IDENTIFICATION

PRODUCT CODE: AC 17948 MC
PRODUCT NAME: CZDUBO DMU-11 FUNC TST PART1
PRODUCT DATE: 3 MARCH 1984
MAINTAINER: ENE DIAGNOSTICS GROUP
AUTHOR: ANTHONY HART
MODIFIED BY: ANTHONY HART

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) 1984 BY DIGITAL EQUIPMENT CORPORATION
THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL PDP UNIBUS MASSBUS
DEC DECUS DECTAPE
***** MODIFICATION HISTORY *****

ORIGINAL RELEASE: 15 DEC 83 ANTHONY HART

VERSION 80 3 MAR-84 ANTHONY HART

TWO NEW TESTS WERE INCLUDED IN THIS PART:

TEST 15 - CSR BIT 4 TEST.
TEST 25 - DIAGNOSTIC FIELD (BMP) TEST.
# Table of Contents

1.0 GENERAL PROGRAM CONSIDERATIONS
   1.1 PROGRAM ABSTRACT
   1.2 SYSTEM REQUIREMENTS
   1.3 RELATED DOCUMENTS AND STANDARDS
   1.4 DIAGNOSTIC HIERARCHY PREREQUISITES
   1.5 ASSUMPTIONS

2.0 OPERATING INSTRUCTIONS
   2.1 COMMANDS
   2.2 SWITCHES
   2.3 FLAGS
   2.4 EXTENDED COMMAND SYNTAX
     2.4.1 START COMMAND
     2.4.2 TESTS SWITCH (/TESTS:<TEST-LIST>)
     2.4.3 PASS SWITCH (/PASS:<PASS-CNT>)
     2.4.4 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)
     2.4.5 END OF PASS SWITCH (/EOP:<INCR>)
     2.4.6 EFFECT OF START COMMAND
     2.4.7 TESTS, PASS, AND FLAGS SWITCHES
     2.4.8 UNITS SWITCH (/UNITS:<UNIT-LIST>)
     2.4.9 EFFECT OF RESTART COMMAND
     2.4.10 CONTINUE COMMAND
     2.4.11 EFFECT OF CONTINUE COMMAND
     2.4.12 PROCEED COMMAND
     2.4.13 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)
     2.4.14 EFFECT OF PROCEED COMMAND
     2.4.15 ADD COMMAND
     2.4.16 EFFECT OF ADD COMMAND
     2.4.17 DROP COMMAND
     2.4.18 EFFECT OF DROP COMMAND
     2.4.19 PRINT COMMAND
     2.4.20 EFFECT OF PRINT COMMAND
     2.4.21 DISPLAY COMMAND
     2.4.22 EFFECT OF DISPLAY COMMAND
     2.4.23 FLAGS COMMAND
     2.4.24 EFFECT OF FLAGS COMMAND
     2.4.25 ZFLAGS COMMAND
     2.4.26 ZFLAGS COMMAND
     2.4.27 CONTROL CHARACTERS
     2.5 HARDWARE QUESTIONS
     2.6 SOFTWARE QUESTIONS
     2.7 EXTENDED P- TABLE DIALOGUE
     2.8 QUICK START-UP PROCEDURE (XXDP+)

3.0 ERROR INFORMATION
   3.1 TYPES OF ERROR MESSAGES
   3.2 SPECIFIC ERROR MESSAGES

4.0 PERFORMANCE AND PROGRESS REPORTS

5.0 TEST SUMMARIES

6.0 EXAMPLE ERROR FREE PASS
1.0 GENERAL PROGRAM CONSIDERATIONS

1.1 PROGRAM ABSTRACT

CZDMUBO IS PART ONE OF THE DHU FUNCTIONAL VERIFICATION TEST. THIS PART OF THE TEST VERIFIES THE RESET, SELFTEST, REGISTER ACCESS, BMP CODE, AND INTERRUPT FUNCTIONS OF THE BOARD ARE FUNCTIONING CORRECTLY.

THIS DIAGNOSTIC HAS BEEN WRITTEN FOR USE WITH THE DIAGNOSTIC RUNTIME SERVICES SOFTWARE (SUPERVISOR). THESE SERVICES PROVIDE THE INTERFACE TO THE OPERATOR AND TO THE SOFTWARE ENVIRONMENT. THIS PROGRAM CAN BE USED WITH XBDP+, ACT, APT, SLIDE AND PAPER TAPE. FOR A COMPLETE DESCRIPTION OF THE RUNTIME SERVICES, REFER TO THE XBDP+ 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 DHU11 FVT:

- UNIBUS PROCESSOR WITH AT LEAST 32K BYTES OF MEMORY.
- DHU BOARDS INSTALLED ON THE UNIBUS.
- APPROPRIATE PROGRAM LOAD DEVICE SUPPORTING XBDP+, MEDIA OR A DOWN LINE LOADING SYSTEM.

1.3 RELATED DOCUMENTS AND STANDARDS

- XBDP+ USER'S MANUAL - DESCRIBES THE RUNNING OF DIAGNOSTICS UNDER THE XBDP+ MONITOR.

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

2.0 OPERATING INSTRUCTIONS

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

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.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>START</td>
<td>Start the diagnostic from an initial state</td>
</tr>
<tr>
<td>RESTART</td>
<td>Start the diagnostic without initializing</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>Continue at test that was interrupted (after +C)</td>
</tr>
<tr>
<td>PROCEED</td>
<td>Continue from an error halt</td>
</tr>
<tr>
<td>EXIT</td>
<td>Return to XXDP+ monitor (XXDP+ operation only!)</td>
</tr>
<tr>
<td>ADD</td>
<td>Activate a unit for testing (all units are considered to be active at start time)</td>
</tr>
<tr>
<td>DROP</td>
<td>Deactivate a unit</td>
</tr>
<tr>
<td>PRINT</td>
<td>Print statistical information (if implemented by the diagnostic - see performance and progress reports section of this document)</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>Type a list of all device information</td>
</tr>
<tr>
<td>FLAGS</td>
<td>Type the state of all flags (see flags section)</td>
</tr>
<tr>
<td>ZFLAGS</td>
<td>Clear all flags (see flags section)</td>
</tr>
</tbody>
</table>

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".

<table>
<thead>
<tr>
<th>SWITCH</th>
<th>EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>/TESTS:LIST</td>
<td>Execute only those tests specified in the list. List is a string of test numbers, for example /TESTS:1,5,7,10. This list will cause tests 1,5,7,8,9,10 to be run. All other tests will not be run.</td>
</tr>
<tr>
<td>/PASS:DDDDD</td>
<td>Execute DDDD passes (DDDDD = 1 to 64000)</td>
</tr>
<tr>
<td>/FLAGS:FLGS</td>
<td>Set specified flags. See the flags section of this document.</td>
</tr>
<tr>
<td>/EOP:DDDDD</td>
<td>Report end of pass message after every DDDD passes only. (DDDDD = 1 to 64000)</td>
</tr>
<tr>
<td>/UNITS:LIST</td>
<td>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)</td>
</tr>
</tbody>
</table>

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 "/TEST:1-5" instead of "/TESTS:1-5".

