTER.
11540 056220 005003              CLR    R3             ;CLEAR THE XOFF FOUND COUNTER.
11541 056222 012701 170001      MOV    #170001,R1    ;INDICATE TO TEST DATA.VALID BIT, TIME-OUT 1MS.
11542 056226 012737 012604 005466 6$: MOV    #5508.,ERRNBR ;SET UP ERROR NUMBER EACH TIME AROUND THE LOOP.
11543 056234 013702 002302      MOV    RBUFA,R2    ;INDICATE TO CHECK RECEIVE BUFFER REGISTER.
11544 056240 004737 037740      JSR    PC,WAIBIS    ;WAIT FOR RECEIVED CHAR OR TIME-OUT.
11545 056244 103032              BCC    50$             ;GO REPORT ERROR IF FIFO EMPTY.
11546 056246 005204              INC    R4             ;COUNT THE CHARACTER.
11547                          ;+
11548                          ; Check if for BMP codes in the fifo, abort the test if any are found.
11549                          ; Save the BMP code on the queue to be reported later.
11550                          ; -
11551 056250 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5509.
11552 056254 004737 027256      JSR    PC,CHKBMP    ;CHECK IF WE HAVE GOT A BMP CODE.
11553 056260 103422              BCS    12$           ;GO REPORT THE ERROR IF WE FOUND A BMP CODE.
11554                          ;+
11555                          ; Check for XOFF character.
11556                          ; -
11557 056262 122702 000023 8$:  CMPB    #23,R2      ;CHECK IF THE RECEIVED CHARACTER WAS AN XOFF.
11558 056266 001001              BNE    10$           ;BRANCH IF CHARACTER WAS NOT AN XOFF.
11559 056270 005203              INC    R3             ;INCREMENT XOFF FOUND COUNT.
11560                          ;+
11561                          ; Check if all the characters including the XON have been removed.
11562                          ; -
11563 056272 020427 000400 10$: CMP    R4,#256.      ;CHECK IF WE HAVE REMOVED ALL THE CHARACTERS.
11564 056276 002753              BLT    6$             ;GO GET THE NEXT CHAR IF WE HAVE NOT FINISHED.
11565                          ;+
11566                          ; Check if the correct number of XOFF's were found in the fifo,
11567                          ; report error if count is incorrect.
11568                          ; -
11569                          ;
11570 056300 013737 012606 005466 MOV    5510.,ERRNBR ;SET UP THE ERROR NUMBER TO 5510.
11571 056306 022703 000040      CMP    #32.,R3      ;COMPARE EXPECTED XOFF COUNT WITH ACTUAL COUNT.
11572 056312 001411              BEQ    60$           ;EXIT TEST IF SUCCESS.
11573 056314 012737 024644 005472 MOV    #ER0503,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
11574 056322 012701 021617      MOV    #EM5402,R1    ;PASS THE MESSAGE TO BE REPORTED.
11575                          ;REPORT THE ERROR "FIFO BAD, ALARM SIGNAL DEFECTIVE".
11576 056326 104460 12$:      ERROR          ;
11577 056330 000402              BR     60$           >>>> ERROR <<<<.
11578                          ;ABORT THE TEST.          TRAP    C$ERROR

```


HARDWARE TEST - FI3QLA -

11579 056332 004737 036106
11580 056336 005037 002362
11581
11582 056342
056342
056342 104401

50\$: JSR PC,TSABRT
60\$: CLR CTRLCF

ENDTST

;REPORT TEST ABORTED. ERROR # SHOWS REASON.
;INDICATE THAT WE ARE NOT WITHIN A TEST.

L10073: TRAP C\$ETST

HARDWARE TEST

- FI3QAI -

```

11584
11585
11586
11587
11588
11589
11590
11591
11592
11593
11594
11595
11596
11597
11598 056344
      056344
11599      000041
11600 056344 012737 000041 002364
11601 056352 012737 177777 002362
11602 056360 012737 000001 005464
11603 056366 012737 012741 005466
11604 056374 012737 021715 005470
11605
11606
11607
11608
11609
11610 056402 004737 030172
11611 056406 103402
11612 056410 000137 057026
11613 056414
11614
11615
11616
11617 056414 004737 030772
11618 056420 103402
11619 056422 000137 057026
11620
11621
11622
11623
11624
11625 056426 004737 031372
11626
11627
11628
11629
11630
11631
11632
11633
11634 056432 005237 005466
11635 056436 012700 000206
11636 056442 004737 040250
11637 056446 012700 177670
11638 056452 004737 040324
11639 056456 012704 000012

```

```

.SBTTL HARDWARE TEST - FI3QAI -
;*****
; - FIFO 3/4 ALARM LEVEL ACTIVE/INACTIVE TEST -
;
; This test verifies that the DUT's fifo 3/4 level alarm system
; becomes active and inactive at the correct levels.
; Any BMP code found will invalidate the test and cause it to be aborted.
; However the BMP code will be placed on the BMP code queue, to be
; reported later.
; The characters are transmitted on the first available active line, in
; internal loopback mode.
;*****
;-----
      BGNTST
;
; T33:
      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV  #TNUM,TSTNUM    ;SET UP THE TEST NUMBER. (56)
      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
      MOV  #1,ERRTP        ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV  #5601,ERRNBR    ;SET ERROR NUMBER TO 5601.
      MOV  #EM5601,ERRMSG  ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
;
; 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 >>>> 5601 <<<<.
;
      JSR  PC,CLNRST      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCS  2$             ;SKIP EXITING TEST A SUCCESSFUL RESET.
      JMP  60$            ;EXIT THIS TEST.
2$:
;
; Find an active line on which to perform the test.
;
      JSR  PC,FINACT      ;FIND AN ACTIVE LINE.
      BCS  .+6            ;SKIP EXIT OF TEST IF ACTIVE LINE FOUND.
      JMP  60$            ;EXIT TEST.
;
; Initialize the 256 byte data pattern.
; Ensure the data pattern is free from XON's or XOFF's to prevent errors.
; Note: the first two characters and the last two characters will be the same.
;
      JSR  PC,INDTPX      ;INITIALISE THE DATA PATTERN.
;
; Transmit a 256 character data pattern using DMA, on a single channel
; at 38.4k baud, 8 bits per character, odd parity, 2 stop bits.
;
;
; Set internal loopback, enable IAUTO and receiver on the selected line.
; Transmit the first 191 characters on the first available active line.
;
      INC  ERRNBR          ;SET ERROR NUMBER TO 5602.
      MOV  #206,R0         ;PASS INTERNAL LOPBCK, ENABLE RX AND IAUTO.
      JSR  PC,WTWLNC       ;INITIALISE THE LINE CONTROL REGISTER.
      MOV  #177670,R0      ;PASS THE LPR CONTENTS.
      JSR  PC,WTWLPR       ;SET THE LPR CONTENTS TO 38.4K BAUD.
      MOV  #10.,R4         ;PASS DELAY TIME OF 10 MILLI SECONDS.

```


HARDWARE TEST

- FI3QAI -

```

11640 056462 004737 030370      JSR    PC,DELAY      ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
11641 056466 010105              MOV    R1,R5          ;COPY THE LINE NUMBER.
11642 056470 012702 004012      MOV    #0UFBAS,R2    ;PASS THE START OF THE DATA PATTERN TO TX.
11643 056474 012703 000277      MOV    #191.,R3      ;PASS THE LENGTH OF THE DATA PATTERN.
11644 056500 004737 030624      JSR    PC,DODMA      ;TRANSMIT THE DATA PATTERN.
11645 056504 103146              BCC    50$            ;EXIT IF ERROR FOUND DURING DMA TX.
11646
11647      ;+
11648      ; Wait for DMA to complete, then wait for the last character to arrive in
11649      ; the fifo.
11650 056506 005237 005466      INC    ERRNBR        ;SET ERROR NUMBER TO 5603.
11651 056512 012701 170454      MOV    #170454,R1    ;PASS TIME-OUT VALUE OF 300 MILLI SECS.
11652 056516 013702 002300      MOV    CSRA,R2       ;PASS THE ADDRESS OF THE CSR.
11653 056522 004737 037740      JSR    PC,WAIBIS     ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
11654 056526 103135              BCC    50$            ;BRANCH IF FIFO EMPTY, ABORT THE TEST.
11655 056530 012704 000005      MOV    #5,R4         ;PASS DELAY OF 5 MILLI SECS
11656 056534 004737 030370      JSR    PC,DELAY      ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.
11657
11658      ;+
11659      ; Transmit a null character which will cause an XOFF to be generated.
11660      ; -
11661 056540 005237 005466      INC    ERRNBR        ;SET ERROR NUMBER TO 5604.
11662 056544 010501              MOV    R5,R1          ;PASS THE LINE NUMBER.
11663 056546 012702 004012      MOV    #0UFBAS,R2    ;PASS THE START OF THE DATA PATTERN TO TX.
11664 056552 012703 000001      MOV    #1,R3         ;PASS THE NUMBER OF
11665 056556 004737 030624      JSR    PC,DODMA      ;TX A NULL CHARACTER TO CAUSE AN XOFF.
11666 056562 103117              BCC    50$            ;ABORT THE TEST IF ERROR FOUND DURING DMA TX.
11667
11668      ;+
11669      ; Wait for the XOFF to be received before continuing the test.
11670      ; -
11671 056564 005237 005466      INC    ERRNBR        ;SET ERROR NUMBER TO 5605.
11672 056570 012701 170012      MOV    #170012,R1    ;PASS TIME-OUT VALUE OF 10 MILLI SECS.
11673 056574 013702 002300      MOV    CSRA,R2       ;PASS THE ADDRESS OF THE CSR.
11674 056600 004737 037740      JSR    PC,WAIBIS     ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
11675 056604 103106              BCC    50$            ;IF NO TX_ACTION WAS RECEIVED, ABORT THE TEST.
11676 056606 012704 000005      MOV    #5,R4         ;PASS DELAY OF 5 MILLI SECS.
11677 056612 004737 030370      JSR    PC,DELAY      ;WAIT FOR XOFF TO GET INTO THE FIFO.
11678 056616 010577 123456      MOV    R5,0CSRA     ;SELECT THE LINE READY FOR TRANSMISSION.
11679
11680      ;+
11681      ; Read three characters, transmit one character until the first 192 characters
11682      ; have been read from the fifo, ie until the half level is reached.
11683      ; Then read the fifo until empty.
11684      ; Count all XOFF's that are detected.
11685      ; -
11685 056622 005005              CLR    R5             ;CLEAR THE TX FLAG.
11686 056624 005004              CLR    R4             ;CLEAR THE CHARACTER COUNTER.
11687 056626 012703 000300      MOV    #192.,R3      ;SET UP READ COUNTER FOR THE FIRST 192 CHARS.
11688
11689 056632 012700 000003      4$:    MOV    #3,R0          ;SET READ COUNTER.
11690 056636 012701 170005      6$:    MOV    #170005,R1    ;INDICATE TO TEST DATA.VALID BIT, TIME-OUT 5MS.
11691 056642 013702 002302      MOV    RBUFA,R2      ;INDICATE TO CHECK RECEIVE BUFFER REGISTER.
11692 056646 004737 037740      JSR    PC,WAIBIS     ;WAIT FOR RECEIVED CHAR OR TIME OUT.
11693 056652 103046              BCC    14$            ;EXIT LOOP IF TIME-OUT, FIFO EMPTY.
11694 056654 005300              DEC    R0             ;DECREMENT READ COUNTER.
11695 056656 005303              DEC    R3             ;DECREMENT CHAR COUNTER.
11696 056660 003002              BGT    8$             ;SKIP DISBL'G TX IF FIRST 192 CHARS NOT READ.

```

HARDWARE TEST

- FI3QAI -

```

11697 056662 052705 100000      BIS    #BIT15,R5      ;DISABLE ANY FURTHER TRANSMISSIONS.
11698
11699      ;+
11700      ; Check if the read character is a BMP code.
11701      ; If it is a BMP code save it on the queue to be reported later, and
11702      ; abort the test.
11703 056666 012737 012746 005466 8$:  MOV    #5606,ERRNBR  ;SET UP ERROR NUMBER EACH TIME AROUND LOOP.
11704 056674 004737 027256      JSR    PC,CHKBMP  ;CHECK IF CHARACTER IS A BMP CODE.
11705 056700 103446      BCS    16$      ;GO REPORT ERROR AND ABORT TEST IF BMP FOUND.
11706
11707      ;+
11708      ; Check for XOFF character. If one is found, count it.
11709      ; Transmit a null character until the first 192 chars have been read.
11710 056702 122702 000023      10$:  CMPB   #23,R2      ;CHECK IF THE RECEIVED CHARACTER WAS AN XOFF.
11711 056706 001001      BNE    12$      ;BRANCH IF CHARACTER WAS NOT AN XOFF.
11712 056710 005204      INC     R4      ;INCREMENT THE XOFF CHAR FOUND COUNTER.
11713
11714 056712 005700      12$:  TST     R0      ;CHECK READ COUNT, TO SEE IF A CHAR CAN BE TX.
11715 056714 001350      BNE    6$      ;BRANCH IF 3 CHARS HAVE NOT YET BEEN READ.
11716 056716 005705      TST     R5      ;CHECK THE TRANSMISSION ENABLED FLAG.
11717 056720 100744      BMI     4$      ;SKIP TRANSMITTING A CHARACTER IF TX DISABLED.
11718 056722 012777 100000 123352  MOV    #100000,0TXCHA ;TX A NULL CHARACTER.
11719 056730 010446      MOV    R4,-(SP) ;SAVE THE XOFF COUNT ON THE STACK.
11720
11721      ;+
11722      ; Wait for the character to be received before continuing the test.
11723 056732 005237 005466      INC     ERRNBR  ;SET ERROR NUMBER TO 5607.
11724 056736 012701 170012      MOV    #170012,R1 ;PASS TIME-OUT VALUE OF 10 MILLI SECS.
11725 056742 013702 002300      MOV    CSRA,R2  ;PASS THE ADDRESS OF THE CSR.
11726 056746 004737 037740      JSR    PC,WAIBIS ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
11727 056752 103023      BCC    50$      ;IF NO TX_ACTION WAS RECEIVED, ABORT THE TEST.
11728 056754 012704 000005      MOV    #5,R4      ;PASS DELAY OF 5 MILLI SECS.
11729 056760 004737 030370      JSR    PC,DELAY  ;WAIT FOR XOFF TO GET INTO THE FIFO.
11730 056764 012604      MOV    (SP)+,R4 ;RESTORE THE XOFF COUNT.
11731 056766 000721      BR      4$      ;GO RESET THE READ COUNT AND GET NEXT CHAR.
11732
11733      ;+
11734      ; Check if the correct number of XOFF's were found in the fifo
11735      ; Report error if count is incorrect.
11736
11737 056770 012737 012750 005466 14$:  MOV    #5608,ERRNBR ;SET ERROR NUMBER TO 5608.
11738 056776 020427 000077      CMP     R4,#63.  ;COMPARE THE EXPECTED AND ACTUAL XOFF COUNTS.
11739 057002 001411      BEQ     60$      ;EXIT TEST IF SUCCESS.
11740 057004 012737 024644 005472  MOV    #ER0503,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
11741 057012 012701 021617      MOV    #EM5402,R1 ;PASS THE MESSAGE TO BE REPORTED.
11742
11743 057016      16$:  ERROR  ;REPORT THE ERROR "FIFO BAD, ALARM SIGNAL DEFECTIVE".
11744 057016 104460      ;>>>> ERROR <<<<<.
11745 057020 000402      BR      60$      TRAP    C#ERROR
11746 057022 004737 036106      50$:  JSR    PC,TSABRT  ;REPORT TEST ABORTED. ERROR # INDICATES FAULT.
11747 057026 005037 002362      60$:  CLR     CTRLCF  ;INDICATE THAT WE ARE NOT WITHIN A TEST.
11748
11749 057032      ENDTST
057032
057032 104401      L10074:
                                TRAP    C#ETST

```

- FIHAVL -

```

.SBttl  hardware test          - FIHAVL -
;*****
;
;          - FIFO HALF LEVEL ACTIVE/INACTIVE TEST -
;
;
; This test checks that the DUT's fifo half level alarm system
; becomes active and inactive at the correct levels.
; Any BMP code found will invalidate the test and cause it to be aborted.
; However the BMP code will be placed on the BMP code queue, to be
; reported later.
; The characters are transmitted on the first available active line, in
; internal loopback mode.
;
;*****

BGNTST

                                T34::
TNUM == TNUM + 1                ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV    #TNUM,TSTNUM             ;SET UP THE TEST NUMBER. (57)
MOV    #-1,CTRLCF               ;INDICATE THAT WE ARE IN A TEST.
MOV    #1,ERRTYP                ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV    #5701,ERRNBR             ;SET ERROR NUMBER TO 5701.
MOV    #EM5701,ERRMSG           ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
MOV    #ER0503,ERRBLK          ;SELECT THE 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 >>>> 5701 <<<<.
;
JSR    PC,CLNRST                ;RESET THE DHV11 M, REPORT ANY ERRORS FOUND.
BCS    2$                       ;SKIP EXITING TEST A SUCCESSFUL RESET.
JMP     60$                     ;EXIT THIS TEST.
2$:
;
; Find an active line on which to perform the test.
;
JSR    PC,FINACT                ;FIND AN ACTIVE LINE.
BCC    60$                     ;EXIT IF NO ACTIVE LINES AVAILABLE.
;
; Initialize the 256 byte data pattern.
; Ensure the data pattern is free from XON's or XOFF's to prevent errors.
; Note: the first two characters and the last two characters will be the same.
;
JSR    PC,INDTPX                ;INITIALISE THE DATA PATTERN.
;
; Fill the fifo by transmitting 225 chars (ie 225 + 31 XOFF's).
; Transmit data pattern using DMA, on a single channel
; at 38.4k baud, 8 bits per character, odd parity, 2 stop bits.
;
;
; Set internal loopback, enable IAUTO and receiver on the selected line
; Transmit the 225 characters on the first available active line.
;
INC     ERRNBR                  ;SET ERROR NUMBER TO 5702.
JSR     PC,SETPAR               ;SET UP PARAMETERS FOR TRANSMISSION.
MOV     #225,R0                 ;PASS LENGTH OF DATA PATTERN.
JSR     PC,TXDATP               ;TRANSMIT DATA PATTERN.
BCC     50$                     ;EXIT IF ERROR FOUND DURING TX.

```

HARDWARE TEST

- FIHAVL -