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

<table>
<thead>
<tr>
<th>TESTS</th>
<th>PASS</th>
<th>FLAGS</th>
<th>EOP</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RESTART</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>PROCEED</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DROP</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRINT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLAGS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2FLAGS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXIT</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>FLAG</th>
<th>EFFECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>DME</td>
<td>HALT ON ERROR CONTROL IS RETURNED TO RUNTIME SERVICES COMMAND MODE</td>
</tr>
<tr>
<td>LOE</td>
<td>LOOP ON ERROR</td>
</tr>
<tr>
<td>IER*</td>
<td>INHIBIT ALL ERROR REPORTS</td>
</tr>
<tr>
<td>IER*</td>
<td>INHIBIT ALL ERROR REPORTS EXCEPT FIRST LEVEL (FIRST LEVEL CONTAINS ERROR TYPE, NUMBER, PC, TEST AND UNIT)</td>
</tr>
<tr>
<td>IXR*</td>
<td>INHIBIT EXTENDED ERROR REPORTS (THOSE CALLED BY PRINTX MACRO'S)</td>
</tr>
<tr>
<td>PRI</td>
<td>DIRECT MESSAGES TO LINE PRINTER</td>
</tr>
<tr>
<td>PNT</td>
<td>PRINT TEST NUMBER AS TEST EXECUTES</td>
</tr>
<tr>
<td>BOE</td>
<td>&quot;BELL&quot; ON ERROR</td>
</tr>
<tr>
<td>UAM</td>
<td>UNATTENDED MODE (NO MANUAL INTERVENTION)</td>
</tr>
<tr>
<td>ISR</td>
<td>INHIBIT STATISTICAL REPORTS (DOES NOT APPLY TO DIAGNOSTICS WHICH DO NOT SUPPORT STATISTICAL REPORTING)</td>
</tr>
<tr>
<td>IDR</td>
<td>INHIBIT PROGRAM DROPPING OF UNITS</td>
</tr>
<tr>
<td>ADR</td>
<td>EXECUTE AUTODROP CODE</td>
</tr>
<tr>
<td>LOT</td>
<td>LOOP ON TEST</td>
</tr>
<tr>
<td>EVL</td>
<td>EXECUTE EVALUATION (ON DIAGNOSTICS WHICH HAVE EVALUATION SUPPORT)</td>
</tr>
</tbody>
</table>

*SEE THE ERROR INFORMATION SECTION OF THIS DOCUMENT.

See the xdp+ 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:loei:ier:boe
2.4 EXTENDED COMMAND SYNTAX

2.4.1 START COMMAND -

START(/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 (15:8:10 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>*, or <flag>*0, separated by colons, where <flag> has one of
the following values:

  MOE  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 (not related to bell prompting).
  UAM  Run in unattended mode, bypassing manual
       intervention (illegal for this diagnostic).
  ISR  Inhibit statistical reports.
IDU INHIBIT DROP-ING OF UNITS BY DIAGNOSTIC.
(WAS NO EFFECT IN THIS 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 "Q
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:MOE=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.
2.4.2 RESTART COMMAND
******************************************************************************
RESTART/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 DIALOGUE. 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 UNITS SWITCH SHOULD
   NOT BE USED WITH THIS PROGRAM. THE SOFTWARE DIALOGUE MAY OPTIONALLY
   BE REEXECUTED (OPERATOR WILL BE ASKED). THE COMMAND CAN BE USED AFTER
   COMMAND MODE HAS BEEN ENTERED 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
******************************************************************************
CONTINUE/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

PROCEED(/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 :

DROP(P)/UNITS:<UNIT-LIST>
2.4.8 EFFECT OF DROP COMMAND
THE UNITS SPECIFIED WILL BE DROPPED FROM TESTING. THE UNITS
WILL BE RESELECTED ONLY BY THE EXECUTION OF AN ADD OR START
COMMAND. THE UNITS SWITCH MUST BE ENTERED. THIS COMMAND
MUST BE FOLLOWED BY A RESTART OR A CONTINUE COMMAND.

2.4.9 PRINT COMMAND
********************************************************************************
PRI(NT)
********************************************************************************

2.4.9.1 EFFECT OF PRINT COMMAND
THE TOTAL NUMBER OF ERRORS FOR EACH UNIT SINCE THE LAST
START OR RESTART COMMAND ARE PRINTED. THE ISR (INHIBIT
STATISTICAL REPORTING) FLAG IS CLEARED.

2.4.10 DISPLAY COMMAND
********************************************************************************
DIS(PLAY)/UNITS:<UNIT-LIST>
********************************************************************************

2.4.10.1 EFFECT OF DISPLAY COMMAND
THE HARDWARE P-Table FOR THE TEST STATION IS PRINTED IN THE
FORMAT IN WHICH IT WAS ENTERED.

2.4.11 FLAGS COMMAND
********************************************************************************
FLAGS
********************************************************************************

2.4.11.1 EFFECT OF FLAGS COMMAND
THE CURRENT SETTINGS OF ALL FLAGS ARE PRINTED.
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 TELETYPET 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


1. CSR ADDRESS - THIS QUESTION REQUESTS THE CSR ADDRESS OF THE SPECIFIED DMU-11. THE DEFAULT ANSWER FOR THIS QUESTION IS ADDRESS 160460 (OCTAL).


3. ACTIVE LINES BIT MAP - THIS QUESTION REQUESTS AN OCTAL BIT MAP OF THE SERIAL COMMUNICATION LINES ON THE DMU-11 WHICH ARE BEING SELECTED FOR TESTING. IF THE BIT IN THE BIT MAP IS SET WHICH CORRESPONDS TO A PARTICULAR LINE (I.E. BIT 5 FOR LINE 5) THAT LINE WILL BE TESTED BY THE FVT.

2.6 SOFTWARE QUESTIONS

AFTER YOU HAVE ANSWERED THE HARDWARE QUESTIONS OR AFTER A RESTART OR CONTINUE COMMAND, THE RUNTIME SERVICES WILL ASK FOR SOFTWARE PARAMETERS. THESE PARAMETERS WILL GOVERN SOME DIAGNOSTIC SPECIFIC OPERATION MODES. YOU WILL BE PROMPTED BY "CHANGE SW (L) ?" IF YOU WISH TO CHANGE ANY PARAMETERS, ANSWER BY TYPING "Y". THE SOFTWARE QUESTIONS AND THE DEFAULT VALUES ARE DESCRIBED IN THE NEXT PARAGRAPH(S).

1. REPORT UNIT NUMBER AS EACH UNIT IS TESTED - THIS QUESTION ASKS WHETHER THE PROGRAM SHOULD REPORT THE NUMBER OF THE UNIT WHICH IT IS TESTING AS IT BEGINS TO TEST THAT UNIT.

2. ROM VERSION PRINTOUT ON THE FIRST PASS - THIS QUESTION ASKS WHETHER THE PROGRAM SHOULD PRINTOUT THE VERSIONS OF THE ON BOARD PROCESSOR ROMS DURING THE FIRST PASS OF THE PROGRAM.

3. EXTENDED ERROR REPORTING - THIS QUESTION ASKS WHETHER EXTENDED ERROR INFORMATION IS REQUIRED OTHER THAN THE "TEST FAILED" MESSAGE, ON EACH ERROR REPORTED. THE DEFAULT IS "NO" I.E. ONLY A MESSAGE REPORTING THE FACT THAT THE TEST FAILED WILL BE PRINTED.

4. NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE - THIS QUESTION IS ASKED ONLY IF THE PREVIOUS QUESTION WAS ANSWERED "YES". THE QUESTION ASKS FOR THE NUMBER OF DATA ERRORS WHICH SHOULD BE REPORTED INDIVIDUALLY BY THIS PROGRAM FOR EACH LINE FOR EACH TRANSMISSION TEST. ERRORS WHICH ARE NOT REPORTED INDIVIDUALLY ARE REPORTED IN SUMMARY ERROR REPORTS.
2.7 EXTEENDED 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 MUXED-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 REPETITIVE.

TO ILLUSTRATE A MORE EFFICIENT METHOD,-suppose you are testing a fictional device, the XYZ1. 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 XYZ1 with eight units.

0 UNITS (0) ? 8<CR>

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

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

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

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

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

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

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

0 UNITS (0) ? 8<CR>

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

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

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

AS YOU CAN SEE IN THE ABOVE DIALOGUE, THE RUNTIME SERVICES WILL BUILD AS MANY ENTRIES AS IT CAN WITH THE INFORMATION GIVEN IN ANY ONE PASS THROUGH THE QUESTIONS. IN THE FIRST PASS, TWO ENTRIES ARE BUILT SINCE TWO SUB-DEVICES AND Q-FACTORS WERE SPECIFIED. THE SERVICES ASSUME THAT THE CSR ADDRESS IS 160000 FOR BOTH SINCE IT WAS SPECIFIED ONLY ONCE. IN THE SECOND PASS, FOUR ENTRIES WERE BUILT. THIS IS BECAUSE FOUR SUB-DEVICES WERE SPECIFIED. THE "." CONSTRUCT TELLS THE RUNTIME SERVICES TO INCREMENT THE DATA FROM THE FIRST NUMBER TO THE SECOND. IN THIS CASE, SUB-DEVICES 2, 3, 4 AND 5 WERE SPECIFIED. (IF THE SUB-DEVICE WERE SPECIFIED BY ADDRESSES, THE INCREMENT WOULD BE BY 2 SINCE ADDRESSES MUST BE 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.

0 UNITS (0) ? 8<CR>
UNIT 1
CSR ADDRESS (O) ? 16000<CR>
SUB-DEVICE # (O) ? 0 7<CR>
Q-FACTOR (O) 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/UNIBUS AND 50HZ (IF THERE IS A CLOCK) QUESTIONS. NOTE, NOT ALL VERSIONS OF XXDP+ ASK FOR THE CLOCK FREQUENCY

3. TYPE "R NAME", WHERE NAME IS THE NAME OF THE BIN OR BIC FILE FOR THIS PROGRAM

4. TYPE "START"

5. ANSWER THE "CHANGE HW" QUESTION WITH "Y"

6. ANSWER ALL THE HARDWARE QUESTIONS

7. ANSWER THE "CHANGE SW" QUESTION WITH "N"

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

3.1 TYPES OF ERROR MESSAGES

THERE ARE THREE LEVELS OF ERROR MESSAGES THAT MAY BE ISSUED BY
A DIAGNOSTIC: GENERAL, BASIC AND EXTENDED. GENERAL ERROR MESSAGES
ARE ALWAYS PRINTED UNLESS THE "IER" FLAG IS SET (SEE THE FLAGS SECTION
OF THIS DOCUMENT).

THE GENERAL ERROR MESSAGE IS OF THE FORM:

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

WHERE: 
- NAME = DIAGNOSTIC NAME
- TYPE = ERROR TYPE (SYS FATAL, DEV FATAL, HARD OR SOFT)
- NUMBER = ERROR NUMBER
- UNIT NUMBER = 0 - N (N IS LAST UNIT IN TABLE)
- 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 SPECIFIC ERROR MESSAGES

This program is intended to provide a go/no go indication of the functionality of the DHU-11 boards. To execute the program in this mode the operator need only answer the "extended error reporting" software question with "no". The program will then only print the name of the failing test, the test and error numbers. 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:

CZDHU DVC FTL ERR 01603 ON UNIT 02 TST 16 SUB 000 PC: Xxxxxx
DEVICE REGISTER WORD READ/WRITE TEST FAILED.

This error indicates that a fatal error was encountered within the test which tests the read/write capability of the DHU 11 registers.

If the operator had requested extended error reporting the same error would be reported as follows:

CZDHU DVC FTL ERR 01603 ON UNIT 02 TST 16 SUB 000 PC: Xxxxxx
DEVICE REGISTER WORD READ/WRITE TEST FAILED,
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 further information see the switches section of this document.
5.0 TEST SUMMARIES

THE FOLLOWING ARE INCLUDED WITHIN CZDMUB:

1. DEVICE REGISTER ACCESS TEST - VERIFIES THAT THE UUT REGISTERS WILL RESPOND WITH THE CORRECT UNIBUS HANDSHAKING SIGNALS, VERIFIES THAT THE UUT IS AT THE CORRECT 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 CLEARS WITHIN A SHORT TIME AFTER IT IS SET WHEN THE SKIP SELFTEST SEQUENCE IS USED.

4. RX.CHARACTER FIELD TEST - VERIFIES THAT THE DATA BITS OF THE CODES IN THE RXFIFO AFTER A MASTER RESET AND SKIP SELFTEST ARE CONSISTENT WITH THE SKIP SELFTEST CODES.

5. RX.FLAG FIELD TEST - VERIFIES THAT THE 3 DATA STATUS BITS (OVERRUN, FRAMING AND PARITY ERROR BITS) ARE ALL SET ON EACH OF THE SKIP SELFTEST CODES IN THE FIFO AFTER A MASTER 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 HAVE BEEN READ.

7. RX.DATA.VAL.ID TEST - VERIFIES THAT THE RX.DATA.VALID BIT IS SET FOR ALL THE CODES IN THE FIFO AND CLEAR AFTER ALL THE CODES HAVE BEEN READ.

8. RX.LINE FIELD TEST - VERIFIES THAT THE RX.LINE LINE FIELDS ARE CORRECT FOR THE SKIP SELFTEST CODES.

9. BPM CHECK TEST - VERIFIES THAT THE DUT DOES NOT IMMEDIATELY FAIL THE BACKGROUND MONITOR PROGRAM, AS THIS MAY INVALIDATE FURTHER TESTS.

10. SKIP SELFTEST TEST - VERIFIES THAT THE DUT SKIPS THE SELFTEST IN THE TIME ALLOWED, AND THAT THE FIFO CONTAINS THE CORRECT CODES AFTER ITS COMPLETION.

11. DIAGNOSTIC.FAIL (SKIP SELFTEST) TEST - VERIFIES USING THE SKIP SELFTEST SEQUENCE THAT THE DIAG.FAIL BIT GOES TO BOTH THE ACTIVE AND INACTIVE STATES WITHIN THE ALLOWED TIMES.

12. SELFTEST TEST - VERIFIES THAT THE DUT'S SELFTEST EXECUTES WITHIN THE CORRECT TIME AND THAT THE CORRECT CODES ARE RETURNED IN THE FIFO AFTER ITS COMPLETION.

13. SELFTEST FAIL TEST - VERIFIES THAT THE DUT WILL REPORT ERRORS CORRECTLY WHEN IT IS FORCED TO FAIL.
14. ROM VERSION NUMBER - VERIFIES THAT THE ROM VERSION NUMBERS
   ARE REPORTED CORRECTLY AND IF REQUESTED PRINTS THEM OUT.

15. CSR BIT 4 TEST - VERIFIES THAT WHEN SET THIS BIT CAUSES
   THE SELFTEST TO LOOP, AND WHEN CLEARED THE SKIP SELFTEST
   CODES ARE RETURNED IN THE RXFIFO.

16. WORD ACCESS READ/WRITE TEST - VERIFIES THAT THE REGISTERS
   RESPOND CORRECTLY TO READ AND WRITE ACCESSES.

17. WORD ACCESS READ/MODIFY/WRITE TEST - VERIFIES THAT THE
   REGISTERS WILL RESPOND CORRECTLY TO READ/MODIFY/WRITE ACCESES.

18. BYTE ACCESS READ/WRITE TEST - VERIFIES THAT THE REGISTERS WILL
   RESPOND CORRECTLY TO BYTE READ/WRITE ACCESSES.

19. BYTE ACCESS READ/MODIFY/WRITE - VERIFIES THAT THE REGISTERS
   WILL RESPOND CORRECTLY TO BYTE READ/MODIFY/WRITE ACCESSES.

20. ID.BIT TEST - VERIFIES THAT THE ID BIT READS AS SET.

21. TX.ENABLE (INACTIVE) TEST - VERIFIES THAT WHEN A LINE'S TX.ENBL
   BIT IS CLEAR, TRANSMISSION WILL NOT TAKE PLACE ON THAT LINE.

22. TX.ENABLE (ACTIVE) TEST - VERIFIES THAT WHEN A LINE'S TX.ENBL
   BIT IS SET, TRANSMISSION WILL TAKE PLACE ON THAT LINE.

23. INTERRUPT TEST - VERIFIES THAT THE DUT WILL GENERATE RECEPTION
   AND TRANSMISSION INTERRUPTS CORRECTLY.

24. BR LEVEL TEST - VERIFIES THAT THE DUT INTERRUPTS AT THE CORRECT
   BUS REQUEST LEVEL.

25. DIAGNOSTIC FIELD (BMP) TEST - VERIFIES THAT A REQUEST TO THE
   DUT TO REPORT BMP STATUS CODES IS COMPLIED WITH WITHIN THE
   SPECIFIED TIME. ALL ACTIVE LINES ARE TESTED.

26. REPORT BMP CODES TEST - THIS PSEUDO TEST REPORTS THE FIRST 32
   BMP CHARACTERS WHICH WERE DISCOVERED IN THE FIFO DURING THE
   EXECUTION OF THE OTHER TESTS. THIS AVOIDS INTERRUPTION OF THE
   OTHER TESTS BY THESE CODES IF THEY ARE NOT CRITICAL TO THE
   PERFORMANCE OF THE TESTS.
6.0 EXAMPLE ERROR FREE PASS

THE FOLLOWING IS AN EXAMPLE OF AN ERROR FREE PASS DIALOGUE:

.R CZDHB0
CZDHB0.BIN

QRS
CZDHB-B-0
DMU-11 FUNC IST PART1
UNIT IS DMU-11
RESTRT ADDR: 147670
DR>STA/PAS:1

CHANGE HW (L) ? Y

UNIT (D) ? Z

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

UNIT 1
CSR ADDRESS: (O) 160460 ? 160500
INTERRUPT VECTOR ADDRESS: (O) 310 ? 320
ACTIVE LINE BIT MAP: (O) 177777 ? <CR>
INTERRUPT BR LEVEL: (O) 5 ? <CR>

CHANGE SW (L) ? Y

REPORT UNIT NUMBER AS EACH UNIT IS TESTED: (L) Y ? <CR>
ROM VERSION PRINTOUT ON THE FIRST PASS: (L) Y ? <CR>
EXTENDED ERROR REPORTING: (L) N ? Y
NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE: (D) 0 ? 1

TESTING UNIT : 0(D)

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

TESTING UNIT : 1(D)

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

CZDHB EOP
1
0 TOTAL ERRS

DR>
&

.LIST SEQ,LOC,BIN,MEB

.MLIST CND

.SBTTL PROGRAM HEADER

.MCALL SVC

SVC

; INITIALIZE SUPERVISOR MACROS

;----------------------------------------------------------------------
; IF STRUCTURED MACROS ARE TO BE USED, ADD "MCALL STRUCT" AND "STRUCT"
; TO INITIALIZE THE STRUCTURED MACROS.

SVCINS* 1 ; LIST INSTRUCTIONS, SHIFTED RIGHT
SVCST* 1 ; LIST TEST TAGS, SHIFTED RIGHT
SVCSUB* 1 ; LIST SUBTEST TAGS, SHIFTED RIGHT
SVCGBL* 1 ; LIST GLOBAL TAGS, SHIFTED RIGHT
SVCTAG* 1 ; LIST OTHER TAGS, SHIFTED RIGHT

; CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
; TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
; SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
; CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.

;----------------------------------------------------------------------

.ENABLEL ABS
.ENABLEL AMA
  * 2000

;----------------------------------------------------------------------

.BGNMOD

;----------------------------------------------------------------------

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

;----------------------------------------------------------------------

POINTER BGNRPT,BGNSW,BGNSFT,BGNNDU,EARTBL

HEADER CZDHU.B,0,16,0,PRI0?

.L*NAME::

.L*ASCII /C
.L*ASCII /Z/
.L*ASCII /D/
.L*ASCII /H/
.L*ASCII /U/
.L*BYTE 0
.L*BYTE 0
.L*BYTE 0

.L*REV::
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>002100</td>
<td>104025</td>
</tr>
<tr>
<td>002102</td>
<td>003766</td>
</tr>
<tr>
<td>002104</td>
<td>024236</td>
</tr>
<tr>
<td>002106</td>
<td>025060</td>
</tr>
<tr>
<td>002110</td>
<td>025056</td>
</tr>
<tr>
<td>002112</td>
<td>024230</td>
</tr>
<tr>
<td>002114</td>
<td>000000</td>
</tr>
<tr>
<td>002116</td>
<td>000000</td>
</tr>
<tr>
<td>002120</td>
<td>000000</td>
</tr>
<tr>
<td>002120</td>
<td>000000</td>
</tr>
</tbody>
</table>