```

11807 057146 010105      MOV    R1,R5      ;COPY THE LINE NUMBER.
11808
11809      ;*
11810      ; Wait for DMA to complete, then wait for the last character to arrive in
11811      ; the fifo.
11812 057150 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5703.
11813 057154 004737 040014      JSR    PC,WAITTX    ;WAIT FOR TRANSMISSION TO COMPLETE.
11814 057160 103142      BCC    50$      ;GO REPORT ERROR IF TX FAILED TO COMPLETE.
11815
11816      ;*
11817      ; Read the first 130 characters from the fifo if any XON's are found
11818      ; report the error. If any BMP codes are found then save them on the queue
11819      ; and abort the test.
11820 057162 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5704.
11821 057166 012700 000202      MOV    #130,R0      ;PASS THE NUMBER OF CHARS TO READ.
11822 057172 004737 034056      JSR    PC,READBX    ;READ THE FIRST 130 CHARS FROM THE FIFO.
11823 057176 103133      BCC    50$      ;GO REPORT ERROR IF BMP CODE FOUND.
11824 057200 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5705.
11825 057204 005701      TST    R1      ;CHECK IF AN XON WAS FOUND.
11826 057206 001125      BNE    40$      ;GO REPORT ERROR IF AN XON WAS FOUND.
11827
11828      ;*
11829      ; Transmit a null character.
11830
11831 057210 010577 123064      MOV    R5,@CSRA      ;SELECT THE LINE READY FOR TRANSMISSION.
11832 057214 012777 100000 123060      MOV    #100000,@TXCHA ;TRANSMIT A NULL CHARACTER.
11833 057222 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5706.
11834 057226 004737 040014      JSR    PC,WAITTX    ;WAIT FOR TX TO COMPLETE.
11835 057232 103115      BCC    50$      ;GO REPORT ERROR IF TX DID NOT COMPLETE.
11836
11837      ;*
11838      ; Read three characters, to cause the XON to be generated.
11839 057234 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5707.
11840 057240 012700 000003      MOV    #3,R0      ;SET THE READ COUNT TO 3.
11841 057244 004737 034056      JSR    PC,READBX    ;READ 3 CHARACTERS FROM THE FIFO.
11842 057250 103106      BCC    50$      ;GO REPORT ERROR IF FIFO EMPTY.
11843 057252 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5708.
11844 057256 005701      TST    R1      ;CHECK IF AN XON WAS FOUND.
11845 057260 001100      BNE    40$      ;GO REPORT ERROR IF AN XON WAS FOUND.
11846
11847      ;*
11848      ; Transmit 62 characters to bracket the XON and fill the fifo with 191 chare.
11849 057262 012700 000076      MOV    #62,R0      ;PASS LENGTH OF DATA PATTERN.
11850 057266 010501      MOV    R5,R1      ;PASS THE LINE NUMBER.
11851 057270 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5709.
11852 057274 004737 036220      JSR    PC,TXDATP    ;TRANSMIT DATA PATTERN.
11853 057300 103072      BCC    50$      ;EXIT IF ERROR FOUND DURING TX.
11854
11855      ;*
11856      ; Wait for DMA to complete, then wait for the last character to arrive in
11857      ; the fifo.
11858
11859 057302 005237 005466      INC    ERRNBR      ;SET ERROR NUMBER TO 5710.
11860 057306 004737 040014      JSR    PC,WAITTX    ;WAIT FOR TX TO COMPLETE.
11861 057312 103065      BCC    50$      ;GO REPORT ERROR IF TX FAILED TO COMPLETE.
11862
11863      ;*
11863      ; Read the first 126 characters.

```

HARDWARE TEST

- FIHAVL -

```

11864      ; Read the next 4 characters and check if they are in the following order
11865      ; NULL, XOFF, XON, NULL.
11866      :-
11867 057314 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5711.
11868 057320 012700 000176      MOV      #126.,R0      ;SET UP READ COUNTER.
11869 057324 004737 034056      JSR      PC,READ8X      ;READ THE FIRST 126 CHARS.
11870 057330 103056      BCC      50$      ;GO REPORT THE ERROR IF FIFO EMPTY.
11871 057332 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5712.
11872 057336 005701      TST      R1      ;CHECK IF AN XON WAS FOUND.
11873 057340 001050      BNE      40$      ;GO REPORT ERROR IF AN XON WAS FOUND.
11874 057342 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5713.
11875 057346 012701 021617      MOV      #EM5402,R1      ;PASS THE MESSAGE TO BE REPORTED.
11876 057352 013703 002302      MOV      RBUFA,R3      ;GET THE RECEIVER BUFFER ADDRESS.
11877 057356 011302      MOV      (R3),R2      ;READ THE NULL CHARACTER FROM THE FIFO.
11878 057360 120227 000000      CMPB     R2,#000      ;CHECK IF IT IS A NULL CHARACTER.
11879 057364 001036      BNE      40$      ;GO REPORT THE ERROR IF NOT THE SAME.
11880 057366 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5714.
11881 057372 011302      MOV      (R3),R2      ;READ THE XOFF FROM THE FIFO.
11882 057374 120227 000023      CMPB     R2,#23      ;CHECK IF THE READ CHAR IS AN XOFF.
11883 057400 001030      BNE      40$      ;GO REPORT THE ERROR IF NOT THE SAME.
11884 057402 011302      MOV      (R3),R2      ;READ THE XON FROM THE FIFO.
11885 057404 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5715.
11886 057410 120227 000021      CMPB     R2,#21      ;CHECK IF THE READ CHARACTER IS AN XON.
11887 057414 001022      BNE      40$      ;GO REPORT THE ERROR IF NOT THE SAME.
11888 057416 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5716.
11889 057422 011302      MOV      (R3),R2      ;READ THE NULL CHARACTER FROM THE FIFO.
11890 057424 120227 000000      CMPB     R2,#000      ;CHECK IF IT IS A NULL CHARACTER.
11891 057430 001014      BNE      40$      ;GO REPORT THE ERROR IF NOT THE SAME.
11892
11893
11894      ;+
11895      ; Read the remaining characters from the fifo.
11896      :-
11896 057432 012700 000075      6$:      MOV      #61.,R0      ;SET UP READ COUNTER.
11897 057436 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5717.
11898 057442 004737 034056      JSR      PC,READ8X      ;READ THE FIRST 61 CHARS.
11899 057446 103007      BCC      50$      ;GO REPORT THE ERROR IF FIFO EMPTY.
11900 057450 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 5718.
11901 057454 005701      TST      R1      ;CHECK IF AN XON WAS FOUND.
11902 057456 001001      BNE      40$      ;GO REPORT ERROR IF AN XON WAS FOUND.
11903 057460 000404      BR       60$      ;EXIT THE TEST.
11904 057462      40$:      ERROR      ;
11905 057464 000402      BR       60$      ;EXIT THE TEST.
11906
11907 057466 004737 036106      50$:      JSR      PC,TSABRT      ;REPORT TEST ABORTED. ERROR # INDICATES FAULT.
11908 057472 005037 002362      60$:      CLR      CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
11909
11910 057476      ENDTST
11910 057476
11910 057476 104401      L10075.      TRAP      C0ETST

```

HARDWARE TEST

- BREAKB -

```

11912
11913
11914
11915
11916
11917
11918
11919
11920
11921
11922
11923 057500
      057500
11924
11925 057500 012737 177777 002362
11926      000043
11927 057506 012737 000043 002364
11928 057514 012737 000001 005464
11929 057522 012737 014401 005466
11930 057530 012737 022031 005470
11931
11932
11933
11934
11935
11936 057536 004737 030172
11937 057542 103165
11938
11939
11940
11941
11942
11943 057544 004737 036542
11944 057550 004737 035546
11945 057554 012705 000377
11946 057560 012700 000200
11947 057564 004737 040250
11948 057570 012704 000012
11949 057574 004737 030370
11950
11951
11952
11953
11954 057600 012700 156430
11955 057604 004737 040324
11956
11957
11958
11959 057610 013705 002272
11960 057614 004737 036446
11961
11962
11963
11964
11965 057620 005237 005466
11966 057624 004737 033004
11967 057630 103132

```

.SBTTL HARDWARE TEST - BREAKB -

```

;*****
;*                                     - BREAK generation test -
;*   This test verifies that all serial transmit lines can generate a break
;*   by setting the brk bit in the associated LNCRTL register.
;*   Use of the internal loopback feature of the DUARTS is made to minimise
;*   any external effects caused on the serial lines by this test.
;*   Framing error detection is used to indicate the presence of a break,
;*   by setting the appropriate bit in the RBUF register.
;*****

```

BGNTST

T35::

```

;*****
;*                                     - BREAK generation test -
;*   This test verifies that all serial transmit lines can generate a break
;*   by setting the brk bit in the associated LNCRTL register.
;*   Use of the internal loopback feature of the DUARTS is made to minimise
;*   any external effects caused on the serial lines by this test.
;*   Framing error detection is used to indicate the presence of a break,
;*   by setting the appropriate bit in the RBUF register.
;*****

; 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 >>>> 6401 <<<<.
;
JSR    PC,CLNRST    ;RESET THE DMV11-M, REPORT ANY ERRORS FOUND.
BCC    60$          ;EXIT TEST IF FATAL ERROR FOUND.

; Set up Device Under Test (DUT) to:
;   Disable transmission and reception interrupts.
;   Delay for 10 milli-seconds to allow time to clear any breaks.
;
JSR    PC,IXIE0      ;DISABLE TRANSMISSION INTERRUPTS.
JSR    PC,RXIE0      ;DISABLE RECEPTION INTERRUPTS.
MOV     #MAPLNS,R5    ;PASS ACTIVE LINE BIT MAP.
MOV     #200,R0       ;PASS INTERNAL LOOPBACK MODE.
JSR     PC,WTWLNLC    ;SELECT INTERNAL LOOPBACK,DISABLE DMA.
MOV     #10,R4        ;PASS DELAY TIME OF 10 MILLI SECONDS.
JSR     PC,DELAY      ;DELAY TO ALLOW ANY BREAKS TO BE CLEARED.

; Set up transmission and reception parameters for all lines.
; 9600 baud,8 char,1 stopb't,no par ty.
;
MOV     #156430,R0    ;SET UP BAUD RATE,ETC.
JSR     PC,WTWLPR     ;SET COMMUNICATION PARAMETERS ON ALL LINES.

; Enable transmitters on all active lines.
;
MOV     ACTLNS,R5     ;PASS ACTIVE LINE BIT MAP.
JSR     PC,TXENB1     ;ENABLE TRANSMISSIONS ON ALL LINES.

; Purge the FIFO of any unwanted characters.
; This routine reports errors with numbers >>>> 6402 thru 6404 <<<<.
;
INC     ERRNBR        ;SET ERROR NUMBER TO 6402.
JSR     PC,PUFIFR     ;PURGE FIFO.
BCC     60$          ;ABORT TEST IF FIFO WILL NOT CLEAR.

```


HARDWARE TEST

- BREAKB

```

11968      ;
11969      ; Verify break generation on individual lines.
11970      ; Clear breaks on all lines.
11971      ; Delay for 10 milli-seconds to allow time for any breaks to be cleared.
11972      ; Select line, set break bit in LNCTRL register.
11973      ; Test for a character in the FIFO with frame error.
11974      ;
11975 057632 005002 2$: CLR R2 ;CLEAR LINE COUNTER.
11976 057634 012703 000001 MOV #1,R3 ;SET UP ACTIVE LINE BIT MASK.
11977 057640 030337 002272 4$: BIT R3,ACTLNS ;CHECK IF THIS LINE IS ACTIVE.
11978 057644 001434 BEQ B$ ;GO SELECT NEXT LINE IF THIS ONE IS INACTIVE.
11979 057646 012700 000200 MOV #200,R0 ;SET UP PARAMETER TO CLEAR BREAK BITS.
11980 057652 004737 040250 JSR PC,WTWLN ;CLEAR BREAK BIT, RESELECT INTERNAL LOOPBACK.
11981 057656 012704 000012 MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI SECONDS.
11982 057662 004737 030370 JSR PC,DELAY ;DELAY TO ALLOW BREAKS TO BE CLEARED.
11983      ;
11984      ; Set break bit on selected line.
11985      ; Set up parameters to test for the frame error bit set in RBUF.
11986      ; Time-out = 5 milli seconds.
11987      ; Call routine to check for condition found.
11988      ;
11989 057666 010305 6$: MOV R3,R5 ;COPY ACTIVE LINE BIT MASK.
11990 057670 012700 000214 MOV #214,R0 ;SET BREAK, RESELECT LOOPBACK, ENABLE RECEPTION.
11991 057674 004737 040250 JSR PC,WTWLN ;SET BREAK ON SELECTED LINE.
11992      ;
11993      ; Delay for 5 ms to allow time for break to be generated and received.
11994      ; Verify reception of a character with frame error bit set.
11995      ;
11996 057700 012704 000005 MOV #5.,R4 ;SET DELAY VALUE TO 5 MILLI SECS.
11997 057704 004737 030370 JSR PC,DELAY ;ALLOW TIME FOR CHARACTER RECEPTION.
11998 057710 017700 122366 MOV @RBUFA,R0 ;GET CHARACTER FROM RBUF REGISTER.
11999 057714 032700 020000 BIT #BIT13,R0 ;CHECK FOR FRAME ERROR BIT.
12000 057720 001006 BNE B$ ;SKIP ERROR REPORT IF SET.
12001 057722 012701 022060 MOV #EM6402,R1 ;SELECT MESSAGE TO BE PRINTED.
12002      ;REPORT ERROR BREAK NOT RECEIVED ON LINE #nn"
12003 057726      ERDF 6405,EM6401,ER6401 ; >>>> ERROR #6405 <<<<.
12004      057726 104455 TRAP C$ERDF
12005      057730 014405 .WORD 6405
12006      057732 022031 .WORD EM6401
12007      057734 025204 .WORD ER6401
12008      ;
12009      ; Verify break generation on all lines simultaneously.
12010      ; Clear breaks on all lines.
12011      ; Delay for 10 milli-seconds to allow time for any breaks to be cleared
12012      ; Purge the FIFO.
12013      ; Set break bit in LNCTRL registers on all active lines.
12014      ; Test for characters in the FIFO with frame error.
12015      ;
12016 057750 012705 000377 MOV #MAPLNS,R5 ;SET UP LINE TO CLEAR BREAKS ON.
12017 057754 012700 000200 MOV #200,R0 ;SET UP PARAMETER TO CLEAR BREAK BITS.
12018 057760 004737 040250 JSR PC,WTWLN ;CLEAR BREAK BIT, RESELECT INTERNAL LOOPBACK.
12019 057764 012704 000012 MOV #10.,R4 ;PASS DELAY TIME OF 10 MILLI SECONDS.
12020 057770 004737 030370 JSR PC,DELAY ;DELAY TO ALLOW BREAKS TO BE CLEARED.

```

HARDWARE TEST

BREAKB

```

12021
12022
12023
12024 057774 004737 032722
12025 060000 103044
12026
12027
12028
12029
12030 060002 013705 002272
12031 060006 012700 000214
12032 060012 004737 040250
12033
12034
12035
12036
12037 060016 012704 000012
12038 060022 004737 030370
12039 060026 010502
12040 060030 004737 031674
12041 060034 017701 122242
12042 060040 100011
12043 060042 032701 020000
12044 060046 001406
12045 060050 000301
12046 060052 042701 177400
12047 060056 004737 031646
12048 060062 040005
12049 060064 005302
12050 060066 001362
12051 060070 005705
12052 060072 001411
12053 060074 012701 022060
12054
12055 060100
      060100 104455
      060102 014406
      060104 022031
      060106 025204
12056 060110 000402
12057
12058 060112 004737 036106
12059 060116 005037 002362
12060 060122
      060122
      060122 104401

; Purge the FIFO of unwanted characters.
JSR PC,PUFIFO ;PURGE FIFO.
BCC 50$ ;GO REPORT ERROR IF FAILED TO CLEAN_OUT FIFO.

; Set up parameters for setting the break bit on all active lines.
; Then call routine to do it.
10$: MOV ACTLNS,R5 ;SET UP ACTIVE LINE BIT MASK.
      MOV #214,R0 ;SET BREAK,RESELECT LOOPBACK,ENABLE RECEPTION.
      JSR PC,WTWLCN ;SET BREAK ON SELECTED LINES.

; Delay for 10 milli seconds, to allow time for reception.
; Test for characters in fifo with frame error bit set.
      MOV #10,R4 ;SET DELAY VALUE TO 10 MILLI SECS.
      JSR PC,DELAY ;ALLOW TIME FOR CHARACTER RECEPTION.
      MOV R5,R2 ;COPY ACTIVE LINE BIT MAP.
      JSR PC,MAPCNT ;COUNT THE NUMBER OF LINES AVAILABLE.
12$: MOV RBUF,R1 ;GET CHARACTER FROM RBUF REGISTER.
      BPL 14$ ;BRANCH IF DATA_VALID NOT SET.
      BIT #BIT13,R1 ;CHECK FOR FRAME ERROR BIT.
      BEQ 14$ ;DO NOT CLR FLG FOR THIS LINE IF FRAME BIT CLR.
      SWAB R1 ;GET LINE NUMBER IN LOW BYTE.
      BIC #177400,R1 ;CLEAR EVERYTHING BUT THE LINE NUMBER.
      JSR PC,LINBIT ;CALC BIT MASK FROM LINE NUMBER.
      BIC R0,R5 ;CLEAR LINE FLAG.
14$: DEC R2 ;DECREMENT THE LINE NUMBER COUNTER.
      BNE 12$ ;LOOP TO GET THE NEXT CHARACTER.
      TST R5 ;CHECK IF ANY BREAKS NOT RECEIVED.
      BEQ 60$ ;EXIT TEST IF ALL CLE ?
      MOV #EM6402,R1 ;SELECT MESSAGE TO BE PRINTED.
;REPORT ERROR "BREAK NOT RECEIVED ON LINE #nn".
ERRDF 6406,EM6401,ER6401; >>>> ERROR #6407 <<<<.
                                TRAP C$ERDF
                                .WORD 6406
                                .WORD EM6401
                                .WORD ER6401

BR 60$ ;EXIT THE TEST.

50$: JSR PC,TSABRT ;ABORT THE TEST.
60$: CLR CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.
      ENDTST

L10076: TRAP C$ETST

```


HARDWARE TEST

- NORERR -

```

12062 .SBTTL HARDWARE TEST - NORERR -
12063 ;*****
12064 ;* - NO OVERRUN ERROR TEST -
12065 ;*
12066 ;* This test verifies that the DUT will not report data overrun
12067 ;* errors when they do not occur.
12068 ;* This test puts 256 characters in the DUT FIFO plus 4 in each active
12069 ;* UART and verifies that no overrun errors are reported.
12070 ;* Any BMP code found will invalidate the test and cause it to be aborted.
12071 ;* However the BMP code will be placed on the BMP code queue, to be
12072 ;* reported later.
12073 ;*
12074 ;*****
12075
12076 060124 BGNTST
12077 060124
12077 000044 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
12078 060124 012737 000044 002364 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (66)
12079 060132 012737 177777 002362 MOV #1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
12080 060140 012737 000001 005464 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
12081 060146 012737 014711 005466 MOV #6601,ERRNBR ;SET ERROR NUMBER TO 6601.
12082 060154 012737 022121 005470 MOV #EM6601,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
12083 ;*
12084 ; Reset the DUT to a known state, remove the status codes from the fifo.
12085 ; Clear TX and RX interrupt enable bits in the CSR.
12086 ; This subroutine reports error >>>> 6601 <<<<.
12087 ;-
12088 060162 004737 030172 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
12089 060166 103402 BCS .+6 ;SKIP EXIT OF TEST IF NO FATAL ERROR FOUND.
12090 060170 000137 060616 JMP 60$ ;EXIT THE TEST, FATAL ERROR WAS FOUND.
12091 ;*
12092 ; Find an active line on which to perform the test.
12093 ; Initialize the 256 byte data pattern.
12094 ;
12095 060174 004737 030772 JSR PC,FINACT ;FIND AN ACTIVE LINE.
12096 060200 103402 BCS .+6 ;SKIP EXIT OF TEST IF NO FATAL ERROR FOUND.
12097 060202 000137 060616 JMP 60$ ;EXIT THE TEST, FATAL ERROR WAS FOUND.
12098 060206 004737 031342 JSR PC,INDATP ;INITIALISE DATA PATTERN.
12099 ;*
12100 ; Transmit a 265 character data pattern using DMA, on a single channel
12101 ; at 38.4k baud, 8 bits per character, odd parity, 2 stop bits.
12102 ;-
12103 ;*
12104 ; Set internal loopback on the selected line.
12105 ; Transmit the data pattern on the first available active line.
12106 ;-
12107 060212 005237 005466 INC ERRNBR ;SET THE ERROR REPORT NUMBER TO 6602.
12108 060216 012700 000204 MOV #204,R0 ;PASS PARAMETER FOR INTERNAL LOPBCK,ENABLE RX.
12109 060222 004737 040250 JSR PC,WTWLCN ;INITIALISE THE LINE CONTROL REGISTER.
12110 060226 012700 177670 MOV #177670,R0 ;PASS THE LPR CONTENTS.
12111 060232 004737 040324 JSR PC,WTWLPN ;SET THE LPR CONTENTS TO 38.4K BAUD.
12112 060236 012704 000012 MOV #10,R4 ;PASS DELAY TIME OF 10 MILLI SECONDS.
12113 060242 004737 030370 JSR PC,DELAY ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
12114 060246 012702 004012 MOV #8UFBAS,R2 ;PASS THE START OF THE DATA PATTERN TO TX.
12115 060252 012703 000400 MOV #8UFMID BUFBAS,R3 ;PASS THE LENGTH OF THE DATA PATTERN.
12116 060256 004737 030624 JSR PC,DODMA ;TRANSMIT THE DATA PATTERN.
12117 060262 103153 BCC 50$ ;EXIT IF ERROR FOUND DURING DMA TX.

```

HARDWARE TEST

- NORERR -

```

12118
12119
12120
12121
12122 060264 005237 005466
12123 060270 012701 170536
12124 060274 013702 002300
12125 060300 004737 037740
12126 060304 103142
12127 060306 012704 000005
12128 060312 004737 030370
12129
12130
12131
12132
12133 060316 013705 002272
12134 060322 012700 000204
12135 060326 004737 040250
12136 060332 012700 177670
12137 060336 004737 040324
12138 060342 012704 000012
12139 060346 004737 030370
12140
12141 060352 012702 004012
12142 060356 012703 000004
12143 060362 005001
12144 060364 005237 005466
12145 060370 010100
12146 060372 006300
12147 060374 036037 002534 002272
12148 060402 001403
12149 060404 004737 030624
12150 060410 103100
12151 060412 005201
12152 060414 020127 000010
12153 060420 002763
12154
12155 060422 005237 005466
12156 060426 012701 170040
12157 060432 013702 002300
12158 060436 004737 037740
12159 060442 103063
12160 060444 012704 000005
12161 060450 004737 030370
12162
12163
12164
12165
12166 060454 013702 002272
12167 060460 004737 031674
12168 060464 006302
12169 060466 006302
12170 060470 012705 000400
12171 060474 060205
12172 060476 005004
12173 060500 012737 014716 005466
12174 060506 017702 121570

```

```

;+
; Wait for DMA to complete, then wait for the last character to arrive in
; the fifo.
;-
      INC      ERRNBR      ;SET ERROR NUMBER TO 6603.
      MOV      #170536,R1  ;PASS TIME-OUT VALUE OF 350 MILLI SECS.
      MOV      CSRA,R2     ;PASS THE ADDRESS OF THE CSR.
      JSR      PC,WAIBIS    ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
      BCC      50$         ;ABORT THE TEST IF TIME-OUT ON DMA COMPLETION.
      MOV      #5,R4       ;PASS DELAY OF 5 MILLI SECS.
      JSR      PC,DELAY     ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.

;+
; Transmit 4 characters on each active line.
;-
      MOV      ACTLNS,R5   ;ALTER PARAMETERS FOR ALL ACTIVE LINES.
      MOV      #204,R0     ;PASS PARAMETER FOR INTERNAL LOPBCK,ENABLE RX.
      JSR      PC,WTWLCN   ;INITIALISE THE LINE CONTROL REGISTER.
      MOV      #177670,R0  ;PASS THE LPR CONTENTS.
      JSR      PC,WTWLPR   ;SET THE LPR CONTENTS TO 38.4K BAUD.
      MOV      #10.,R4     ;PASS DELAY TIME OF 10 MILLI SECONDS.
      JSR      PC,DELAY     ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.

      MOV      #BUFBAS,R2  ;PASS THE START OF THE DATA PATTERN TO TX.
      MOV      #4,R3       ;PASS THE LENGTH OF THE DATA PATTERN.
      CLR      R1          ;CLEAR THE LINE COUNTER.
      INC      ERRNBR      ;SET ERROR NUMBER TO 6604.
      MOV      R1,R0
      ASL      R0          ;CALCULATE THE LINE OFFSET FROM THE LINE #.
      BIT      BITTBL(R0),ACTLNS ;TEST FOR THIS LINE BEING ACTIVE.
      BEQ      4$         ;SKIP THE TX ON THIS LINE IF IT IS NOT ACTIVE.
      JSR      PC,DODMA    ;TRANSMIT THE 5 CHAR DATA PATTERN.
      BCC      50$         ;ABORT IF ERROR FOUND DURING DMA TX.
      INC      R1          ;INCREMENT THE LINE COUNTER.
      CMP      R1,#NUMLNS  ;TEST FOR ALL POSSIBLE LINES HANDLED
      BLT      2$         ;LOOP IF NOT ALL LINES HANDLED.

      INC      ERRNBR      ;SET ERROR NUMBER TO 6605.
      MOV      #170040,R1  ;PASS TIME-OUT VALUE OF 32 MILLI SECS.
      MOV      CSRA,R2     ;PASS THE ADDRESS OF THE CSR.
      JSR      PC,WAIBIS    ;WAIT FOR A DMA TO COMPLETE, TX_ACTION SET.
      BCC      50$         ;ABORT THE TEST IF TIME-OUT ON DMA COMPLETION.
      MOV      #5,R4       ;PASS DELAY OF 5 MILLI SECS.
      JSR      PC,DELAY     ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.

;+
; Read the FIFO checking for overrun errors. Report errors if found.
; Abort the test if a BMP code was found in the fifo.
;-
      MOV      ACTLNS,R2   ;GET THE NUMBER OF ACTIVE LINES.
      JSR      PC,MAPCNT
      ASL      R2
      MOV      #256.,R5    ;MULTIPLY NUMBER OF ACTIVE LINES BY 4.
      ADD      R2,R5
      CLR      R4          ;CALCULATE NUMBER OF CHARACTERS TO RX.
      MOV      #6606.,ERRNBR ;SET UP ERROR NUMBER EACH TIME AROUND LOOP.
      MOV      @RBUFA,R2   ;READ A CHARACTER FROM THE FIFO.

```

HARDWARE TEST

- NORERR -

```

12175 060512 100032      BPL      10$      ;EXIT THE READ LOOP IF THE FIFO IS EMPTY.
12176
12177      ;+
12178      ; Check if the read character is a BMP code.
12179      ; If it is a BMP code save it on the queue to be reported later, and
12180      ; abort the test.
12181 060514 004737 027256      JSR      PC,CHKBMP      ;CHECK IF CHARACTER IS A BMP CODE.
12182 060520 103002      BCC      8$      ;BRANCH IF NOT A BMP CODE.
12183 060522      ERROR      ;          >>>> ERROR #6606 <<<<<.
12184 060522 104460      TRAP      C$ERROR
12185 060524 000434      BR      60$      ;EXIT THIS TEST.
12186 060526 005237 005466      8$:      INC      ERRNBR      ;SET ERROR NUMBER TO 6607.
12187 060532 005204      INC      R4      ;COUNT THIS CHARACTER.
12188 060534 020405      CMP      R4,R5      ;COMPARE # OF CHARS WITH MAX # OF CHARS.
12189 060536 003025      BGT      50$      ;ABORT TEST IF TOO MANY VALID CHARS READ.
12190 060540 032702 040000      BIT      #BIT14,R2      ;TEST THE OVERRUN BIT OF THE READ CHAR.
12191 060544 001755      BEQ      6$      ;LOOP TO READ THE NEXT CHAR IF NO ERROR.
12192 060546 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 6608.
12193 060552 012737 025302 005472      MCV      #ER7801,ERRBLK      ;SELECT THE CORRECT ERROR REPORTING ROUTINE.
12194 060560 012701 022147      MOV      #EM6602,R1      ;PASS THE MESSAGE TO BE REPORTED.
12195 060564 010203      MOV      R2,R3
12196 060566 000303      SWAB      R3
12197 060570 042703 177760      BIC      #177760,R3      ;GET FAILING LINE NUMBER.
12198      ;REPORT "OVERRUN ERROR REPORTED WHEN NONE FORCED, ON LINE nn ..."
12199 060574      ERROR      ;          >>>> ERROR #6608 <<<<<.
12200 060576 104460      TRAP      C$ERROR
12201 060576 000740      BR      6$      ;LOOP TO READ THE NEXT CHAR.
12202 060600 012737 014721 005466 10$:      MOV      #6609,ERRNBR      ;SET ERROR NUMBER TO 6609.
12203 060606 020405      CMP      R4,R5      ;COMPARE NUMBER OF CHARS READ WITH EXPECTED.
12204 060610 001402      BEQ      60$      ;EXIT TEST WITHOUT ABORT IF CORRECT # OF CHARS.
12205
12206 060612 004737 036106      50$:      JSR      PC,TSABRT      ;ABORT THE TEST, NON-RELATED TEST ERROR FOUND.
12207 060616 005037 002362      60$:      CLR      CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
12208 060622      ENDTST
12209 060622
12210 060622 104401      L10077:
12211      TRAP      C$ETST

```

HARDWARE TEST

- ORERR -

```

12210
12211
12212
12213
12214
12215
12216
12217
12218
12219
12220
12221
12222
12223
12224 060624
      060624
12225      000045
12226 060624 012737 000045 002364
12227 060632 012737 177777 002362
12228 060640 012737 000001 005464
12229 060646 012737 015055 005466
12230 060654 012737 022221 005470
12231
12232
12233
12234
12235
12236 060662 004737 030172
12237 060666 103402
12238 060670 000137 061544
12239
12240
12241
12242
12243 060674 004737 030772
12244 060700 103402
12245 060702 000137 061544
12246 060706 004737 031342
12247
12248
12249
12250
12251
12252
12253
12254
12255 060712 005237 005466
12256 060716 012700 000204
12257 060722 004737 040250
12258 060726 012700 177670
12259 060732 004737 040324
12260 060736 012704 000012
12261 060742 004737 030370
12262 060746 012702 004012
12263 060752 012703 000400
12264 060756 004737 030624
12265 060762 103402

```

```

.SBTTL  HARDWARE TEST          - ORERR -
;*****
;                                - OVERRUN ERROR TEST -
;
;      This test verifies that the DUT will report data overrun errors when
;      they occur.
;      This test puts 256 characters in the DUT FIFO plus 5 in each active
;      UART and verifies that overrun errors are reported on all active lines.
;      Any BMP code found will invalidate the test and cause it to be aborted.
;      However the BMP code will be placed on the BMP code queue, to be
;      reported later.
;- *****

      BGNTST

      T37::
      TNUM == TNUM + 1          ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV    #TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (67)
      MOV    #-1,CTRLCF        ;INDICATE THAT WE ARE IN A TEST.
      MOV    #1,ERRTYP         ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
      MOV    #6701,ERRNBR      ;SET ERROR NUMBER TO 6701.
      MOV    #EM6701,ERRMSG    ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
;+
; 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 >>>> 6701 <<<<.
;-
      JSR    PC,CLNRST         ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
      BCS    .+6               ;SKIP EXIT OF TEST IF NO FATAL ERROR FOUND.
      JMP    60$               ;EXIT THE TEST, FATAL ERROR WAS FOUND.
;+
; Find an active line on which to perform the test.
; Initialize the 256 byte data pattern.
;-
      JSR    PC,FINACT         ;FIND AN ACTIVE LINE.
      BCS    .+6               ;IF ACTIVE LINE IS FOUND, DON'T ABORT TEST.
      JMP    60$               ;ABORT THE TEST, NO ACTIVE LINES WERE FOUND.
      JSR    PC,INDATP         ;INITIALISE DATA PATTERN.
;+
; Transmit a 265 character data pattern using DMA, on a single channel
; at 38.4k baud, 8 bits per character, odd parity, 2 stop bits.
;
;+
; Set internal loopback on the selected line.
; Transmit the data pattern on the first available active line.
;
      INC    ERRNBR            ;SET ERROR NUMBER TO 6702.
      MOV    #204,R0           ;PASS PARAMETER FOR INTERNAL LOPBCK,ENABLE RX.
      JSR    PC,WTWLNCR        ;INITIALISE THE LINE CONTROL REGISTER.
      MOV    #177670,R0        ;PASS THE LPR CONTENTS.
      JSR    PC,WTWLPRL        ;SET THE LPR CONTENTS TO 38.4K BAUD.
      MOV    #10.,R4           ;PASS DELAY TIME OF 10 MILLI SECONDS.
      JSR    PC,DELAY          ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
      MOV    #BUFBAS,R2        ;PASS THE START OF THE DATA PATTERN TO TX.
      MOV    #BUFMID-BUFBAS,R3 ;PASS THE LENGTH OF THE DATA PATTERN.
      JSR    PC,DODMA          ;TRANSMIT THE DATA PATTERN.
      BCS    .+6               ;IF NO ERROR FOUND DURING DMA TX, DON'T ABORT.

```

HARDWARE TEST

- ORERR -

```

12266 060764 000137 061540      JMP      50$      ;ABORT TEST, ERROR FOUND DURING DMA TX.
12267                               ;+
12268                               ; Wait for DMA to complete, then wait for the last character to arrive in
12269                               ; the fifo.
12270                               ; -
12271 060770 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 6703.
12272 060774 012701 170536      MOV      #170536,R1      ;PASS TIME-OUT VALUE OF 350 MILLI SECS.
12273 061000 013702 002300      MOV      CSRA,R2      ;PASS THE ADDRESS OF THE CSR.
12274 061004 004737 037740      JSR      PC,WAIBIS      ;WAIT FOR DMA TO COMPLETE, TX_ACTION SET.
12275 061010 103402              BCS      .+6      ;IF NO TIME-OUT ON DMA COMPLETION, DON'T ABORT.
12276 061012 000137 061540      JMP      50$      ;ABORT TEST, TIME-OUT ON DMA COMPLETION.
12277 061016 012704 000005      MOV      #5,R4      ;PASS DELAY OF 5 MILLI SECS.
12278 061022 004737 030370      JSR      PC,DELAY      ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.
12279                               ;+
12280                               ; Transmit 5 characters on each active line.
12281                               ; -
12282 061026 013705 002272      MOV      ACTLNS,R5      ;ALTER PARAMETERS FOR ALL ACTIVE LINES.
12283 061032 012700 000204      MOV      #204,R0      ;PASS PARAMETER FOR INTERNAL LOPBCK,ENABLE RX.
12284 061036 004737 040250      JSR      PC,WTWLCNC      ;INITILAISE THE LINE CONTROL REGISTER.
12285 061042 012700 177670      MOV      #177670,R0      ;PASS THE LPR CONTENTS.
12286 061046 004737 040324      JSR      PC,WTWLPR      ;SET THE LPR CONTENTS TO 38.4K BAUD.
12287 061052 012704 000012      MOV      #10.,R4      ;PASS DELAY TIME OF 10 MILLI SECONDS.
12288 061056 004737 030370      JSR      PC,DELAY      ;WAIT FOR LNCTRL AND LPR REGS TO BE UPDATED.
12289
12290 061062 012702 004012      MOV      #8UFBAS,R2      ;PASS THE START OF THE DATA PATTERN TO TX.
12291 061066 012703 000005      MOV      #5,R3      ;PASS THE LENGTH OF THE DATA PATTERN.
12292 061072 005001              CLR      R1      ;CLEAR THE LINE COUNTER.
12293 061074 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 6704.
12294 061100 010100              MOV      R1,R0
12295 061102 006300              ASL      R0      ;CALCULATE LINE OFFSET FROM THE LINE #.
12296 061104 036037 002534 002272 BIT      BITTBL(R0),ACTLNS ;TEST FOR THIS LINE BEING ACTIVE.
12297 061112 001405              BEQ      4$      ;SKIP THE TX ON THIS LINE IF IT IS NOT ACTIVE.
12298 061114 004737 030624      JSR      PC,DODMA      ;TRANSMIT THE 5 CHAR DATA PATTERN.
12299 061120 103402              BCS      .+6      ;IF NO TIME-OUT ON DMA COMPLETION, DON'T ABORT.
12300 061122 000137 061540      JMP      50$      ;ABORT TEST, TIME-OUT ON DMA COMPLETION.
12301 061126 005201              INC      R1      ;INCREMENT THE LINE NUMBER COUNTER.
12302 061130 020127 000010      CMP      R1,#NUMLNS      ;TEST FOR ALL POSSIBLE LINES HANDLED
12303 061134 002761              BLT      2$      ;LOOP IF NOT ALL LINES HANDLED.
12304
12305 061136 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 6705.
12306 061142 012701 170040      MOV      #170040,R1      ;PASS TIME OUT VALUE OF 32 MILLI SECS.
12307 061146 013702 002300      MOV      CSRA,R2      ;PASS THE ADDRESS OF THE CSR.
12308 061152 004737 037740      JSR      PC,WAIBIS      ;WAIT FOR A DMA TO COMPLETE, TX ACTION SET.
12309 061156 103170              BCC      50$      ;ABORT THE TEST IF TIME-OUT ON DMA COMPLETION.
12310 061160 012704 000005      MOV      #5,R4      ;PASS DELAY OF 5 MILLI SECS.
12311 061164 004737 030370      JSR      PC,DELAY      ;WAIT FOR LAST CHAR TO ARRIVE IN THE FIFO.
12312                               ;+
12313                               ; Read 256 chars from the FIFO checking for BMP codes.
12314                               ; Abort the test if a BMP code was found in the fifo.
12315                               ; -
12316 061170 012704 000400      MOV      #256.,R4      ;SET UP THE CHARACTER COUNTER.
12317 061174 012737 015062 005466 6$: MOV      #6706.,ERRNBR      ;SET UP ERROR NUMBER EACH TIME AROUND LOOP.
12318 061202 017702 121074      MOV      #8RBUFA,R2      ;READ A CHARACTER FROM THE FIFO.
12319 061206 100154              BPL      50$      ;ABORT THE TEST IF DATA.VALID IS CLEAR.
12320 061210 005237 005466      INC      ERRNBR      ;SET ERROR NUMBER TO 6707.
12321 061214 004737 027256      JSR      PC,CHKBMP      ;CHECK IF CHARACTER IS A BMP CODE.
12322 061220 103545              BCS      24$      ;REPORT ERROR AND ABORT TEST IF A BMP CODE.

```

HARDWARE TEST

- ORERR -