L$LOAD::  EMT  E$LOAD
L$RTP::  .WORD  L$ERRTBL
L$ICP::  .WORD  L$INIT
L$ICCP:: .WORD  L$CLEAN
L$ACP::  .WORD  L$AUTO
L$RTP::  .WORD  L$PROT
L$TEST:: .WORD  0
L$PLY::  .WORD  0
L$TIME:: .WORD  0
**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 26**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>002132</td>
<td>000032</td>
</tr>
<tr>
<td>002142</td>
<td>025012</td>
</tr>
<tr>
<td>002152</td>
<td>025476</td>
</tr>
<tr>
<td>002162</td>
<td>025726</td>
</tr>
<tr>
<td>002172</td>
<td>026172</td>
</tr>
<tr>
<td>002182</td>
<td>026370</td>
</tr>
<tr>
<td>002192</td>
<td>026562</td>
</tr>
<tr>
<td>002202</td>
<td>027000</td>
</tr>
<tr>
<td>002212</td>
<td>027206</td>
</tr>
<tr>
<td>002222</td>
<td>027414</td>
</tr>
<tr>
<td>002232</td>
<td>027610</td>
</tr>
<tr>
<td>002242</td>
<td>030024</td>
</tr>
<tr>
<td>002252</td>
<td>030252</td>
</tr>
<tr>
<td>002262</td>
<td>030424</td>
</tr>
<tr>
<td>002272</td>
<td>030496</td>
</tr>
<tr>
<td>002282</td>
<td>031034</td>
</tr>
<tr>
<td>002292</td>
<td>031434</td>
</tr>
<tr>
<td>002302</td>
<td>031776</td>
</tr>
<tr>
<td>002312</td>
<td>032244</td>
</tr>
<tr>
<td>002322</td>
<td>032434</td>
</tr>
<tr>
<td>002332</td>
<td>032752</td>
</tr>
<tr>
<td>002342</td>
<td>033214</td>
</tr>
<tr>
<td>002352</td>
<td>033330</td>
</tr>
<tr>
<td>002362</td>
<td>033646</td>
</tr>
<tr>
<td>002372</td>
<td>034222</td>
</tr>
<tr>
<td>002382</td>
<td>035360</td>
</tr>
<tr>
<td>002392</td>
<td>036322</td>
</tr>
<tr>
<td>002402</td>
<td>036616</td>
</tr>
</tbody>
</table>

**LIDISPATCH:**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>000012</td>
<td>T1</td>
</tr>
<tr>
<td>000022</td>
<td>T2</td>
</tr>
<tr>
<td>000032</td>
<td>T3</td>
</tr>
<tr>
<td>000042</td>
<td>T4</td>
</tr>
<tr>
<td>000052</td>
<td>T5</td>
</tr>
<tr>
<td>000062</td>
<td>T6</td>
</tr>
<tr>
<td>000072</td>
<td>T7</td>
</tr>
<tr>
<td>000082</td>
<td>T8</td>
</tr>
<tr>
<td>000092</td>
<td>T9</td>
</tr>
<tr>
<td>000102</td>
<td>T10</td>
</tr>
<tr>
<td>000112</td>
<td>T11</td>
</tr>
<tr>
<td>000122</td>
<td>T12</td>
</tr>
<tr>
<td>000132</td>
<td>T13</td>
</tr>
<tr>
<td>000142</td>
<td>T14</td>
</tr>
<tr>
<td>000152</td>
<td>T15</td>
</tr>
<tr>
<td>000162</td>
<td>T16</td>
</tr>
<tr>
<td>000172</td>
<td>T17</td>
</tr>
<tr>
<td>000182</td>
<td>T18</td>
</tr>
<tr>
<td>000192</td>
<td>T19</td>
</tr>
<tr>
<td>000202</td>
<td>T20</td>
</tr>
<tr>
<td>000212</td>
<td>T21</td>
</tr>
<tr>
<td>000222</td>
<td>T22</td>
</tr>
<tr>
<td>000232</td>
<td>T23</td>
</tr>
<tr>
<td>000242</td>
<td>T24</td>
</tr>
<tr>
<td>000252</td>
<td>T25</td>
</tr>
<tr>
<td>000262</td>
<td>T26</td>
</tr>
</tbody>
</table>
SBITLE 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.

BGNHW DFPTBL

.lword L10000 L$HW/2

DFPTBL:

.word 160460 ;DEFAULT CSR ADDRESS

.word 310 ;DEFAULT VECTOR ADDRESS

.word 177777 ;DEFAULT ACTIVE LINES BIT MAP

.byte 5 ;DEFAULT BR LEVEL

.even

ENDNW

L10000:
.SBTTL SOFTWARE P TABLE

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

BGNSW SFPTBL

.LISW: .WORD L10001 L$SW/2
SFPTBL::

.LISW: .WORD 21 ;BIT MAP OF PROGRAM CONTROL FLAGS
NDERPT: .WORD 0 ;DEFAULT NUMBER OF INDIVIDUAL DATA ERRORS TO RP'T.

ENDSW

L10001:
.SBTL1 GLOBAL EQUATES SECTION

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

000020  NUMLN=20 ;NUMBER OF LINES ON DHU11 IS 16.
177777  MAPLN=177777 ;BIT MAP OF LINES ON DHU11.

****** DEVICE REGISTER OFFSETS FROM THE CSR'S ADDRESS ******
LRPO=4 ;LINE PARAMETER REGISTER OFFSET FROM THE CSR ADDRESS
FSLSO=6 ;FIPOSIZE/STATUS REGISTER OFFSET FROM THE CSR ADDRESS
TXBFCO=16 ;TRANSMIT COUNT REGISTER OFFSET FROM THE CSR ADDRESS

EQUALS

; BIT DEFINITIONS

100000 BIT15= 100000
040000 BIT14= 40000
020000 BIT13= 20000
010000 BIT12= 10000
004000 BIT11= 4000
002000 BIT10= 2000
001000 BIT09= 1000
000400 BIT08= 400
000200 BIT07= 200
000100 BIT06= 100
000040 BIT05= 40
000020 BIT04= 20
000010 BIT03= 10
000004 BIT02= 4
000002 BIT01= 2
000001 BIT00= 1

001000 BIT9= BIT09
000400 BIT8= BIT08
000200 BIT7= BIT07
000100 BIT6= BIT06
000040 BIT5= BIT05
000020 BIT4= BIT04
000010 BIT3= BIT03
000004 BIT2= BIT02
000002 BIT1= BIT01
000001 BIT0= BIT00

; EVENT FLAG DEFINITIONS
; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

000040 EF.START== 32. ;START COMMAND WAS ISSUED
000037 EF.RESTART== 31. ;RESTART COMMAND WAS ISSUED
000036 EF.CONTINUE== 30. ;CONTINUE COMMAND WAS ISSUED
GLOBAL EQUATES SECTION

000035 EF:new** 29.
000034 EF:pwr** 28.

; A NEW PASS HAS BEEN STARTED
; A POWER FAIL/POWER UP OCCURRED

; PRIORITY LEVEL DEFINITIONS

000040 PRI07** 340
000040 PRI06** 300
000040 PRI05** 240
000040 PRI04** 200
000040 PRI03** 140
000040 PRI02** 100
000040 PRI01** 40
000040 PRI00** 0

; OPERATOR FLAG BITS

000004 EVL** 4
000010 LDT** 10
000020 ADR** 20
000040 IDU** 40
000100 ISA** 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 MOE** 100000
.SBTTL  GLOBAL DATA SECTION

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

;******************************************************************************
; UNIT VARIABLE AREA
;******************************************************************************

ACTLNG:: .WORD 177777  ;ACTIVE LINE BIT MAP.
RXVECA:: .WORD 300   ;RX VECTOR ADDRESS.
TXVECA:: .WORD 304   ;TX VECTOR ADDRESS.
UNITN::  .WORD 0   ;UNIT NUMBER.
BRLEVL::  .BYTE 4   ; INTERRUPT BUS REQUEST LEVEL.