```

12323 061222 005304          DEC      R4          ;COUNT THIS CHARACTER.
12324 061224 001363          BNE      6$          ;LOOP IF NOT 256 CHARS READ FROM FIFO.
12325                                     ;+
12326                                     ; Read the remaining and verify 1 overrun plus 1 char from each line.
12327                                     ; -
12328 061226 005004          CLR      R4          ;CLEAR THE OVERRUN ERROR FLAGS.
12329 061230 012700 003712    MOV      0RXCNTR,R0
12330 061234 004737 030214    JSR      PC,CLR16W   ;CLEAR RX CHAR COUNT TABLE.
12331 061240 012737 015064 005466 8$:    MOV      06708,ERRNBR ;SET UP ERROR NUMBER EACH TIME AROUND LOOP.
12332 061246 017702 121030    MOV      0RBUFA,R2   ;READ A CHARACTER FROM THE FIFO.
12333 061252 100047          BPL      14$          ;GO ANALYZE THE RESULTS IF ALL CHARS READ.
12334 061254 004737 027256    JSR      PC,CHKBMP   ;CHECK IF CHAR IS A BMP CODE.
12335 061260 103525          BCS      24$          ;REPORT ERROR AND ABORT TEST IF A BMP CODE.
12336 061262 005237 005466    INC      ERRNBR      ;SET ERROR NUMBER TO 6709.
12337 061266 010200          MOV      R2,R0
12338 061270 000300          SWAB     R0
12339 061272 042700 177760    BIC      0177760,R0   ;CALCULATE THE LINE NUMBER OF THE CHAR.
12340 061276 006300          ASL      R0            ;FORM WORD TABLE OFFSET FOR TABLE ACCESS.
12341 061300 042702 007400    BIC      07400,R2     ;REMOVE LINE NUMBER FROM THE READ CHAR.
12342 061304 036037 002534 002272    BIT      BITTBL(R0),ACTLNS ;TEST FOR ACTIVE LINE.
12343 061312 001512          BEQ      50$          ;ABORT TEST IF FOR INACTIVE LINE.
12344 061314 005237 005466    INC      ERRNBR      ;SET ERROR NUMBER TO 6710.
12345 061320 005760 003712    TST      RXCNTR(R0)   ;CHECK THE RX CHAR COUNTER FOR THIS LINE.
12346 061324 001006          BNE      10$          ;IS THIS FIRST CHAR ON LINE?
12347 061326 020227 140000    CMP      R2,0140000   ;YES, TEST FOR NULL CHAR WITH OVERRUN.
12348 061332 001414          BEQ      12$          ;IS CHAR A NULL?
12349 061334 056004 002534    BIS      BITTBL(R0),R4 ;NO, SET THE OVERRUN BIT ERROR FLAG FOR LINE.
12350 061340 000411          BR       12$          ;GO COUNT THE CHAR AND CONTINUE.
12351 061342 026027 003712 000004 10$:    CMP      RXCNTR(R0),04
12352 061350 002073          BGE      50$          ;5TH CHAR ON THIS LINE? YES, ABORT.
12353 061352 032702 040000    BIT      0BIT14,R2   ;NO, CHECK OVERRUN BIT.
12354 061356 001402          BEQ      12$          ;IS OVERRUN BIT CLEAR? YES, GO COUNT CHAR.
12355 061360 056004 002534    BIS      BITTBL(R0),R4 ;NO, SET THE OVERRUN BIT ERROR FLAG FOR LINE.
12356 061364 005260 003712 12$:    INC      RXCNTR(R0) ;COUNT THIS CHARACTER.
12357 061370 000723          BR       8$          ;LOOP UNTIL ALL CHARS ARE READ FROM FIFO.
12358                                     ;+
12359                                     ; Test for abort condit'ns. Only none abort conditions are:
12360                                     ; 1) 2 chars RXed on a line and no overrun error bit failure detected.
12361                                     ; 2) 2 to 4 chars RXed on a line and an overrun bit failure detected.
12362                                     ; -
12363 061372 005001          14$:    CLR      R1          ;INITIALIZE LINE LOOP, CLEAR LINE OFFSET.
12364 061374 012737 015067 005466 16$:    MOV      06711,ERRNBR ;SET UP ERROR NUMBER EACH TIME AROUND LOOP.
12365 061402 036137 002534 002272    BIT      BITTBL(R1),ACTLNS
12366 061410 001415          BEQ      18$          ;LINE ACTIVE? NO, NEXT LINE.
12367 061412 026127 003712 000002    CMP      RXCNTR(R1),02   ;YES.
12368 061420 002447          BLT      50$          ;FEWER THAN 2 CHARS RXED? YES, ABORT.
12369 061422 036104 002534          BIT      BITTBL(R1),R4   ;NO.
12370 061426 001006          BNE      18$          ;OVERRUN BIT ERROR FLAG SET? YES, NEXT LINE.
12371 061430 005237 005466    INC      ERRNBR      ;SET LINE NUMBER TO 6712.
12372 061434 026127 003712 000002    CMP      RXCNTR(R1),02
12373 061442 001036          BNE      50$          ;NOT 2 CHARS RXED? YES, ABORT. NO, NEXT LINE.
12374 061444 062701 000002 18$:    ADD      02,R1      ;SET LINE OFFSET TO THE NEXT LINE.
12375 061450 020127 000020    CMP      R1,0NUMLNS*2
12376 061454 002747          BLT      16$          ;ALL LINES DONE? NO, LOOP. YES, CONTINUE.
12377                                     ;+
12378                                     ; Check for overrun error bit failures, print error message f found.
12379                                     ;

```


- ORERR -

PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419

HARDWARE TEST

- DMAADR -

```

12406 .SBTTL  HARDWARE TEST                      - DMAADR
12407 ;* *****
12408 ;*
12409 ;* DMA ADDRESSING TEST -
12410 ;* THIS TEST VERIFIES , AS FAR AS POSSIBLE , THAT THE DUT CAN PERFORM A
12411 ;* DMA FROM A FULL 22 BIT OR 18 BIT ADDRESS. THE TEST RELIES ON FINDING A
12412 ;* COMPLEMENTARY PAIR OF ADDRESSES BETWEEN THE TOP OF PHYSICAL MEMORY AND
12413 ;* THE START OF THE TOP OF THE DIAGNOSTIC PROGRAM .
12414 ;* THIS MAY INVOLVE REMOVING PART OF THE DIAGNOSTIC RUNTIME SERVICES AND
12415 ;* THEN RESTORING. THE NUMBER OF BITS THAT HAVE BEEN SUCCESSFULLY TESTED
12416 ;* WILL BE PRINTED AT THE CONSOLE AT THE END OF THE TEST.
12417 ;*
12418 ;*
12419 061552 BGNTST
12419 06155  T38::
12420
12421 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER
12422 061552 012737 000046 002364 MOV  #TNUM,TSTNUM ;SET UP THE TEST NUMBER
12423 061560 012737 177777 002362 MOV  #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST
12424 061566 012737 000001 005464 MOV  #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE
12425 061574 012737 010461 005466 MOV  #4401,ERRNBR ;SET ERROR NUMBER TO 4401
12426 061602 012737 020333 005470 MOV  #EM4401,ERRMSG ;SET ERROR MESSAGE ADDRESS IN TABLE
12427 061610 012737 024644 005472 MOV  #ER0503,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE
12428
12429 ;*
12430 ; CLEAR THE SUCCESS FLAG TO INDICATE TEST FAILURE IN CASE IT DOES
12431 ;
12432
12433 061616 005037 063344 CLR  SUCSS ;INDICATE FAILURE , IN CASE THE DUT FAILS
12434
12435
12436 ;*
12437 ; SET UP THE 004 TRAP VECTOR TO CATCH ANY EXPECTED TRAPS THAT OCCUR
12438 ;
12439
12440 061622 013737 000004 002422 MOV  #04,TP4VEC ;SAVE EXISTING 004 TRAP VECTOR
12441 061630 012737 041042 000004 MOV  #TP4RTN,#04 ;SET 004 TRAP VECTOR TO OUR SERVICE ROUTINE
12442
12443 ;*
12444 ; RESET THE DUT TO A KNOWN STATE,REMOVE THE STATUS CODES FROM THE FIFO.
12445 ; CLEAR TX AND RX INTERUPT ENABLE BITS IN THE CSR
12446 ;
12447
12448 061636 004737 030172 JSR  PC,CLNRST ;RESET THE DHV11 M , REPORT ANY ERRORS
12449 061642 103402 BCS  .+6 ;SKIP EXIT OF TEST IF NO FATAL ERROR FOUND.
12450 061644 000137 063306 JMP  604 ;EXIT THE TEST. FATAL ERROR WAS FOUND.
12451
12452
12453 ;*
12454 ; DETERMINE WHETHER MEMORY MANAGEMENT IS PRESENT
12455 ;
12456
12457 061650 005737 002470 TST  MMPRES ;IF MEM MGT IS PRESENT THEN
12458 061654 001007 BNE  11 ;AVOID SETTING THE DMA TEST ADDR FOR
12459 ; A 16 BIT MACHINE.
12460 061656 012737 001252 002432 MOV  #1252,DMTSTA ;SET UP THE FIRST DMA TEST ADDR FOR
12461 ; A 16 BIT MACHINE

```


HARDWARE TEST

- DMAADR -

```

12462 061664 012737 000021 063334      MOV    #17.,BITSTD    ;SET THE BITS TESTED TO 16 + 1
12463 061672 000513                    BR      10$          ;SINCE MEM MGT ISN'T PRESENT
12464                                     ;THERE'S NO NEED TO DETERMINE WHETHER ITS
12465                                     ;A 22 OR AN 18 BIT MACHINE, SO BRANCH.
12466
12467
12468
12469                                     ;*
12470                                     ; DETERMINE WHETHER THE HOST IS AN 18 OR A 22 BIT MACHINE. THIS IS ACCOMPLISHED
12471                                     ; BY TRYING TO READ A 22 BIT ADDRESS AND COMPARING THE READ DATA WITH THAT
12472                                     ; FROM THE EQUIVALENT 18 BIT ADDRESS.
12473                                     ;-
12474
12475                                     ;*
12476                                     ; SET UP THE PARS 1 THROUGH 5 TO RELOCATE TO THE SAME ADDRESS
12477                                     ;
12478 061674 012700 000200      1$:      MOV    #200,R0          ;SET THE PAGE BASE ADDRESS TO 200 $$$
12479 061700 012701 002476      MOV    #PARATB+2,R1        ;POINT AT THE START OF THE PAR ADDRESS TABLE $$$
12480
12481 061704 010031            2$:      MOV    R0,B(R1)+        ;LOAD THE PAR
12482 061706 062700 000200      ADD     #200,R0              ;CALCULATE THE NEXT PAGE ADDRESS
12483 061712 022701 002510      CMP     #PAR6A,R1            ;LOOP UNTIL PARS 0 THROUGH 5 $$$
12484 061716 001372            BNE     2$                    ;ARE LOADED.
12485
12486
12487                                     ;*
12488                                     ; SET UP THE PDRS FOR , NO ABORT/TRAP,UPWARD EXPANSION,128 BLOCKS PER PAGE
12489                                     ;
12490
12491 061720 012700 077406            MOV    #77406,R0          ;BIT PATTERN FOR THE PDRS
12492 061724 012701 002514            MOV    #PDRATB,R1        ;POINT AT START OF PDR ADDR TABLE
12493 061730 010031            4$:      MOV    R0,B(R1)+        ;
12494 061732 022701 002534      CMP     #PDRATE,R1            ;LOOP UNTIL ALL PDRS HAVE
12495 061736 001374            BNE     4$                    ;BEEN SET UP.
12496
12497                                     ;*
12498                                     ; SET THE MEM MGT STATUS REG #3 FOR, 22 BIT ADDRESSING.
12499                                     ; NO UNIBUS MAPPING, NO D SPACE
12500                                     ;-
12501
12502 061740 012777 000020 120520      MOV    #20.,MMMSR3        ;SET UP STATUS REG #3
12503
12504                                     ;*
12505                                     ; USE PAR #6 TO DETERMINE WHETHER THIS IS AN 18 OR A 22 BIT MACHINE BY SETTING
12506                                     ; IT TO 100000. THIS WILL SELECT A 22 BIT ADDRESS ON A 22 BIT MACHINE,
12507                                     ; ON AN 18 BIT MACHINE HOWEVER, THE MSBS WILL BE LOST AND THE MEMORY LOCATION
12508                                     ; READ WILL BE THE EQUIVALENT 18 BIT ADDRESS.
12509                                     ; PAR #6 IS USED BECAUSE WE CAN BE SURE THAT THE DIAGNOSTIC WILL NOT COME
12510                                     ; UNDER ITS INFLUENCE.
12511                                     ;-
12512
12513 061746 012777 100000 120532      MOV    #100000,PAR5A    ;LOAD THE PAGE ADDR INTO THE PAGE ADDR REG. $$$
12514
12515                                     ;*
12516                                     ; SET UP THE LOOP TO ATTEMPT TO READ A 22 BIT ADDR
12517                                     ;-
12518

```

HARDWARE TEST

- DMAADR -

```

12519 061754 012703 000005      MOV    #5,R3      ;INITIALISE LOOP COUNT
12520 061760 012702 005260      MOV    #SDPBAS,R2    ;SELECT THE VIRTUAL ADDRESS #SDPBAS, THIS WILL BE
12521                                ;BITS 0 TO 12 OF THE PHYSICAL ADDRESS
12522 061764 010204              MOV    R2,R4      ;SAVE THE DATA ADDRESS
12523 061766 062704 120000      ADD     #120000,R4    ;CONVERT THE VIRT. ADDR INTO AN ADDR WITHIN $$$
12524                                ;THE INFLUENCE OF PAR #6.
12525 061772 012737 000001 002426  MOV    #BIT0,BITLNG ;INDICATE A 22 BIT MACHINE IN CASE IT IS
12526                                ;
12527 062000 106427 000340      MTPS    #PRI07     ;DISABLE CLOCK INTERRUPTS
12528                                ;
12529 062004 010400              MOV    R4,R0      ;SET UP THE MOVE SOURCE
12530 062006 012701 063340      MOV     #DEST,R1    ;SET UP THE DESTINATION
12531                                ;
12532                                ;*
12533                                ; ENABLE THE MEM MGT AND ATTEMPT TO READ THE 22 BIT ADDRESS. IF A TRAP OCCURS
12534                                ; THEN THE HOST MUST BE A 22 BIT MACHINE SINCE WE HAVE ENSURED THAT
12535                                ; THE 18 BIT ADDRESS EXISTS.
12536                                ;
12537                                ;
12538 062012 012777 000001 120444      MOV    #BIT0,#MMSR0 ;ENABLE MEM MGT
12539 062020 004737 030112      JSR     PC,CKTRAP ;PERFORM THE MOVE AND CHECK FOR A TRAP
12540 062024 005077 120434      CLR     #MMSR0    ;DISABLE MEM MGT
12541                                ;
12542 062030 106427 000240      MTPS    #PRI05     ;ENABLE CLOCK INTERRUPTS
12543                                ;
12544 062034 005737 002424      TST     TP4FLG    ;DID A TRAP OCCUR ?
12545 062040 001022              BNE     #0        ;YES . THEN JUMP AND INDICATE A 22 BIT MACHINE
12546                                ;
12547                                ;*
12548                                ; SINCE A TRAP HASN'T OCCURED THEN EITHER THE MACHINE IS A 22 BIT MACHINE WITH
12549                                ; MEMORY AT THE ADDRESS JUST READ , OR , ITS AN 18 BIT MACHINE IN WHICH CASE
12550                                ; THE ADDRESS JUST READ WOULD BE ONE OF THE DATA WORDS AT ADDRESS #SDPBAS.
12551                                ;
12552                                ;
12553 062042 023712 063340      CMP     DEST,(R2)    ;COMPARE READ WORD WITH DATA WORD
12554 062046 001017              BNE     #0        ;IF NOT THE SAME THEN JUMP AND INDICATE A
12555                                ; 22 BIT MACHINE.
12556                                ;
12557                                ;*
12558                                ; IN ORDER TO MAKE SURE THAT THIS REALLY ISN'T A 22 BIT MACHINE I.E. THAT
12559                                ; THE MATCH OF THE READ WORD AND THE DATA WORD WAS NOT MERELY COINCIDENCE.
12560                                ; ANOTHER FOUR ADDRESSES ARE READ AND COMPARED WITH THE DATA.
12561                                ;
12562                                ;
12563 062050 062702 000002      ADD     #2,R2      ;SELECT NEXT TEST DATA ADDRESS
12564 062054 062704 000002      ADD     #2,R4      ;SELECT NEXT READ ADDRESS
12565 062060 005303              DEC     R3        ;DECREMENT THE DATA COUNT
12566 062062 001346              BNE     #6        ;REPEAT THE READ ATTEMPTS UNTILL ALL FIVE DATA
12567                                ; WORDS HAVE BEEN SUCCESSFULLY MATCHED, AND THEN
12568 062064 005037 002426      CLR     BITLNG    ;INDICATE AN 18 BIT MACHINE.
12569                                ;
12570                                ;*
12571                                ; SET UP THE HIGHEST POSSIBLE TEST ADDRESS IN DMTSTA
12572                                ;
12573                                ;
12574 062070 012737 005252 002432      MOV     #5252,DMTSTA ;SET UP THE FIRST DMA TEST ADDRESS FOR THE
12575                                ; 18 BIT MACHINE

```

HARDWARE TEST

DMAADR -