;******************************************************************************
; DEVICE REGISTER ADDRESS TABLE
;******************************************************************************

DRADAT::
    CSRA::  .WORD 160020  ;DMU11 CSR ADDRESS.
    RXMRA:: .WORD 160022  ;DMU11 RECEIVE BUFFER/TIMER ADDRESS.
    LPRA::  .WORD 160024  ;DMU11 LINE PARAMETER REGISTER ADDRESS.
    FDATA:: .WORD 160026  ;DMU11 FIFO SIZE/LINE STATUS REGISTER ADDRESS.
    FSDAT:: .WORD 160028  ;DMU11 FIFO DATA REGISTER ADDRESS.
    LNTCTR:: .WORD 160030  ;DMU11 LINE CONTROL REGISTER ADDRESS.
    TXMRA:: .WORD 160032  ;DMU11 TRANSMIT BUFFER 1 REGISTER ADDRESS.
    TXMRA:: .WORD 160034  ;DMU11 TRANSMIT BUFFER 2 REGISTER ADDRESS.
    TXBFC:: .WORD 160036  ;DMU11 TRANSMIT BUFFER COUNT REGISTER ADDRESS.

;******************************************************************************
; REGISTER MESSAGE ADDRESS TABLE
;******************************************************************************

RMATBB::
    DROMG:: .WORD 0  ;ADDRESS OF "CSR" MESSAGE.
    DROMG:: .WORD 0  ;ADDRESS OF "RBUF" MESSAGE.
    DROMG:: .WORD 0  ;ADDRESS OF "LPR" MESSAGE.
    DROMG:: .WORD 0  ;ADDRESS OF "STAT" MESSAGE.
    DROMG:: .WORD 0  ;ADDRESS OF "LNCTR" MESSAGE.
    DROMG:: .WORD 0  ;ADDRESS OF "BUFADD1" MESSAGE.
    DROMG:: .WORD 0  ;ADDRESS OF "BUFADD2" MESSAGE.
    DROMG:: .WORD 0  ;ADDRESS OF "BUFFCT" MESSAGE.

;******************************************************************************
; ASSORTED GLOBAL VARIABLES:
;******************************************************************************

BUFPTR:: .WORD 0  ;STORAGE FOR RECEIVE CHARACTER BUFFER POINTER.
EXERR:: .WORD 0  ;"EXIT ON ERROR" FLAG.
CTRLCF:: .WORD 0  ;STORAGE FOR THE CONTROL-C FLAG.
TESTAT:: .WORD 0  ;STORAGE FOR THE INTERRUPT ENABLE BIT STATES.
PSGC:: .WORD 0  ;STORAGE FOR PASS COUNT USED IN ROM VERSION TEST.
RXINTC:: .WORD 0  ;STORAGE FOR RECEIVER INTERRUPT FLAGS.
GLOBAL DATA SECTION

1534 002316 000000
1535 002320 000000
1536 002322 000000
1537 002324 000001
1538 002326 000000
1539 002330 000000
1540 002332 000000
1541
1542
1543
1544
1545 002344 177546
1546 002346 000000
1547 002348 000100
1548 00234A 000074
1549 00234C 000000
1550 00234E 000000
1551 002350 000170
1552 002352 000170
1553 002354 000021
1554 002356 000062
1555
1556
1557
1558
1559 002360 177572
1560 002362 000000
1561 002364 000000
1562 002366 172340
1563
1564
1565
1566 002370 137660
1567 002372 177777
1568 002374 000007
1569 002376 177777
1570 002380 166051
1571 002382 000000
1572 002384 000000
1573 002386 177774
1574 002388 000000
1575
1576
1577
1578
1579 002410 000001
1580 002412 000002
1581 002414 000004
1582 002416 000010
1583 002420 000020
1584 002422 000040
1585 002424 000100
1586 002426 000200
1587 002430 000400
1588 002432 001000
1589 002434 002000
1590 002436 004000

RXINF:.WORD 0 ;STORAGE FOR RECEIVER INTERRUPT FLAGS.
T4FALG:.WORD 0 ;FLAGS SET WHEN AN EXPECTED 004 TRAP OCCURS.
T4VEC:.WORD 0 ;STORAGE FOR THE NORMAL 004 TRAP VECTOR.
T4VEC:.WORD 0 ;STORAGE FOR THE 004 TRAP VECTOR.
TIXINIT:.WORD 0 ;STORAGE FOR TRANSFORM INTERRUPT COUNT.
TXINF:.WORD 0 ;STORAGE FOR TRANSMIT INTERRUPT FLAGS.
WORD1:.WORD 0 ;LOCATION FOR PASSING INDIRECT PARAMETERS.

***************
LINE TIME CLOCK VARIABLES AND STORAGE.
***************

CSR ADDRESS OF THE LTC
CLKPR:.WORD 177546 ;INTERUPT PRIORITY LEVEL OF THE LTC.
CLKVEC:.WORD 0 ;INTERRUPT VECTOR ADDRESS OF THE LTC.
CLKFRC:.WORD 0 ;INTERRUPT FREQUENCY OF THE LTC.
HARDWARE CLOCK COUNTER #1.
HARDWARE CLOCK COUNTER #2.
HARDWARE BREAK COUNTER LOCATION.
BREAK COUNT VALUE IN CLOCK TICKS.
NUMBER OF MILLI-SECONDS PER L1C TICK.
LOOP COUNT (USED BY MSLOOP) TO DELAY 1 MS.

***************
MEMORY MANAGEMENT VARIABLES AND FLAGS.
***************

MMR0:.WORD 177572 ;ADDRESS OF MEM MGT STATUS REGISTER #0.
MMG1:.WORD 0 ;MEM MGT PRESENT FLAG (0 IF MM NOT PRESENT).
MMG2:.WORD 0 ;MEM MGT ENABLED FLAG (0 IF MM NOT ENABLED).
MMG3:.WORD 0 ;ADDRESS OF MEM MGT PAR #0.

***************
BIT MASK TABLE OF UN-USED DMU DEVICE REGISTER BITS.
***************

UNBITB:.WORD 137660 ;UNUSED BIT MASK FOR THE CSR.

***************
TABLE OF WORDS WITH CORRESPONDING BIT SET FOR GENERATION OF BIT MAPS.
***************

BITBL:.WORD 1 ;BIT 0 SET.
.BIT 2 SET.
.BIT 3 SET.
.BIT 4 SET.
.BIT 5 SET.
.BIT 6 SET.
.BIT 7 SET.
.BIT 8 SET.
.BIT 9 SET.
.BIT 10 SET.
.BIT 11 SET.
GLOBAL DATA SECTION

1591  002440  010000
1592  002442  020000
1593  002444  030000
1594  002446  100000
1595
1596
1597
1598
1599  002450
1400  002450  000000
1401  002452  000000
1402  002454  000000
1403  002456  000000
1404  002460  000000
1405
1406
1407
1408
1409  002462  000000
1410  002464
1411
1412
1413
1414
1415  002524  000000
1416  002526
1417  002726
1418
1419
1420
1421
1422  002726
1423  002726
1424  003326
1425  003526
1426  003726
1427  003726
1428
1429
1430
1431
1432
1433
1434  003766
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445

.LWORD 100000  BIT 12 SET.
.LWORD 200000  BIT 13 SET.
.LWORD 400000  BIT 14 SET.
.LWORD 100000  BIT 15 SET.