```

12576 062076 012737 000023 063334      MOV    #19.,BITSTD      ;SET THE BITS TESTED TO 18 BITS + 1.
12577 062104 000406                      BR      10$            ;AVOID SETTING DMTSTA FOR THE 22 BIT MACHINE
12578 062106 012737 125252 002432 8$:    MOV    #125252,DMTSTA    ;SET UP THE FIRST PAGE ADDR FOR
12579                                ;A 22 BIT MACHINE.
12580 062114 012737 000027 063334      MOV    #23.,BITSTD      ;SET THE BITS TESTED TO 22 BITS +1.
12581
12582                                ;*
12583                                ; TRY AND FIND A COMPLEMENTARY PAIR OF ADDRESSES WITHIN THE MEMORY AND SAVE
12584                                ; THE CONTENTS OF THE TWO AREAS. THE TEST IS ABANDONED IF A COMPLEMENTARY
12585                                ; PAIR HAS NOT BEEN FOUND BEFORE THE AREA OF MEMORY CONTAINING THE
12586                                ; DIAGNOSTIC IS ENCOUNTERED.
12587                                ;*
12588
12589 062122 012737 041020 000004 10$:    MOV    #TP4BRT,004      ;CHANGE THE 004 TRAP VECTOR TO POINT TO
12590                                ;TP4BRT SINCE THIS IS THE ROUTINE ASSOCIATED
12591                                ;WITH THE BYTE SUBROUTINE CKTRPB.
12592
12593 062130                                MEMORY FFREM              ;GET THE ADDRESS OF THE FIRST FREE WORD
12594 062130 104431                                TRAP      C$MEM
12595 062132 010037 002430                                MOV      RO,FFREM
12596                                ;OF MEMORY ABOVE THE DIAGNOSTIC.
12597
12596 062136 012701 004012      MOV    #BUF8AS,R1      ;POINT AT THE BUFFER WHERE THE CONTENTS OF
12597                                ;THE MEMORY BEING READ ARE TO BE SAVED.
12598 062142 005004      CLR      R4              ;CLEAR THE COMPLEMENTARY PAIR INDICATOR (CPI)
12599 062144 106427 000340      MTPS   #PRI07          ;DISABLE LTC INTERRUPTS
12600
12601 062150 005204      12$:    INC      R4              ;INCREMENT THE CPI
12602 062152 005005      CLR      R5              ;INDICATE THAT A SAVE OF THE DATA AT
12603                                ;(DMTSTA) IS REQUIRED.
12604 062154 012703 000020      MOV    #16.,R3              ;SET THE NUMBER OF BYTES TO BE READ
12605 062160 004737 030502      JSR     PC,DMRW          ;SAVE THE DATA CONTAINED AT ADDRESS DMTSTA.
12606 062164 012701 004412      MOV    #BJFMID,R1      ;POINT AT SECOND STORAGE AREA
12607 062170 005737 002424      TST     TP4FLG          ;IF WE HAVE VALID MEMORY THEN AVOID CLEARING
12608 062174 001403      BEQ     14$              ;THE CPI AND RESETTING THE SAVE AREA ADDR
12609
12610 062176 005004      CLR      R4              ;CLEAR THE CPI.
12611 062200 012701 004012      MOV    #BUF8AS,R1      ;RESET THE ADDR FOR THE SAVED DATA STORE
12612 062204 022704 000002      14$:    CMP     #2,R4              ;IF A PAIR OF COMPLEMENTARY ADDRESSES HAVE
12613                                ;BEEN FOUND THEN
12614 062210 001447      BEQ     17$              ;GO AND WRITE THE TEST DATA TO THESE ADDRS.
12615 062212 013737 002432 063342      MOV    DMTSTA,ODTSTA    ;SAVE THE OLD DMTSTA
12616 062220 000241      CLC                      ;CLEAR CARRY READY FOR THE ROTATION
12617 062222 006037 002432      ROR     DMTSTA          ;COMPLEMENT THE DMTSTA TO PRODUCE THE NEXT
12618                                ; DMA TEST ADDR.
12619 062226 005337 063334      DEC     BITSTD          ;DECREMENT THE NUMBER OF BITS TESTED COUNT
12620
12621                                ;*
12622                                ; CHECK THAT THE NEW DMTSTA IS NOT INSIDE THE DIAGNOSTIC PROGRAM
12623                                ;*
12624
12625 062232 032737 176000 002432      BIT     #176000,DMTSTA    ;IS THE DMTSTA > 1252 . IF IT IS THEN WE'RE
12626                                ; SAFE SO.
12627 062240 001343      BNE     12$              ;BRANCH AND CONTINUE WITH THE SEARCH
12628 062242 004737 030430      JSR     PC,DM16B        ;CONVERT THE DMTSTA TO A PHYSICAL ADDR.
12629 062246 020037 002430      CMP     RO,FFREM          ;ARE WE INSIDE THE DIAGNOSTIC REGION ?
12630 062252 103336      BHS     12$              ;NO . THEN BRANCH AND CONTINUE WITH THE SEARCH

```

HARDWARE TEST

- DMAADR -

```

12631
12632
12633
12634
12635
12636
12637
12638
12639 062254 022737 000252 002432      CMP    #252,DMTSTA      ;IF THE BIT HAS ALREADY BEEN SET THEN
12640 062262 001014                      BNE    15$              ;ABANDON THE TEST,AFTER REPORTING THE ERROR.
12641                                     ; BECAUSE NO SUITABLE MEMORY HAS BEEN FOUND.
12642 062264 012737 000652 002432      MOV    #652,DMTSTA      ;SET THE BIT
12643 062272 062700 040000              ADD    #40000,R0        ;ADD THE BIT INTO THE PHYSICAL ADDR
12644 062276 020037 002430              CMP    R0,FFREM        ;IF WE'RE NOW STILL INSIDE THE DIAGNOSTIC THEN
12645 062302 103404                      BLO    15$              ;REPORT ERROR AND ABANDON THE TEST.
12646 062304 012737 000016 063334      MOV    #14,.BITSTD    ;OTHERWISE SET THE BITS TESTED TO 14 BITS.
12647 062312 000716                      RR      12$            ;CONTINUE WITH THE SEARCH.
12648
12649
12650 062314 005237 005466              15$: INC    ERRNBR        ;SET THE ERROR NUMBER TO 4402
12651 062320 012701 020355              MOV    #EM4402,R1      ;SELECT MESSAGE TO BE REPORTED.
12652                                     ; " NO SUITABLE ADDR FOUND. DMA TEST ABORTED "
12653
12654
12655 062324 000137 063304              16$: JMP     34$              ;JUMP TO THE ERROR.
12656
12657
12658
12659                                     ;*
12660                                     ; WRITE THE TEST DATA INTO THE TWO AREAS JUST FOUND. IF A TRAP OCCURS WHILE
12661                                     ; WE ARE WRITING DATA INTO THESE AREAS THEN THE HOST MACHINE IS AT FAULT.
12662                                     ;*
12663 062330 012700 005260              17$: MOV    #SDPBAS,R0      ;SET UP THE SOURCE ADDR FOR THE MOVE AS OUR
12664                                     ;TEST DATA PATTERN.
12665 062334 013737 002432 063336      MOV    DMTSTA,DUMY      ;SAVE THE LOWER DMTSTA
12666 062342 013737 063342 002432      MOV    ODTSTA,DMTSTA    ;START WITH THE HIGHER OF THE TWO
12667                                     ; COMPLEMENTARY ADDRESSES.
12668 062350 012703 000020              MOV    #16.,R3        ;SET THE NUMBER OF DATA BYTES TO BE WRITTEN
12669 062354 012705 000001              MOV    #1,R5          ;INDICATE TO WRITE TO DMTSTA
12670
12671
12672 062360 012701 000340              MOV    #340,R1        ;SET PRIORITY 7 TO DISABLE THE CLOCK
12673 062364 004737 036016              JSR    PC,STPSW        ;
12674
12675 062370 005237 005466              INC    ERRNBR        ;SET THE ERROR NUMBER TO 4403
12676 062374 012701 020423              MOV    #EM4403,R1      ;SELECT THE MESSAGE.
12677                                     ; "HOST FAILURE. WRITE FAILED TO AN ADDR WHICH
12678                                     ; HAD BEEN SUCCESSFULLY READ. TEST ABANDONED "
12679
12680 062400 004737 030502              JSR    PC,DMRW        ;PERFORM THE TRANSFER
12681 062404 005737 002424              TST    TP4FLG        ;EXIT IF HOST FAILURE
12682 062410 001345                      BNE    16$              ;AND REPORT ERROR.
12683 062412 013737 063336 002432      MOV    DUMY,DMTSTA    ;SELECT THE LOWER DMA TEST ADDR.
12684 062420 012700 005304              MOV    #SDP28,R0      ;SELECT THE NEXT DATA PATTERN
12685 062424 004737 030502              JSR    PC,DMRW        ;PERFORM THE TRANSFER
12686 062430 005737 002424              TST    TP4FLG        ;EXIT IF HOST FAILURE
12687 062434 001333                      BNE    16$              ;

```

HARDWARE TEST

- DMAADR -

```

12688
12689
12690      ;*
12691      ; SET UP THE DHV11-M TO PERFORM THE DMA.
12692      ;*
12693
12694      ;*
12695      ; SET INTERNAL LOOPBACK, ENABLE THE RECIEVER FUNCTION ON THE LINE.
12696      ; SET THE LPR ON THE LINE TO 38.4K BAUD, 8 BITS PER CHARACTER, ODD PARITY.
12697      ; 2 STOP BITS. ENABLE THE TRANSMITTER ON THE LINE.
12698      ;*
12699 062436 005237 005466      INC      ERRNBR      ;SET THE ERRNBR TO 4404
12700
12701
12702 062442 004737 030772      JSR      PC,FINACT      ;FIRST FIND AN ACTIVE LINE ON WHICH TO PERFORM
12703                                ; THE DMA.
12704 062446 010102      MOV      R1,R2      ;SAVE THE LINE NUMBER ON WHICH THE DMA WILL OCCUR
12705 062450 012701 020520      MOV      @EM4404,R1      ;SELECT THE MESSAGE.
12706                                ; "NO ACTIVE LINES , TEST ABANDONED"
12707 062454 103402      BCS      .+6      ;EXIT IF A LINE COULD NOT BE FOUND ,AFTER FIRST
12708 062456 000137 063106      JMP      30$      ;RESTORING THE CONTENTS OF MEMORY.
12709 062462 010201      MOV      R2,R1      ;RESTORE THE ACTIVE LINE NUMBER.
12710
12711      ;*
12712      ; AN ACTIVE LINE HAS BEEN FOUND
12713      ;*
12714 062464 012700 000204      MOV      @204,R0      ;PASS THE LNCTRL CONTENTS
12715 062470 004737 040250      JSR      PC,WTWLNC      ;INITIALISE THE LNCTRL REGISTER
12716 062474 012700 177670      MOV      @177670,R0      ;PASS THE LPR CONTENTS
12717 062500 004737 040324      JSR      PC,WTWLPR      ;INITIALISE THE LPR REGISTER
12718 062504 004737 036446      JSR      PC,TXENBL      ;ENABLE TRANSMITTER ON THE LINE
12719
12720      ;*
12721      ; INITIATE THE DMA
12722      ;*
12723
12724 062510 013705 063342      MOV      ODTSTA,R5      ;START FROM THE HIGHER OF THE PAIR OF ADDR.
12725 062514 012704 005260      MOV      @SDPBAS,R4      ;SET UP THE ADDR OF THE DATA PATTERN
12726 062520 010137 063332      MOV      R1,80$      ;SAVE THE LINE NUMBER FOR THE DMA
12727 062524 012737 000002 063330      MOV      @2,70$      ;INITIALISE THE LOOP COUNT
12728
12729 062532 012700 000052      18$: MOV      @52,R0      ;SET UP THE LSB'S
12730 062536 005003      CLR      R3      ;CLEAR THE REG THAT WILL HOLD THE 6 MSB'S
12731 062540 012702 000006      MOV      @6,R2      ;CONVERT THE DMTSTA INTO
12732 062544 006305      20$: ASL      R5      ;A PHYSICAL ADDRESS WITH
12733 062546 006103      ROL      R3      ;THE MSB S IN REG @3.
12734 062550 005302      DEC      R2      ;
12735 062552 001374      BNE      20$      ;
12736 062554 032705 000100      BIT      @100,R5      ;TEST BIT @6 OF THE DMTSTA
12737 062560 001402      BEQ      22$      ;
12738 062562 012700 000025      MOV      @25,R0      ;ALTER THE LSB S IF BIT @6 WAS SET.
12739 062566 060005      22$: ADD      R0,R5      ;ADD IN THE LSB'S
12740 062570 052703 000200      BIS      @200,R3      ;SET BIT @7.
12741
12742 062574 013777 063332 117476      MOV      80$,@CSRA      ;SELECT THE LINE ON WHICH TO PERFORM THE DMA.
12743
12744 062602 012737 010465 005466      MOV      @4405,ERRNBR      ;SET ERROR NUMBER 4405

```

HARDWARE TEST

- DMAADR -

```

12745 062610 012701 020557      MOV    #EM4405,R1      ;SELECT THE MESSAGE,
12746                               ; " DMA_START BIT FOUND SET BEFORE DMA INIT.
12747                               ;TEST ABANDONED"
12748 062614 105777 117474      TSTB    @TXAD2A      ;TEST THE DUT DMA-START BIT
12749 062620 100532              BMI     30$      ;EXIT WITH ERROR IF SET ,AFTER FIRST RESTORING
12750                               ;THE CONTENTS OF MEMORY.
12751 062622 012777 000020 117466  MOV    #16.,@TXBFCA    ;SET UP CHARACTER COUNT
12752 062630 010577 117456      MOV    R5,@TXAD1A    ;SET UP BITS 0 TO 15 OF THE PHYISCAL ADDR.
12753 062634 110377 117454      MOVB    R3,@TXAD2A    ;SET UP BITS 16 TP 21 , AND INITIATE THE DMA.
12754
12755                               ;*
12756                               ; WAIT FOR THE DMA TO COMPLETE AND THE LAST CHARACTER TO BE RECIEVED
12757                               ;*
12758
12759 062640 012701 170144      MOV    #170144,R1    ;TEST BIT 15, TIME OUT OF 100 MS.
12760 062644 013702 002300      MOV    CSRA,R2      ;PASS THE ADDR OF THE REG TO TEST.
12761
12762 062650 005237 005466      INC     ERRNBR      ;SET ERROR NUMBER TO 4406
12763
12764 062654 004737 037740      JSR     PC,WAIBIS    ;WAIT FOR BIT TO SET
12765 062660 012701 020653      MOV    #EM4406,R1    ;SELECT THE MESSAGE ,
12766                               ; " TIME-OUT OCCURED WAITING FOR DMA TO
12767                               ;COMPLETE. TEST ABANDONED"
12768 062664 103110              BCC     30$      ;EXIT IF TIME-OUT OCCURED, AFTER FIRST ,
12769                               ;RESTORING THE CONTENTS OF MEMORY.
12770 062666 010402              MOV    R4,R2      ;SAVE R4
12771 062670 012704 000005      MOV    #5,R4      ;SET 5 MS DELAY
12772 062674 004737 030370      JSR     PC,DELAY    ;DELAY TO ALLOW LAST CHARACTER TO BE RECIEVED
12773 062700 010204              MOV    R2,R4      ;RESTORE R4
12774
12775                               ;*
12776                               ; READ THE CONTENTS OF THE RXFIFO AND COMPARE THEM WITH THE CORRECT DATA
12777                               ;*
12778
12779 062702 005003              CLR     R3      ;CLEAR THE READ DATA COUNTER
12780 062704 012705 000200      MOV    #128.,R5    ;SET THE MAX BMP CODE READ COUNT
12781
12782 062710 012737 010467 005466 24$:  MOV    #4407.,ERRNBR    ;SET THE ERRNBR TO 4407
12783 062716 012701 020727      MOV    #EM4407,R1    ;SELECT THE MESSAGE ,
12784                               ; " RXFIFO EMPTY TOO SOON, DMA FAILED
12785                               ;TEST ABANDONED"
12786
12787 062722 017702 117354      MOV    @RBUFA,R2    ;READ THE CHARACTER FROM THE FIFO
12788 062726 100067              BPL     30$      ;BRANCH TO REPORT ERROR IF FIFO EMPTY TOO SOON,
12789                               ;AFTER FIRST RESTORING THE CONTENTS OF MEMORY.
12790 062730 012700 170301      MOV    #170301,R0    ;SET UP BIT MASK OF A BMP CODE
12791 062734 040200              BIC     R2,R0      ;TRY TO CLEAR THE BMP CODE MASK
12792 062736 001011              BNE     28$      ;BRANCH IF NOT A BMP CODF
12793 062740 004737 035624      JSR     PC,SAVBMP    ;SAVE THE BMP CODE ON THE QUEUE
12794
12795 062744 005237 005466      INC     ERRNBR      ;SET THE ERRNBR TO 4408
12796 062750 012701 021011      MOV    #EM4408,R1    ;SELECT THE MESSAGE,
12797                               ; " TOO MANY BMP CODES FOUND IN THE RXFIFO.
12798                               ;TEST ABANDONED"
12799
12800 062754 005305              DEC     R5      ;DEC THE MAX BMP CODE READ COUNT
12801 062756 001453              BEQ     30$      ;GO REPORT ERROR IF TOO MANY BMP CODES FOUND .

```


HARDWARE TEST

- DMAADR -