;************************************************************************
;* GPR SAVE AREA ZERO.
;************************************************************************

GPRSOB::  .LWORD 0  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.
;************************************************************************

ERMSFL::  .LWORD 0  ERROR SUMMARY REPORT FLAGS.
ERCNTB::  .BLKW 16.  TABLE OF ERROR COUNTERS.

;************************************************************************
;* STORAGE AREA FOR THE BMP CODE QUEUE.
;************************************************************************

BMPQCP::  .LWORD 0  ;POINTER USED TO ACCESS THE NEXT CELL IN QUE.
BMPQCO::  .BLKW 64.  ;STORAGE FOR 32 CELLS, TEST0 PLUS BMP CODE.
           .BLKW 16.  ;LAST ADDRESS PLUS 2 OF THE BMP CODE QUEUE.

;************************************************************************
;* GENERAL TABLE AND BUFFER AREA--513 WORDS.
;************************************************************************

BUFFAS::  .LWORD 0  ;BASE OF MEMORY BUFFER.
EALTL::  .BLKW 128.  ;FIRST HALF OF GENERAL TABLE OR BUFFER.
BUFMD::  .BLKW 64.  ;SECOND HALF OF GENERAL TABLE OR BUFFER.
BUF3QT::  .BLKW 64.  ;LAST QUARTER OF THE BUFFER AREA.
BUFEND::  .BLKW 64.  ;END OF GENERAL PURPOSE MEMORY BUFFER.
ENDEBF::  .BLKW 16.  ;BUFFER OVERFLOW SPACE.

ERRBL::  .LWORD 0
ERRTP::  .LWORD 0
ERRNR::  .LWORD 0
ERRMSG:: .LWORD 0
ERRBLK:: .LWORD 0

.EVEN
GPR HANDLING ROUTINES FOR SUBROUTINE CALLS.

There are 4 routines and macro definitions used for the handling of GPR values during subroutine calls within this program. The four routines/macro calls have the following names:

- **SAVE** - macro definition used at the beginning of a subroutine to save the GPR contents for later restoration.
- **PASS** - macro definition used at the end of a subroutine to restore the previously saved GPR contents and to leave the contents of the specified GPR(s) intact (not restored).
- **PREG05** - subroutine which is called from the save and pass macro expansions which actually performs the actions on the GPRs.

During a subroutine which uses these GPR save routines the values of the GPRs are stored on the stack in the following stack frame:

```
SP  -> RET PC INTO PREG05 ROUTINE.
SP+2 -> GPR RO CONTENTS.
SP+4 -> GPR R1 CONTENTS.
SP+6 -> GPR R2 CONTENTS.
SP+8 -> GPR R3 CONTENTS.
SP+10-> GPR R4 CONTENTS.
SP+12-> GPR R5 CONTENTS.
SP+14-> RET PC INTO CALLER OF SUB'TNE WHICH CALLED PREG05.
```

Each level of sub'tne calling uses 8 words of stack overhead. The save and pass macros can also be used in "straight line code" to save and restore the GPR values. In any case, after the issuing of a pass call, the GPRs will be restored to the values they had prior to the last save call (except for the exceeded, or passed intact, GPRs specified as parameters to the pass call). And the SP will also be restored to its condition before the last save call. The programmer must be sure that the SP has the same value when the pass macro is called as it had immediately after the save macro was called.
.SBTL GPR FRAME ACCESS EQUATES

<table>
<thead>
<tr>
<th>Offset</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000036</td>
<td>LPCSLT</td>
<td>Offset for last return PC.</td>
</tr>
<tr>
<td>000016</td>
<td>PC5LOT</td>
<td>Offset for return PC.</td>
</tr>
<tr>
<td>000014</td>
<td>R5SLOT</td>
<td>Offset for R5.</td>
</tr>
<tr>
<td>000012</td>
<td>R4SLOT</td>
<td>Offset for R4.</td>
</tr>
<tr>
<td>000010</td>
<td>R3SLOT</td>
<td>Offset for R3.</td>
</tr>
<tr>
<td>000006</td>
<td>R2SLOT</td>
<td>Offset for R2.</td>
</tr>
<tr>
<td>000004</td>
<td>R1SLOT</td>
<td>Offset for R1.</td>
</tr>
<tr>
<td>000002</td>
<td>R0SLOT</td>
<td>Offset for R0.</td>
</tr>
</tbody>
</table>
**GLOBAL MACRO DEFINITION**

**SAVE**

**BEGINNING OF A SUBROUTINE TO SAVE THE CONTENTS OF THE GPRs R0 THRU R5.**

**Inputs:** SP UNCHANGED SINCE SUBROUTINE WAS ENTERED

**R5SLOT - OFFSET TO STACK SLOT FOR R5 (EQUATED TO 14 OCTAL)**

**Outputs:** GPR SAVE AREA ON THE STACK IS LOADED WITH THE CONTENTS OF GPRs

**TOP OF STACK LOADED WITH THE RETURN ADDRESS INTO PREG05**

**Calling Sequence:** SAVE

**Comments:** NO ARGUMENTS ARE ALLOWED.

**THE PASS MACRO SHOULD BE CALLED TO RESTORE THE GPR VALUES.**

**Subordinate Routines Called:** PREG05.

**END**

**SAVE**

**MACRO** SAVE

**LIST**

**JSR** R5, PREG05 ; CALL REGISTER SAVE SUBRT.

**NLIST**

**ENDM** SAVE
**GLOBAL MACRO Definition**

**.SBTIL GLOBAL MACRO DEFINITION - PASS**

- **This macro is used in conjunction with the SAVE macro. It is called at end of a subroutine to pass parameters in GPRs back to the calling routine by altering the GPR save area on the stack and then returning to pregos to restore the GPRs to their saved values.**

**Inputs:** Only allowed arguments are "R0" thru "R5".

**Outputs:** The GPA values are put in their respective slots on the stack.

**Calling sequence:** PASS R0,R1,...

**Comments:** Any combination of GPR arguments may be listed in any order.

For example, the following are legal:

PASS R1
PASS R4,R0,R2

The GPRs listed as arguments will be passed intact to the calling routine, all other GPRs will be restored.

The SP must be at its original value when PASS is called.

**The Macro call**

PASS R0,R3

Expands into the following assembly code:

MOV R0,RO,ROSLOT(SP) ;PUT RO IN STACK SLOT.
MOV R3,RS5,ROSLOT(SP) ;PUT R3 IN STACK SLOT.
JSR PC,(SP)* ;RETURN TO PREGOS SUBRT.

In this example GPRs R1, R2, R4, and R5 will be restored to their values contained in the stack frame and RO and R3 will be left at their values prior to this pass call.

**Subordinate routines called:** (PREGRT - LABEL WITHIN PREGOS, VALUE ON STACK.)

```
.MACRO PASS A,B,C,D,E,F
.IMP X,{'A','B','C','D','E','F'}
.IF NB,X
.LIST MOV X,X,RO,ROSLOT(SP) ;PUT X IN STACK SLOT.
.NLIST .ENDC
.ENDM
.LIST JSR PC,(SP)* ;RETURN TO PREGOS SUBRT.
.NLIST .ENDM PASS
```
GLOBAL SUBROUTINE

SBTIL GLOBAL SUBROUTINE - PREGOS -

PRESERVE REGISTERS RO THROUGH R5 FOR SUBROUTINE CALLS.