```

12802
12803 062760 000753          BR      24$      ;AFTER FIRST RESTORING THE CONTENTS OF MEMORY.
12804                                     ;DON'T COUNT THE BMP CODE AS A VALID CHARACTER
12805
12806 062762 012737 010471 005466 28$:  MOV    #4409.,ERRNBR  ;SET THE ERRNBR TO 4409
12807 062770 010201          MOV    R2,R1      ;SAVE THE CHARACTER FROM THE FIFO
12808 062772 012702 021054          MOV    #EM4409,R2  ;SELECT THE MESSAGE
12809                                     ; " BAD BIT BETWEEN BITS 0 AND "
12810 062776 012737 026440 005472      MOV    #ER9101,ERRBLK ;SELECT THE ERROR ROUTINE.
12811 063004 012737 177777 063344      MOV    #-1,SUCSS  ;INDICATE 'BAD BITS' FAILURE
12812
12813 063012 122401          CMPB   (R4)+,R1      ;COMPARE CHAR FROM FIFO WITH THE CORRECT DATA.
12814 063014 001034          BNE     30$          ;BRANCH IF INCORRECT AND RESTORE MEM CONT'S.
12815 063016 005037 063344          CLR     SUCSS   ;INDICATE NON TEST SPECIFIC FAILURE E.G. TIME-OUTS
12816 063022 005203          INC     R3          ;COUNT THIS CHARACTER.
12817 063024 022703 000020          CMP     #16.,R3  ;HAVE WE RECIEVED ALL THE CHARACTERS ?
12818 063030 001327          BNE     24$          ;LOOP UNTIL ALL CHARACTERS (NON-BMP) ARE READ.
12819 063032 005337 063330          DEC     70$     ;DECREMENT THE LOOP COUNT
12820 063036 001420          BEQ     29$          ;BRANCH IF BOTH DMA'S ARE COMPLETED
12821 063040 012704 005304          MOV     #SDP2B,R4  ;SET UP THE SECOND DATA PATTERN
12822 063044 013705 002432          MOV     DMTSTA,R5 ;SET UP THE OTHER DMA TEST ADDRESS
12823
12824 063050 012737 010472 005466      MOV     #4410.,ERRNBR ;SET ERRNBR TO 4410
12825 063056 012701 021111          MOV     #EM4410,R1 ;SELECT THE MESSAGE
12826                                     ; RXFIFO FAILED TO PURGE. TEST ABANDONED "
12827 063062 012737 024644 005472      MOV     #ER0503,ERRBLK ;SELECT THE ERROR ROUTINE
12828
12829 063070 004737 032722          JSR     PC,PUFIFO ;PURGE THE RXFIFO
12830 063074 103004          BCC     30$          ;EXIT WITH ERROR IF FIFO WOULD NOT PURGE
12831                                     ;AFTER FIRST RESTORING THE CONTENTS OF MEMORY.
12832 063076 000615          BR      18$          ;OTHERWISE REPEAT.
12833
12834 063100 012737 000001 063344 29$:  MOV     #1,SUCSS  ;INDICATE THAT WE HAVE BEEN ABLE TO TEST.
12835                                     ;SOME OF THE BITS.
12836
12837
12838
12839          ;+
12840          ; RESTORE THE ORIGINAL DATA IN THE MEMORY
12841          ;-
12842 063106 013737 002432 063336 30$:  MOV     DMTSTA,DUMY ;START WITH THE HIGHER OF THE PAIR OF DMTSTA
12843 063114 013737 063342 002432      MOV     ODTSTA,DMTSTA ;
12844 063122 012700 004012          MOV     #BUF8AS,R0 ;POINT AT THE START OF THE SAVED DATA AREA
12845 063126 012705 000001          MOV     #1,R5      ;SELECT WRITE TO (DMTSTA)
12846 063132 012703 000020          MOV     #16.,R3    ;PASS NUMBER OF BYTES TO BE WRITTEN
12847 063136 004737 030502          JSR     PC,DMRW   ;RESTORE THE DATA
12848 063142 005737 002424          TST     TP4FLG   ;GO REPORT ERROR IF A TRAP OCCURED
12849 063146 001012          BNE     31$          ;
12850 063150 013737 063336 002432      MOV     DUMY,DMTSTA ;NOW RESTORE THE DATA FROM THE LOWER
12851                                     ;OF THE PAIR OF TEST ADDRESSES.
12852 063156 012700 004412          MOV     #BUFMD R0 ;POINT AT THE START OF THE SAVED DATA AREA
12853 063162 004737 030502          JSR     PC,DMRW   ;RESTORE THE DATA
12854
12855 063166 005737 002424          TST     TP4FLG   ;GO REPORT ANY ERRORS IF A NO TRAP
12856 063172 001411          BEQ     32$          ; OCCURED DURING THE RESTORE.
12857
12858 063174 012737 010473 005466 31$:  MOV     #4411.,ERRNBR ;SET THE ERROR NUMBER TO 4411

```

HARDWARE TEST

- DMAADR -

```

12859 063202 012701 021140      MOV    #EM4411,R1      ;SELECT THE MESSAGE ,
12860                                ; " HOST FAILURE. WRITE FAILURE TO AN ADDR
12861                                ;WHICH HAD PREVIOUSLY BEEN SUCCESSFULLY
12862                                ;WRITTEN TO. "
12863 063206 012737 024644 005472      MOV    #ER0503,ERRBLK    ;SELECT THE ERROR ROUTINE
12864 063214 000433                BR      34$          ;REPORT THE ERROR
12865
12866                                ;*
12867                                ; HAS THE TEST BEEN SUCCESSFUL, PRINT THE BITS TESTED IF IT HAS,
12868                                ; REPORT THE ERRORS OTHERWISE.
12869                                ; -
12870
12871
12872 063216 005737 063344      32$:   TST     SUCSS          ;IF THE ERROR IS NON TEST SPECIFIC THEN
12873 063222 001430                BEQ     34$          ;BRANCH TO REPORT ERRORS
12874 063224 013701 063334      MOV     BITSTD,R1        ;LOAD THE NUMBER OF BITS TESTED
12875 063230 005301                DEC     R1            ;DEC TO GIVE THE BIT POSITION OF THE MSB TESTED.
12876 063232 022737 000001 063344      CMP     #1,SUCSS      ;IF THE BITS TESTED ARE BAD THEN
12877 063240 001021                BNE     34$          ;BRANCH AND REPORT ERRORS.
12878
12879                                ;*
12880                                ; OTHERWISE DETERMINE IF PRINTING OF THE SUCCESSFULLY TESTED BITS WAS REQUESTED.
12881                                ; -
12882
12883
12884 063242 032737 000040 002262      BIT     #BIT05,OPTION    ; PRINT THE BITS TESTED IF THE SOFTWARE
12885 063250 001416                BEQ     60$          ;OPTION HAS REQUESTED IT
12886 063252 010102                MOV     R1,R2          ;CALCULATE THE NUMBER OF BITS WHICH HAVE
12887 063254 005202                INC     R2            ; BEEN TESTED SUCCESSFULLY.
12888 063256                PRINTB    #EF4401,R1,R2        ;PRINT THE NUMBER OF BITS TESTED MESSAGE.
12889                                MOV     R2,-(SP)
12890                                MOV     R1,-(SP)
12891                                MOV     #EF4401,-(SP)
12892                                MOV     #3,-(SP)
12893                                MOV     SP,R0
12894                                TRAP     C#PNTB
12895                                ADD      #10,SP
12896 063302 000401                BR      60$          ;EXIT THE TEST
12897
12898
12899 063304                34$:   ERROR                    ; REPORT ERRORS
12900 063304 104460                TRAP     C#ERROR
12901
12902
12903 063306 106427 000240      60$:   MTPS    #PRI05          ;ENABLE THE CLOCK
12904 063312 013737 002422 000004      MOV     TP4VEC,#44      ;RESTORE THE NORMAL 004 TRAP VECTOR
12905 063320 005037 002362      CLR     CTRLCF          ;INDICATE THAT WE ARE NOT WITHIN A TEST
12906
12907                                EXIT     TST
12908
12909                                TRAP     C#EXIT
12910                                .WORD    L10101-.
12911
12912                                ;*
12913                                ; ***** LOCAL VARIABLE AREA *****
12914                                ; -
12915
12916
12917
12918
12919
12920
12921
12922
12923
12924
12925
12926
12927
12928
12929
12930
12931
12932
12933
12934
12935
12936
12937
12938
12939
12940
12941
12942
12943
12944
12945
12946
12947
12948
12949
12950
12951
12952
12953
12954
12955
12956
12957
12958
12959
12960
12961
12962
12963
12964
12965
12966
12967
12968
12969
12970
12971
12972
12973
12974
12975
12976
12977
12978
12979
12980
12981
12982
12983
12984
12985
12986
12987
12988
12989
12990
12991
12992
12993
12994
12995
12996
12997
12998
12999

```


HARDWARE TEST

- DMAADR -

```

12906 063330 000000      70$: .WORD 0      ;COUNTER FOR THE NUMBER OF DMA'S COMPLETED
12907 063332 000000      80$: .WORD 0      ;SAVE AREA FOR THE ACTIVE LINE NUMBER
12908 063334 000000      BITSTD: .WORD 0    ;NUMBER OF BITS TESTED
12909 063336 000000      DUMY: .WORD 0      ;DUMMY VARIABLE
12910 063340 000000      DEST: .WORD 0      ;SAVE AREA FOR READ WORD , WHEN DETERMINING
12911                                     ;MACHINE ADDRESS LENGTH.
12912 063342 000000      ODTSTA: .WORD 0     ;HIGHER OF THE PAIR OF COMPLEMENTARY ADDR.
12913 063344 000000      SUCSS: .WORD 0     ;SUCCESS INDICATOR, 1 - ERROR DUE TO BAD BITS
12914                                     ;                1 - SUCCESSFUL TEST
12915                                     ;                0 - OTHER ERRORS
12916
12917      ;+
12918      ;***** END *****
12919      ;-
12920
12921
12922
12923 063346      ENDTST
12924 063346      L10101: TRAP C#ETST
12924 063346 104401

```

HARDWARE TEST

- SHTSNG -

```

12926 .SBTTL HARDWARE TEST - SHTSNG -
12927 ;++ *****
12928 ;* - Short Single Character Mode Test -
12929 ;* This test verifies that the Device Under Test (DUT) will perform
12930 ;* transmission and reception correctly using the single character
12931 ;* mode interrupts. This test is performed at 38.4K baud, 1 stop
12932 ;* bit, odd parity, 8 bits per character. A single 256 char data pattern
12933 ;* is sent to each active line.
12934 ;*
12935 ;-- *****
12936
12937 063350 BGNTST
12938 063350
12938 12938 000047 T39::
12939 063350 012737 000047 002364 TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
12940 063356 012737 177777 002362 MOV #TNUM,TSTNUM ;SET UP THE TEST NUMBER. (90)
12941 063364 012737 000001 005464 MOV #-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
12942 063372 012737 021451 005466 MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
12943 063400 012737 022321 005470 MOV #9001.,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
12944 063406 005037 002650 MOV #EM9001,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRMBL.
12945 CLR ERSMBF ;INITIALIZE THE "REPORT ERROR SUMMARY" FLAGS.
12946 ;+
12947 ; Reset the DUT to a known state, remove the status codes from the fifo.
12948 ; Clear TX and RX interrupt enable bits in the CSR.
12949 ; This subroutine reports error >>>> 9001 <<<<.
12950 063412 004737 030172 JSR PC,CLNRST ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
12951 063416 103402 BCS .+6 ;SKIP EXIT OF TEST IF NO FATAL ERROR FOUND.
12952 063420 000137 063704 JMP 60$ ;EXIT THE TEST, FATAL ERROR WAS FOUND.
12953 063424 012737 021452 005466 MOV #9002.,ERRNBR ;SET THE ERROR NUMBER.
12954
12955 ;+
12956 ; Set up for Transmit and Receive interrupts.
12957 ;--
12958 063432 106427 000240 MTPS #PRI05 ;DISABLE DEVICE INTERRUPTS.
12959 063436 SETVEC TXVECA,#TXSCHR,#PRI05 ;SELECT SINGLE CHAR TX INT SERVICE RTN.
12960 063436 012746 000240 MOV #PRI05,-(SP)
12961 063442 012746 041272 MOV #TXSCHR,-(SP)
12962 063446 013746 002270 MOV TXVECA,-(SP)
12963 063452 012746 000003 MOV #3,-(SP)
12964 063456 104437 TRAP C$SVEC
12965 063460 062706 000010 ADD #10,SP
12966 063464 SETVEC RXVECA,#RXCHRS,#PRI05 ;SELECT RX INT SERVICE RTN.
12967 063464 012746 000240 MOV #PRI05,-(SP)
12968 063470 012746 040570 MOV #RXCHRS,-(SP)
12969 063474 013746 002266 MOV RXVECA,-(SP)
12970 063500 012746 000003 MOV #3,-(SP)
12971 063504 104437 TRAP C$SVEC
12972 063506 062706 000010 ADD #10,SP
12973 063512 106427 000000 MTPS #PRI00 ;ALLOW INTERRUPTS.
12974 ;+
12975 ; Clear the error counter table.
12976 ; This table will accumulate error count totals for each line during this test.
12977 ;--
12978 063516 012700 003452 MOV #ERCNTB,R0
12979 063522 004737 030214 JSR PC,CLR16W ;CLEAR THE RX ERROR COUNTERS TABLE.
12980 ;+
12981 ; Transmit and receive 256 byte data patterns.

```

- SHTSNG -

```

12970      ;
12971      ;
12972      ; Initialize the long data pattern and parameters for the SCHKST call.
12973      ;
12974 063526 012737 021465 005466      MOV     #9013.,ERRNBR      ;SET THE ERROR NUMBER TO 9013.
12975 063534 012702 004012      4$:    MOV     #BUFBAS,R2      ;INITIALIZE THE LONG DATA
12976 063540 005003              CLR     R3              ; PATTERN IN THE GENERAL
12977 063542 110322      6$:    MOVB    R3,(R2)+      ; DATA BUFFER TO A 256
12978 063544 005203              INC     R3              ; BYTE PATTERN COUNTING
12979 063546 020227 004412      CMP     R2,#BUFMID      ; FROM ZERO TO 255.
12980 063552 103773              BLO     6$              ;
12981      ;
12982      ; Initialize for, and get the LPR contents.
12983      ;
12984 063554 005037 002634              CLR     GPRS08      ;CLEAR THE GPR SAVE AREA R1 STORAGE TO INDICATE
12985              ; THAT THIS IS THE FIRST TIME IN GETLP2.
12986 063560 004737 036036      8$:    JSR     PC,SWAPO      ;SWAP GPRS WITH GPR SAVE AREA ZERO FOR GETLP1.
12987 063564 004737 031134              JSR     PC,GETLP2      ;GET NEXT SET OF LPR CONTENTS, OR CARRY CLEAR.
12988 063570 004737 036036              JSR     PC,SWAPO      ;SWAP BACK GPRS AND GPR SAVE AREA ZERO.
12989 063574 103036              BCC     10$      ;EXIT LOOP IF ALL COMBINATIONS OF LPRS DONE.
12990 063576 010001              MOV     R0,R1      ;PASS THE LPR CONTENTS TO GETTIM AND VANSUP.
12991 063600 004737 031264              JSR     PC,GETTIM      ;GET TIME-OUT BASED ON MINIMUM BAUDRATE IN USE.
12992 063604 012702 004012              MOV     #BUFBAS,R2      ;SET UP POINTER TO START OF SHORT DATA PATTERN.
12993 063610 012703 000400              MOV     #BUFMID-BUFBAS,R3      ;SET UP THE DATA PATTN LENGTH.
12994 063614 012704 000001              MOV     #1,R4      ;SPECIFY TO SEND 1 DATA PATTERN TO EACH LINE.
12995 063620 004737 037466              JSR     PC,VANSUP      ;SET UP "VANILLA FLAVORED" TX/RX.
12996 063624 004737 027530              JSR     PC,CHRMASK      ;GET THE BIT MASK OF UNUSED TX/RX BITS.
12997 063630 004737 032722              JSR     PC,PURIFO      ;PURGE THE DUT RECEIVE CHARACTER FIFO.
12998 063634 004737 033176              JSR     PC,PURRXB      ;PURGE THE RX CHAR BUFFER IN MEMORY.
12999 063640 004737 031442              JSR     PC,INICHR      ;SEND INITIAL CHARS TO ALL ACTIVE LINES.
13000 063644 012737 021465 005466      MOV     #9013.,ERRNBR      ;SET THE ERROR NUMBER TO 9013.
13001      ;
13002      ; The following routine reports the error with numbers 9013 thru 9018.
13003      ;
13004 063652 004737 033232              JSR     PC,RDCHRS      ;READ AND VERIFY THE RX CHARACTERS.
13005 063656 012737 021473 005466      MOV     #9019.,ERRNBR      ;SET THE ERROR REPORT NUMBER TO 9019.
13006      ;
13007      ; The following routine reports the error with numbers 9019 thru 9022.
13008      ;
13009 063664 004737 037154              JSR     PC,TXRREP      ;REPORT FINAL ERRORS FROM RX/RX.
13010      ;
13011      ; Loop to select the next baudrate and line parameters.
13012      ;
13013 063670 000733              BR     8$
13014      ;
13015 063672 012737 021477 005466      10$:    MOV     #9023.,ERRNBR      ;SELECT NUMBER 9023 FOR THE NEXT ERROR REPORT.
13016 063700 004737 034502              JSR     PC,REPSMR      ;REPORT ERROR SUMMARIES IF CALLED FOR.
13017      ;
13018      ;
13019      ; All done, have completed the test.
13020      ; Disable interrupts.
13021      ; Clear the interrupt vectors.
13022      ;
13023 063704 106427 000240      60$:    MTPS     #PRI05      ;DISABLE DEVICE INTERRUPTS.
13024 063710              CLRVEC    TXVECA      ;RETURN TX INT VECTOR TO UNUSED POOL.
                                MOV     TXVECA,R0
                                TRAP     C$CVEC

```

HARDWARE TEST - SHTSNG -

13025	063716			CLRVEC	RXVECA	;RETURN RX INT VECTOR TO UNUSED POOL.	
	063716	013700	002266			MOV	RXVECA,RO
	063722	104436				TRAP	C#CVEC
13026	063724	005037	002362	CLR	CTRLCF	;INDICATE THAT WE ARE NOT WITHIN A TEST.	
13027							
13028	063730			ENDTST			
	063730						
	063730	104401				L10102:	
						TRAP	C#ETST

HARDWARE TEST

SHTDMA -

```

13030 .SBTTL  HARDWARE TEST          - SHTDMA -
13031 ;* *****
13032 ;*          - Short DMA Mode Test -
13033 ;*      This test verifies that the Device Under Test (DUT) will perform
13034 ;*      transmission and reception correctly using the DMA mode transmission.
13035 ;*      This test is performed at 9.6K baud, 5 bits per character, 2 stop bits
13036 ;*      and even parity. A single 512 byte data pattern is transmitted and
13037 ;*      received to/from each active line.
13038 ;*
13039 ; - *****
13040 063732 BGNTST
13041 063732                                T40::
13041 000050      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
13042 063732 012737 000050 002364      MOV    #TNUM,TSTNUM      ;SET UP THE TEST NUMBER.      (91)
13043 063740 012737 177777 002362      MOV    #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
13044 063746 012737 000001 005464      MOV    #1,ERRTYP      ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
13045 063754 012737 021615 005466      MOV    #9101,ERRNBR      ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
13046 063762 012737 023647 005470      MOV    #EM9101,ERRMSG      ;SET ERROR MESSAGE ADDRESS IN ERRtbl.
13047 063770 005037 002650      CLR     ERSRFR      ;INITIALIZE THE "REPORT ERROR SUMMARY" FLAGS.
13048 ;*
13049 ; Reset the DUT to a known state, remove the status codes from the f'fo.
13050 ; Clear TX and RX interrupt enable bits in the CSR.
13051 ; This subroutine reports error >>>> 9101 <<<<.
13052 ; -
13053 063774 004737 030172      JSR     PC,CLNRST      ;RESET THE DHV11-M, REPORT ANY ERRORS FOUND.
13054 064000 103402      BCS     2$      ;SKIP AROUND TEST EXIT IF NO FATAL ERROR FOUND
13055 064002 000137 064270      JMP     60$      ;RESET FAILURE, ABORT THIS TEST.
13056 ;*
13057 ; Set up for Transm t interrupts.
13058 ; -
13059 064006 106427 000240      2$:      MTPS    #PRI05      ;DISABLE DEVICE INTERRUPTS.
13060 064012      SETVEC    TXVECA,#TXDMA,#PRI05      ;SELECT DMA TX INT SERVICE RTN.
13061 064012 012746 000240      MOV     #PRI05,-(SP)
13062 064016 012746 041064      MOV     #TXDMA,-(SP)
13063 064022 013746 002270      MOV     TXVECA,-(SP)
13064 064026 012746 000003      MOV     #3,-(SP)
13065 064032 104437      TRAP    C$SVEC
13066 064034 062706 000010      ADD     #10,SP
13067 064040      SETVEC    RXVECA,#RXCHRS,#PRI05      ;SELECT RX INT SERVICE RTN.
13068 064040 012746 000240      MOV     #PRI05,-(SP)
13069 064044 012746 040570      MOV     #RXCHRS,-(SP)
13070 064050 013746 002266      MOV     RXVECA,-(SP)
13071 064054 012746 000003      MOV     #3,-(SP)
13072 064060 104437      TRAP    C$SVEC
13073 064062 062706 000010      ADD     #10,SP
13074 064066 106427 000000      MTPS    #PRI00      ;ALLOW INTERRUPTS.
13075 ;*
13076 ; Transmit, receive, and check 512 byte data patterns.
13077 ; -
13078 ;*      Initialize the 512 byte pattern and the various data pattern pointers.
13079 ; -
13080 064072 005001      CLR     R1      ;CLEAR THE DATA BYTE COUNTER.
13081 064074 012702 004012      MOV     #BUFBAS,R2      ;GET THE BASE OF THE DATA PATTERN BUFFER.
13082 064100 110122      MOV     R1,(R2)+      ;WRITE A BYTE OF THE DATA PATTERN.
13083 064102 105201      INCB    R1      ;GET THE NEXT BYTE FOR THE DATA PATTERN.
13084 064104 001375      BNE     6$      ;LOOP UNTIL FIRST 1/2 OF PATTERN IS DONE.

```

HARDWARE TEST

- SHTDMA

```

13074 064106 105301      81:  DECB  R1          ;GET THE NEXT BYTE FOR THE DATA PATTERN.
13075 064110 110122      MOVB  R1,(R2)+    ;WRITE A BYTE OF THE DATA PATTERN.
13076 064112 105701      TSTB  R1          ;CHECK FOR DONE WRITING DATA PATTERN.
13077 064114 001374      BNE   81          ;LOOP IF DATA PATTERN IS NOT DONE.
13078 064116 110122      101:  MOVB  R1,(R2)+    ;WRITE A BYTE OF THE 32 BYTE OVERFLOW REGION.
13079 064120 005201      INC   R1          ;COUNT THIS BYTE.
13080 064122 020127 000040  CMP   R1,#32.    ;TEST FOR 32 BYTES WRITTEN.
13081 064126 001373      BNE   101         ;LOOP UNTIL 32 BYTES ARE WRITTEN.
13082
13083      ;*
13084      ; Specify 9.6K baud, 2 stop bits, even parity, and 5 bits per character.
13085      ; Perform DMA transmission and reception of 512 byte data pattern.
13086      ;*
13087      ; The following routine reports the error with numbers 914 thru 921.
13088      ; LPR CHANGE bit error flags may be set by this subroutine.
13089      ;*
13090 064130 012701 156740 121:  MOV   #156740,R1    ;9.6K BAUD,2 STOP, EVEN PARITY, 5 BITS/CHAR.
13091 064134 004737 031264  JSR   PC,GETTIM    ;GET TIME-OUT BASED ON MINIMUM BAUDRATE IN USE.
13092 064140 012702 004012  MOV   #8UBAS,R2    ;SET UP THE START ADR OF THE DATA PATTERN.
13093 064144 012703 001000  MOV   #512.,R3     ;SET UP THE DATA PATTERN LENGTH.
13094 064150 012704 000001  MOV   #1,R4        ;SPECIFY TO SEND 1 DATA PATTERN TO EACH LINE.
13095 064154 004737 037466  JSR   PC,VANS JP    ;SET UP "VANILLA FLAVORED" TX/RX.
13096 064160 012737 177740 002366  MOV   #177740,IBM    ;FORM BIT MAP OF UNUSED BITS FOR 5 BITS/CHAR.
13097 064166 012737 021633 005466  MOV   #9115.,ERRNBR ;SET ERROR NUMBER TO 9115.
13098
13099      ;*
13100      ; This routine reports errors with numbers >>>> 9115 thru 9117 <<<<.
13101 064174 004737 033004  JSR   PC,PUFIFR    ;PURGE THE DUT RECEIVE CHARACTER FIFO.
13102 064200 103033      BCC   601         ;ABORT THIS TEST IF FIFO WOULD NOT PURGE.
13103 064202 012737 021636 005466  MOV   #9118.,ERRNBR ;SET ERROR NUMBER TO 9118.
13104
13105 064210 004737 033176  JSR   PC,PURRXB    ;PURGE THE RX CHAR BUFFER IN MEMORY.
13106 064214 004737 031554  JSR   PC,INIDMA    ;SEND THE FIRST BATCH OF DATA PATTERNS.
13107
13108      ;*
13109      ; This routine reports the error with numbers >>>> 9118 thru 9123 <<<<.
13110 064220 004737 033232  JSR   PC,RDCHRS    ;READ AND VERIFY THE RX CHARACTERS.
13111 064224 012737 021644 005466  MOV   #9124.,ERRNBR ;SET ERROR NUMBER TO 9124.
13112
13113      ;*
13114      ; This routine reports errors with numbers >>>> 9124 thru 9127 <<<<.
13115 064232 004737 037154  JSR   PC,TXRREP    ;REPORT FINAL ERRORS FROM RX/RX.
13116
13117      ;*
13118      ; All done. Have either run out of active lines, or completed the test.
13119      ; Disable interrupts.
13120      ; Clear the interrupt vectors.
13121 064236 106427 000240  MTPS   #PRI05      ;DISABLE DEVICE INTERRUPTS.
13122 064242      CLRVEC  TXVECA    ;RETURN TX INT VECTOR TO UNUSED POOL.
13123 064242 013700 002270  MOV   TXVECA,R0    ;
13124 064246 104436      TRAP   C0CVEC ;
13125 064250      CLRVEC  RXVECA    ;RETURN RX INT VECTOR TO UNUSED POOL.
13126 064250 013700 002266  MOV   RXVECA,R0    ;
13127 064254 104436      TRAP   C0CVEC ;
13128
13129 064256 012737 021666 005466  MOV   #9142.,ERRNBR ;SELECT NUMBER 9142 FOR THE NEXT ERROR REPORT.
13130 064264 004737 034502  JSR   PC,REPSMR    ;REPORT ERROR SUMMARIES IF CALLED FOR.

```


HARDWARE TEST - SHTDMA -

13127	064270	106427	000240	604:	MTPS	#PRI05	;DISABLE DEVICE INTERRUPTS.
13128	064274	005037	002362		CLR	CTRLCF	;INDICATE THAT WE ARE NOT WITHIN A TEST.
13129	064300				ENDTST		
	064300						
	064300	104401					
						L10103:	
						TRAP	C#ETST

HARDWARE TEST

- REPBMP -

```

13131 .SBTTL  HARDWARE TEST          - REPBMP -
13132 ;** *****
13133 ;*
13134 ;*      - Report any BMP codes in the queue -
13135 ;*      This is a pseudo-test used to report any BMP codes that were found
13136 ;*      in the DUT's FIFO during previous test, and logged in the BMP code
13137 ;*      queue.
13138 ;*      It is unlikely that running this pseudo-test alone will produce any
13139 ;*      error reports.
13140 ;*
13141 ; - *****
13141 064302      BGNTST
13141 064302
13142      000051      TNUM == TNUM + 1      ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
13143 064302 012737 000051 002364      MOV  @TNUM,TSTNUM      ;SET UP THE TEST NUMBER. (93)
13144 064310 012737 177777 002362      MOV  #-1,CTRLCF      ;INDICATE THAT WE ARE IN A TEST.
13145 064316 013702 002660      MOV  BMPCQP,R2      ;GET THE CONTENTS OF THE POINTER.
13146 064322 012703 002662      MOV  @BMPCQB,R3      ;GET THE START ADDRESS OF THE QUEUE.
13147 064326 020203      CMP  R2,R3      ;SEE IF THE POINTER HAS MOVED FROM THE BASE.
13148 064330 001411      BEQ  60$      ;EXIT NO CODES IN THE QUEUE.
13149
13150 ;*
13151 ; There is at least one BMP code in the queue. Report the error
13152 ; -
13153 ;Report error BMP CODE FOUND IN TEST nn, BMP CODE:nnnnnn"
13154 064332 012701 024163      MOV  @EM9304,R1      ;PASS THE FIRST MESSAGE TO BE REORTED.
13155 064336      ERRDF  9301,EM9301,ER9301 ;      >>>>> ERROR 09301 <<<<<.
13155      064336 104455      TRAP  C$ERDF
13155      064340 022125      .WORD  9301
13155      064342 024046      .WORD  EM9301
13155      064344 026600      .WORD  ER9301
13156
13157 064346 012737 002662 002660      MOV  @BMPCQB,BMPCQP ;SET POINTER BACK TO THE BEGINING OF THE QUE.
13158
13159 064354 005037 002362      60$: CLR  CTRLCF      ;INDICATE THAT WE ARE NOT WITHIN A TEST.
13160 064360      ENDTST
13160      064360 104401      L10104:
13160      064360      TRAP  C$ETST

```


HARDWARE TEST

REP8MP

```

13163 ;*****
13164 ;
13165 ;          VDME.HWQ
13166 ;
13167 ;*****
13169
13170
13171 .SBTTL  HARDWARE PARAMETER CODING SECTION
13172
13173
13174
13175 ;**
13176 ; THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
13177 ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES  THE
13178 ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
13179 ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
13180 ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
13181 ; WITH THE OPERATOR.
13182 ;**
13183
13184 064362          BGNHRD
13185 064362 000015
13186 064364
13187
13188 ;DEVICE CSR ADDRESS QUESTION:
13189 GPRMA  HWPTQ1.0.0.160000.177776.YES
13190
13191 .WORD  L10105-L$HARD/2
13192 L$HARD::
13193
13194 064364 000031
13195 064366 064416
13196 064370 160000
13197 064372 177776
13198
13199 ;DEVICE INTERRUPT VECTOR QUESTION:
13200 CPRMA  HWPTQ2.2.0.40.776.YES
13201
13202 .WORD  T$CODE
13203 .WORD  HWPTQ1
13204 .WORD  T$LOLIM
13205 .WORD  T$HILIM
13206
13207 ;DEVICE INTERRUPT VECTOR QUESTION:
13208 CPRMA  HWPTQ2.2.0.40.776.YES
13209
13210 .WORD  T$CODE
13211 .WORD  HWPTQ2
13212 .WORD  T$LOLIM
13213 .WORD  T$HILIM
13214
13215 ;INTERRUPT BR  EVEL QUESTION:
13216 GPRMD  HWPTQ5.6.0.177400.0.6.YES
13217
13218 .WORD  T$CODE
13219 .WORD  HWPTQ5
13220 .WORD  177400
13221 .WORD  T$LOLIM
13222 .WORD  T$HILIM
13223
13224 13201
13225 13202
13226 13203 064416          ENDRD
13227
13228 064416
13229
13230 L10105: .EVEN
13231
13232 13204
13233 13211
13234 13212 064416      103      123      122      HWPTQ1: .ASCIZ  /CSR ADDRESS: /
13235 064421      040      101      104
13236 064424      104      122      105
13237 064427      123      123      072
13238 064432      040      000
13239 13213 064434      111      116      124      HWPTQ2: .ASCIZ  /INTERRUPT VECTOR ADDRESS: /
13240 064437      105      122      122

```

HARDWARE PARAMETER CODING SECTION

	064442	125	120	124
	064445	040	126	105
	064450	103	124	117
	064453	122	040	101
	064456	104	104	122
	064461	105	123	123
	064464	072	040	000
13214	064467	111	116	124
	064472	105	122	122
	064475	125	120	124
	064500	040	102	122
	064503	040	114	105
	064506	126	105	114
	064511	072	040	000

13215
13216

HWPTQ5: .ASCIZ /INTERRUPT BR LEVEL: /

.EVEN

HARDWARE PARAMETER CODING SECTION

```
13219 ;*****
13220 ;
13221 ;          VDHE.SWQ
13222 ;
13223 ;*****
13225
13226
13227 .SBTTL SOFTWARE PARAMETER CODING SECTION
13228
13229 ;**
13230 ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
13231 ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
13232 ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
13233 ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
13234 ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
13235 ; WITH THE OPERATOR.
13236 ;--
13237
13238 ; NO SOFTWARE QUESTIONS ARE INCLUDED.
13239 ; BGNSFT
13240
13249 ; ENDSFT
13250
13251
13258 .EVEN
```

13260
13261
13262
13263
13264
13265
13266
13267
13268
13269
13270
13271
13278
13279
13280
13281
13282

13283
13284
13285
13286
13287
13288
13289
13290
13291

```

:*****:
:
:      FVTSKL6.P11
:
:*****:

```

\$PATCH::
 .BLKW 24

LASTAD

```
.EVEN
.WORD    0
.WORD    0
```

```

L$LAST::
                                ENDMOD

```

000001

.END

Symbol table

ACTLNS	002272 G	CBLPRA	003272 G	C#GPHR=	000042	EF.STA=	000040 G	EM0701	013336 G
ADDR	036032	CBMAPA	003304 G	C#GPRI=	000040	EF0503	005630 G	EM0702	013373 G
ADR	= 000020 G	CBOFSA	003310 G	C#INIT=	000011	EF0505	005635 G	EM0703	013553 G
ADRPTR	030124 G	CHCNTB	003612 G	C#INLP=	000020	EF1401	005710 G	EM0801	013736 G
ALTFLD	026760 G	CHKBMP	027256 G	C#MANI=	000050	EF1402	006012 G	EM0802	013776 G
ASSEMB=	000010	CHKEXT	027326 G	C#MAP =	000102	EF1601	006047 G	EM0901	014051 G
BCOUNT	002456 G	CHKLOS	027426 G	C#MEM =	000031	EF1602	006073 G	EM0902	014102 G
BITLNG	002426 G	CHRMASK	027530 G	C#MMU =	000103	EF1603	006135 G	EM1001	014136 G
BITSTD	063334	CHRTOT	002646 G	C#MSG =	000023	EF1604	006177 G	EM1002	014202 G
BITTBL	002534 G	CKCHR	027566 G	C#OPNR=	000034	EF3001	006274 G	EM1003	014267 G
BIT0	= 000001 G	CKINAC	030004 G	C#OPNW=	000104	EF3002	006343 G	EM1101	014345 G
BIT00	= 000001 G	CKIRAP	030112 G	C#PNTB=	000014	EF4401	006412 G	EM1201	014371 G
BIT01	= 000002 G	CKTRPB	030142 G	C#PNTF=	000017	EF6401	006527 G	EM1202	014410 G
BIT02	= 000004 G	CLKBRL	002442 G	C#PNTS=	000016	EF7801	006604 G	EM1203	014474 G
BIT03	= 000010 G	CLKCSR	002440 G	C#PNTX=	000015	EF8401	006642 G	EM1204	014552 G
BIT04	= 000020 G	CLKHRZ	002446 G	C#PUTB=	000072	EF8402	006734 G	EM1205	014606 G
BIT05	= 000040 G	CLKINT	040430 G	C#PUTW=	000073	EF9001	007051 G	EM1301	014632 G
BIT06	= 000100 G	CLKVEC	002444 G	C#QIO =	000377	EF9002	007133 G	EM1302	014656 G
BIT07	= 000200 G	CLNRST	030172 G	C#RDBU=	000007	EF9003	007212 G	EM1401	014715 G
BIT08	= 000400 G	CLR16W	030214 G	C#REFG=	000047	EF9004	007246 G	EM1402	014745 G
BIT09	= 001000 G	CNTERR	030236 G	C#REL =	000077	EF9005	007303 G	EM1403	015033 G
BIT1	= 000002 G	CONMAP	030314 G	C#RESE=	000033	EF9006	007334 G	EM1404	015106 G
BIT10	= 002000 G	CSRA	002300 G	C#REVI=	000004	EF9007	007360 G	EM1405	015160 G
BIT11	= 004000 G	CSRO	= 000000 G	C#RFLA=	000021	EF9008	007454 G	EM1406	015173 G
BIT12	= 010000 G	CTRLCF	002362 G	C#RPT =	000025	EF9009	007513 G	EM1407	015206 G
BIT13	= 020000 G	C#AU =	000052	C#SEFG=	000046	EF9010	007552 G	EM1408	015220 G
BIT14	= 040000 G	C#AUTO=	000061	C#SPRI=	000041	EF9012	007651 G	EM1601	015226 G
BIT15	= 100000 G	C#BRK =	000022	C#SVEC=	000037	EF9013	007765 G	EM1604	015311 G
BIT2	= 000004 G	C#BSEG=	000004	C#TOME=	000076	EF9016	010032 G	EM1701	015357 G
BIT3	= 000010 G	C#BSUB=	000002	DELAY	030370 G	EF9017	010127 G	EM1801	015434 G
BIT4	= 000020 G	C#CLCK=	000062	DEST	063340	EF9018	010203 G	EM1901	015502 G
BIT5	= 000040 G	C#CLEA=	000012	DFPTBL	002250 G	EF9019	010310 G	EM2001	015557 G
BIT6	= 000100 G	C#CLOS=	000035	DIAGMC=	000000	EF9020	010334 G	EM2002	015621 G
BIT7	= 000200 G	C#CLP1=	000006	DMRW	030502 G	EF9101	010415 G	EM2101	015674 G
BIT8	= 000400 G	C#CPBF=	000074	DMTSTA	002432 G	EF9103	010420 G	EM2102	015737 G
BIT9	= 001000 G	C#CPME=	000075	DM16B	030430 G	EF9301	010466 G	EM2201	016033 G
BMPCQB	002662 G	C#CVEC=	000036	DODMA	030624 G	EF9302	010544 G	EM2202	016070 G
BMPCQE	003062 G	C#DCLN=	000044	DPENDB	003312 G	EM0101	032456 G	EM2203	016162 G
BMPCQP	002660 G	C#DODU=	000051	DPLENB	003352 G	EM0102	032542 G	EM2301	016267 G
BOE	= 000400 G	C#DRPT=	000024	DPRSQB	005052 G	EM0103	010727 G	EM2302	016325 G
BRLEVL	002275 G	C#DU =	000053	DPRSQE	005252 G	EM0201	010765 G	EM2401	016363 G
BRTBLB	002574 G	C#EDIT=	000000	DRADRT	002300 G	EM0202	011033 G	EM2601	016417 G
BRTBLE	002634 G	C#ERDF=	000055	DROP	042352	EM0203	011206 G	EM2602	016447 G
BUFBAS	004012 G	C#ERHR=	000056	DR00MG	010644 G	EM0204	011351 G	EM2603	016540 G
BUFEND	005012 G	C#ERRO=	000060	DR02MG	010650 G	EM0301	011530 G	EM2604	016626 G
BUFMID	004412 G	C#ERSF=	000054	DR04MG	010655 G	EM0302	011573 G	EM2605	016722 G
BUFPTR	002360 G	C#ERSO=	000057	DR06MG	010661 G	EM0303	011733 G	EM2606	017005 G
BUF3QT	004612 G	C#ESCA=	000010	DR10MG	010666 G	EM0401	012112 G	EM2607	017044 G
CACHRX	040354 G	C#ESEG=	000005	DR12MG	010675 G	EM0402	012161 G	EM2608	017140 G
CACHTX	040402 G	C#ESUB=	000003	DR14MG	010706 G	EM0501	012331 G	EM2609	017211 G
CALMSL	027032 G	C#ETST=	000001	DR16MG	010717 G	EM0502	012377 G	EM2610	017270 G
CBB	003272 G	C#EXIT=	000032	DUMY	063336	EM0509	012553 G	EM2611	017362 G
CBDPAA	003276 G	C#FREQ=	000101	EDROP	042430	EM0525	012557 G	EM3001	017445 G
CBDPLA	003300 G	C#FRME=	000100	EF.CON=	000036 G	EM0526	012647 G	EM3002	017476 G
CBDPNA	003302 G	C#GETB=	000026	EF.NEW=	000035 G	EM0601	012737 G	EM3003	017572 G
CBLNCA	003274 G	C#GETW=	000027	EF.PWR=	000034 G	EM0602	012773 G	EM3004	017646 G
CBLPBA	003306 G	C#GMAN=	000043	EF.RES=	000037 G	EM0603	013153 G	EM3005	017722 G

Symbol table

EM3101	020014	G	EM9024	023275	G	F\$MOD	=	000000	T\$INIT	=	000041	L\$LADP	002026	G					
EM3102	020050	G	EM9025	023313	G	F\$MSG	=	000011	I\$MOD	=	000041	L\$LAST	064570	G					
EM4001	020111	G	EM9026	023407	G	F\$PROT	=	000021	I\$MSG	=	000041	L\$LOAD	002100	G					
EM4002	020134	G	EM9027	023433	G	F\$PWR	=	000017	I\$PROT	=	000040	L\$LUN	002074	G					
EM4101	020170	G	EM9028	023513	G	F\$RPT	=	000012	I\$PTAB	=	000041	L\$MREV	002050	G					
EM4102	020213	G	EM9030	023572	G	F\$SEG	=	000003	I\$PWR	=	000041	L\$NAME	002000	G					
EM4103	020247	G	EM9101	023647	G	F\$SOFT	=	000005	I\$RPT	=	000041	L\$PRIO	002042	G					
EM4401	020333	G	EM9102	023703	G	F\$SRV	=	000010	I\$SEG	=	000041	L\$PROT	041466	G					
EM4402	020355	G	EM9104	023772	G	F\$SUB	=	000002	I\$SETU	=	000041	L\$PRT	002112	G					
EM4403	020423	G	EM9301	024046	G	F\$SW	=	000014	I\$SRV	=	000041	L\$REPP	002062	G					
EM4404	020520	G	EM9302	024066	G	F\$TEST	=	000001	I\$SUB	=	000041	L\$REV	002010	G					
EM4405	020557	G	EM9303	024116	G	GETCHR	031052	G	I\$TST	=	000041	L\$RPT	041460	G					
EM4406	020653	G	EM9304	024163	G	GETLP2	031134	G	J\$JMP	=	000167	L\$SPC	002056	G					
EM4407	020727	G	ENDETB	005012	G	GETPRM	042074		LGRP1M	002370	G	L\$SPCP	002020	G					
EM4408	021011	G	ENDIT	042246		GETTIM	031264	G	LGRP2M	002372	G	L\$STP	002024	G					
EM4409	021054	G	ERCNTB	003452	G	GMANWD	002434	G	LINBIT	031646	G	L\$STA	002030	G					
EM4410	021111	G	ERLTBL	004012	G	GPRS0B	002634	G	LNCTRA	002310	G	L\$SW	002262	G					
EM4411	021140	G	ERRBLK	005472	G	G\$CNT0	=	000200	LNCTRO	=	000010	L\$TEST	002114	G					
EM5101	021205	G	ERRMSG	005470	G	G\$DELM	=	000372	LOE	=	040000	L\$TIML	002014	G					
EM5102	021233	G	ERRNBR	005466	G	G\$DISP	=	000003	LOPBCK	002274	G	L\$UNIT	002012	G					
EM5103	021271	G	ERRTYP	005464	G	G\$EXCP	=	000400	LOT	=	000010	L10000	002260						
EM5201	021321	G	ERSMRF	002650	G	G\$HILI	=	000002	LPCSLT	=	000036	L10001	002266						
EM5202	021345	G	ER0101	024240	G	G\$LOLI	=	000001	LPRA	002304	G	L10002	024342						
EM5301	021403	G	ER0201	024562	G	G\$NO	=	000000	LPRO	=	000004	L10003	024642						
EM5302	021430	G	ER0503	024644	G	G\$OFFS	=	000400	L\$ACP	002110	G	L10004	024666						
EM5303	021507	G	ER0504	024670	G	G\$OF SI	=	000376	L\$APT	002036	G	L10005	024734						
EM5401	021560	G	ER1601	024736	G	G\$PRMA	=	000001	L\$AU	042436	G	L10006	025032						
EM5402	021617	G	ER1603	025034	G	G\$PRMD	=	000002	L\$AUT	002070	G	L10007	025112						
EM5501	021660	G	ER3001	025114	G	G\$PRML	=	000000	L\$AUTO	042274	G	L10010	025202						
EM5601	021715	G	ER6401	025204	G	G\$RADA	=	000140	L\$CCP	002106	G	L10011	025300						
EM5701	021763	G	ER7801	025302	G	G\$RADB	=	000000	L\$CLEA	042276	G	L10012	025326						
EM6401	022031	G	ER9001	025330	G	G\$RADD	=	000040	L\$CO	002032	G	L10013	025416						
EM6402	022060	G	ER9002	025420	G	G\$RADL	=	000120	L\$DEPO	002011	G	L10014	025564						
EM6601	022121	G	ER9003	025566	G	G\$RADO	=	000020	L\$DESC	005544	G	L10015	025746						
EM6602	022147	G	ER9004	025750	G	G\$XFER	=	000004	L\$DESP	002076	G	L10016	026050						
EM6701	022221	G	ER9005	026052	G	G\$YES	=	000010	L\$DEVP	002060	G	L10017	026276						
EM6702	022244	G	ER9007	026300	G	HELP	=	000000	L\$DISP	002124	G	L10020	026354						
EM9001	022321	G	ER9008	026356	G	HOE	=	100000	L\$DLY	002116	G	L10021	026436						
EM9003	022355	G	ER9101	026440	G	HWPTQ1	064416		L\$DTP	002040	G	L10022	026464						
EM9004	022377	G	ER9102	026466	G	HWPTQ2	064434		L\$DTYP	002034	G	L10023	026576						
EM9006	022415	G	ER9301	026600	G	HWPTQ5	064467		L\$DU	042326	G	L10024	026756						
EM9007	022470	G	EVL	=	000004	G	IBE	=	010000	G	L\$DUT	002072	G	L10025	041464				
EM9008	022553	G	EXCNTB	003412	G	IBM	002366	G	L\$DVTY	005534	G	L10027	042272		L10030	042274			
EM9009	022634	G	E\$END	=	002100		IDU	=	000040	G	L\$EF	002052	G	L10031	042324				
EM9010	022660	G	E\$LOAD	=	000035		IER	=	020000	G	L\$ENVI	002044	G	L10032	042434				
EM9011	022704	G	FFREM	002430	G	IESTAT	002374	G	L\$ERRT	005464	G	L10033	042442		L10034	042732			
EM9012	022714	G	FINACT	030772	G	INDATP	031342	G	L\$ETP	002102	G	L10035	043146		L10036	043376			
EM9013	022724	G	F\$AU	=	000015		INDTPX	031372	G	L\$EXP1	002046	G	L10037	043560		L10040	043736		
EM9014	022733	G	F\$AUTO	=	000020		INICHR	031442	G	L\$EXP4	002064	G	L10041	044140		L10042	044332		
EM9015	023027	G	F\$BGN	=	000040		INIDMA	031554	G	L\$EXP5	002066	G	L10043	044524		L10044	044676		
EM9016	023043	G	F\$CLEA	=	000007		ISR	=	000100	G	L\$HARD	064364	G	L10045	045076				
EM9017	023052	G	F\$DU	=	000016		IXE	=	004000	G	L\$HIME	002120	G						
EM9018	023163	G	F\$END	=	000041		I\$AU	=	000041		L\$HPCP	002016	G						
EM9019	023173	G	F\$HARD	=	000004		I\$AUTO	=	000041		L\$HPTP	002022	G						
EM9020	023210	G	F\$HW	=	000013		I\$CLN	=	000041		L\$HW	002250	G						
EM9022	023234	G	F\$INIT	=	000006		I\$DU	=	000041		L\$ICP	002104	G						
EM9023	023253	G	F\$JMP	=	000050		I\$HRD	=	000041		L\$INIT	041474	G						

Symbol table

L10046	045310	0\$AU	=	000000	RESETT	034530	G	SVCGBL	=	000000	T\$HILI	=	000006
L10047	045556	0\$BGNR	=	000000	RMATBB	002340	G	SVCINS	=	000001	T\$LAST	=	000001
L10050	046120	0\$BGNS	=	000000	ROLDAP	034642	G	SVCSUB	=	000001	T\$LOLI	=	000000
L10051	046330	0\$DU	=	000001	RRXNDN	034674	G	SVCTAG	=	000001	T\$LSYM	=	010000
L10052	046544	0\$ERRT	=	000001	RSTRPT	034734	G	SVCTST	=	000001	T\$LTNO	=	000051
L10053	047010	0\$GNSW	=	000001	RTXNDN	035316	G	SWAPO	=	036036	T\$NEST	=	177777
L10054	047260	0\$POIN	=	000001	RXBCNT	003066	G	S\$LSYM	=	010000	T\$NS0	=	000000
L10055	047366	0\$SETU	=	000000	RXBDTX	000030	G	TIMER1	=	002450	T\$NS1	=	000004
L10056	047576	PARATB		002474	RXBEND	003270	G	TIMER2	=	002452	T\$PTNU	=	000000
L10057	050032	PARATE		002514	RXBETX	000020	G	TIMER3	=	002454	T\$SAVL	=	177777
L10060	050332	PAROA		002474	RXBFUL	000100	G	TNUM	=	000051	T\$SEGL	=	177777
L10061	050670	PAR1A		002476	RXBIPT	003064	G	TP4BRT	=	041020	T\$SUBN	=	000000
L10062	051702	PAR2A		002500	RXBOPT	003062	G	TP4FLG	=	002424	T\$TAGL	=	177777
L10063	052554	PAR3A		002502	RXBRRT	040500	G	TP4RTN	=	041042	T\$TAGN	=	010106
L10064	053032	PAR4A		002504	RXBSTA	003070	G	TP4VEC	=	002422	T\$TEMP	=	000000
L10065	053420	PAR5A		002506	RXCHRS	040570	G	TRPAD2	=	030154	T\$TEST	=	000051
L10066	054040	PAR6A		002510	RXCNTB	003712	G	TSABRT	=	036106	T\$TSTM	=	177777
L10067	054434	PAR7A		002512	RXDONF	002654	G	TSTNUM	=	002364	T\$TSTS	=	000001
L10070	055056	PASCNT		002376	RXDSBL	035356	G	TXAD1A	=	002312	T\$AU	=	010033
L10071	055346	PCSL0T	=	000016	RXENBL	035452	G	TXAD10	=	000012	T\$AUT	=	010030
L10072	055650	PDRATB		002514	RXIE0	035546	G	TXAD2A	=	002314	T\$CLE	=	010031
L10073	056342	PDRATE		002534	RXIE1	035600	G	TXAD20	=	000014	T\$DU	=	010032
L10074	057032	PDR0A		002514	RXINPT	040734	G	TXBFCA	=	002316	T\$HAR	=	010105
L10075	057476	PDR1A		002516	RXINTC	002412	G	TXBFCO	=	000016	T\$HW	=	010000
L10076	060122	PDR2A		002520	RXINTF	002414	G	TXCHA	=	002302	T\$INI	=	010027
L10077	060622	PDR3A		002522	RXPTRB	003552	G	TXCHRO	=	000002	T\$MSG	=	010024
L10100	061550	PDR4A		002524	RXTOUT	002402	G	TXCNTB	=	003652	T\$PRO	=	010026
L10101	063346	PDR5A		002526	RXVECA	002266	G	TXDATP	=	036220	T\$RPT	=	010025
L10102	063730	PDR6A		002530	ROSLOT	000002	G	TXDBLF	=	002656	T\$SW	=	010001
L10103	064300	PDR7A		002532	R1SLOT	000004	G	TXDMA	=	041064	T\$TES	=	010104
L10104	064360	PMSFLG		002436	R2SLOT	000006	G	TXDONE	=	036242	T1	=	042444
L10105	064416	PNT	=	001000	R3SLOT	000010	G	TXDONF	=	002652	T10	=	044700
MAPCNT	031674	PREGRT		005516	R4SLOT	000012	G	TXDSBL	=	036352	T11	=	045100
MAPLNS	=	PREG05		005474	R5SLOT	000014	G	TXENBL	=	036446	T12	=	045312
MFUNIT	005572	PRI	=	002000	SAVBMP	035624	G	TXENBM	=	002410	T13	=	045560
MMENAB	002472	PRI00	=	000000	SAVPRI	002406	G	TXIE0	=	036542	T14	=	046122
MMPRES	002470	PRI01	=	000040	SAVTEN	002404	G	TXIE1	=	036574	T15	=	046332
MMSRO	002464	PRI02	=	000100	SCBCTB	005252	G	TXINTC	=	002416	T16	=	046546
MMSR3	002466	PRI03	=	000140	SCBCTE	005254	G	TXINTF	=	002420	T17	=	047012
MSG1	024344	PRI04	=	000200	SCNSTB	005254	G	TXINTR	=	041216	T18	=	047262
MSG2	024422	PRI05	=	000240	SCNSTE	005256	G	TXPTRB	=	003512	T19	=	047370
MSG3	024501	PRI06	=	000300	SCTPTB	005256	G	TXRINI	=	036620	T2	=	042734
MSLCNT	002462	PRI07	=	000340	SCTPTE	005260	G	TXRLNB	=	005424	T20	=	047600
MSLGET	031726	PROTBL		005344	SDPBAS	005260	G	TXRLNE	=	005444	T21	=	050034
MS' OOP	032042	PRTLPR		032640	SDPEND	005300	G	TXROFF	=	037074	T22	=	050334
MSTICK	002460	PUFIFO		032722	SDP2B	005304	G	TXRON	=	037130	T23	=	050672
MUL16U	032056	PUFIFR		033004	SDP2E	005324	G	TXRREP	=	037154	T24	=	051704
NDERPT	002264	PURRXB		033176	SETPAR	035672	G	TXRXLB	=	005364	T25	=	052556
NEWCHR	032132	RBUFA		002302	SFPTBL	002262	G	TXRXLE	=	005424	T26	=	053034
NEWPAS	042054	RBUFO	=	000002	SKPSTS	035740	G	TXSCHR	=	041272	T27	=	053422
NEWRES	042046	RDCHRS		033232	STATA	002306	G	TXVECA	=	002270	T28	=	054042
NEWSTA	041540	RDMAST		033604	STATO	=	000006	T\$ARGC	=	000003	T29	=	054436
NUMLNS	=	RDPDR		033636	STGTRB	005444	G	T\$CODE	=	003032	T3	=	043150
ODTSTA	063342	READBX		034056	STPSW	036016	G	T\$ERRN	=	022125	T30	=	055060
OOPS	032412	REGTST		034140	STSTB	003752	G	T\$EXCP	=	000000	T31	=	055350
OPTION	002262	REPCOD		034350	STSTE	004012	G	T\$FLAG	=	000040	T32	=	055652
0\$APTS	=	REPSMR		034502	SUCSS	063344		T\$GMAN	=	000000	T33	=	056344

Symbol table

T34	057034 G	T40	063732 G	UAM	= 000200 G	WAIBIS	037740 G	WTWLPR	040324 G
T35	057500 G	T41	064302 G	UNBTB	002320 G	WAITTX	040014 G	X\$ALWA=	000000
T36	060124 G	T5	043562 G	UNITN	002276 G	WOPDR	040054 G	X\$FALS=	000040
T37	060624 G	T6	043740 G	UNSDIV	037234 G	WORD1	002400 G	X\$OFFS=	000400
T38	061552 G	T7	044142 G	UPDCHR	037370 G	WTWLNC	040250 G	X\$TRUE=	000020
T39	063350 G	T8	044334 G	VANSUP	037466 G	WTWLNS	040300 G	\$PATCH	064514 G
T4	043400 G	T9	044526 G	WAIBIC	037664 G				

. ABS. 064570 000 (RW,I,GBL,ABS,OVR)
000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 296
Work file writes: 312
Size of work file: 34680 Words (136 Pages)
Size of core pool: 19714 Words (75 Pages)
Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:59:22.86
CVDHEBO.BIN,CVDHEBO.LST/-SP=SVC40/ML,CVDHEBO.P11