* INPUTS: THE RETURN ADDRESS BACK INTO THE CALLING ROUTINE MUST BE IN
  GPR RS. (I.E. MACROS USE "JSA RS, PREGOS").

* OUTPUTS: REGISTERS RO THROUGH R5 ARE SAVED ON THE STACK.

* CALLING SEQUENCE:
  [SUBROUTINE CODE]
  PASS

* MACRO EXPANSION CALLS PREGOS.

* MACRO EXPANSION RECALLS PREGOS.

* COMMENTS:
  THIS ROUTINE IS RE-ENTRANT.

* PARAMETERS MAY BE PASSED OUT OF A SUBROUTINE BY MODIFYING THE
  REGISTER SAVE AREA ON THE STACK. USE THE PASS GPRN MACRO
  TO RETURN GPR VALUES INTACT.

* USE THE RNSLOT OFFSETS FROM THE SP TO PASS OTHER PARAMETERS.

* EXAMPLE: MOV VALUE, RNSLOT(SP)

* MAKE SURE THE SP IS AT ITS ORIGINAL VALUE WHEN YOU DO THIS.

* SUBORDINATE ROUTINES CALLED: NONE.

******************************************************************************

PREGOS:

R5 HAS BEEN LOADED ON THE STACK BY THE SUBROUTINE CALL

MOV R4, -(SP)
  ; SAVE R4

MOV R3, -(SP)
  ; SAVE R3

MOV R2, -(SP)
  ; SAVE R2

MOV R1, -(SP)
  ; SAVE R1

MOV R0, -(SP)
  ; SAVE R0

MOV R5, -(SP)
  ; PUSH RETURN PC ON TOP OF STACK

MOV R5SLOT(SP), R5
  ; RESTORE R5 TO VALUE IT HAD BEFORE CALLS

JSR PC, @(SP)
  ; CALL THE SUBROUTINE AT THE RETURN ADDRESS

; FROM THE PREGOS CALL, PUTTING THE PRESENT
; PC ON THE STACK AS A RETURN ADDRESS INTO

; THIS (PREGOS) ROUTINE.

****

; THE FOLLOWING CODE IS EXECUTED WHEN THE CALLING ROUTINE DOES A
; "RETURN" [JSR PC, @(SP)] USING THE PC DEPOSITED ON THE STACK ABOVE.

; ---

PREGRT:

MOV (SP), R5
  ; PUT RETURN PC IN R5.

MOV (SP), R0
  ; RESTORE R0.

MOV (SP), R1
  ; RESTORE R1.

MOV (SP), R2
  ; RESTORE R2.

MOV (SP), R3
  ; RESTORE R3.

MOV (SP), R4
  ; RESTORE R4.

RTS R5
  ; RETURN TO THE SUBROUTINE WHICH CALLED PREGOS.

; RESTORING R5 IN THE PROCESS.
THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS, MESSAGES, AND ASCII INFORMATION THAT ARE USED IN MORE THAN ONE TEST.

NAMES OF DEVICES SUPPORTED BY PROGRAM

DEVTYP <DMU 11> L#DVTYP:: .ASCIZ /DMU 11/

.EVEN

TEST DESCRIPTION

DESCR PT <DMU 11 FUNC TST PART1> L#DESC:: .ASCIZ /DMU-11 FUNC TST PART1

.EVEN

.EVEN
; ********** FORMAT1 STATEMENTS USED IN PRINT CALLS **********

004074 EF0503: .ASCIZ /#T5N/
004101 EF0505: .ASCIZ /#A /#05#/A ILLEGAL INTERRUPTS RECEIVED./#N/
004154 EF1401: .ASCIZ /#A #2#/A ROM VERSION NUMBERS: PROC 1 = #2#/A(0) PROC 2 = #2#/A(0)#/N/
004256 EF1402: .ASCIZ /#A #2#/A ROM VERSION NUMBER #/N/
004313 EF1601: .ASCIZ /#A #2#/A. TEST ABORTED ./N/
004345 EF1602: .ASCIZ /#A EXPECTED DATA: #O6#/A (0)./N/
004407 EF1603: .ASCIZ /#A ACTUAL DATA: #O6#/A (0)./N/
004451 EF1604: .ASCIZ /#A BAD BIT(S) IN DEVICE #2#/A REGISTER FOR LINE #2#/A (0)./N/
004546 EF3001: .ASCIZ /#A EXPECTED OR CORRECT VALUE: #03#/N/
004615 EF3002: .ASCIZ /#A ACTUAL OR MEASURED VALUE: #03#/N/
004664 EF3006: .ASCIZ /#A #2#/A #2#/A(0)#/N/
004710 EF9010: .ASCIZ /#A NUMBER OF ERRORS DETECTED ON LINE #2#/A IS #05#/N/
004777 EF9013: .ASCIZ /#A UNEXPECTED #2#/A FOR LINE #2#/A(0) IN FIFO AFTER RESET./N/
005074 EF9017: .ASCIZ /#A #2#/A (WITH ERROR FLAGS) IS #O6#/A(0)#/N/
005150 EF9018: .ASCIZ /#A #2#/A IN SELFTEST CODE FIFO SLOT FOR LINE #2#/A(0) AFTER RESET./N/
005230 EF9301: .ASCIZ /#A #2#/A(0). BMP CODE REPORTED = #03#/A(0)#/N/
005333 EF9302: .ASCIZ /#A OVERFLOW OCCURRED (MORE THAN 31 BMP CODES FOUND IN QUEUE)./N/
005433 MFUNIT: .ASCIZ /#A #2#/A TESTING UNIT = #04#/N/
1700 .EVEN
1701 .LIST BIN
**Global Error Messages**

00446 DRO0MG: ASCIZ /CSR/
005470 DRO0MG: ASCIZ /RBUF/
005475 DRO0MG: ASCIZ /LPR/
005501 DRO0MG: ASCIZ /SIZE,STAT/
005517 DR0MG: ASCIZ /LCTRL/
005526 DR12MG: ASCIZ /TBUFFAD1/
005537 DR14MG: ASCIZ /TBUFFAD2/
005550 DR16MG: ASCIZ /TBUFCT/
005560 EM0103: ASCIZ /DEVICE REGISTER ACCESS ERRORS/
005616 EM0201: ASCIZ /MASTER RESET TEST FAILED/
005651 EM0202: ASCIZ /MASTER RESET BIT DID NOT CLEAR AFTER BOARD RESET. /
005739 EM0203: ASCIZ /WAITED 5 SECONDS, BIT DEFECTIVE OR FIRMWARE MUNG/
006024 EM0203: ASCIZ /MASTER RESET BIT CLEAR IMMEDIATELY AFTER BOARD RESET. /
006114 ASCIZ /BIT DEFECTIVE OR BOARD FIRMWARE ERROR. /
006167 EM0204: ASCIZ /MR BIT WENT CLEAR WITHIN 1/2 SECOND OF BOARD RESET. /
006265 EM0205: ASCIZ /BIT DEFECTIVE OR SELFTEST WAS (INCORRECTLY) SKIPPED. /
006346 EM0301: ASCIZ /MASTER RESET (SKIP SELFTEST) TEST FAILED. /
006420 EM0302: ASCIZ /MR BIT CLR WITHIN 10 MILISECOND AFTER BOARD RESET. /
006505 ASCIZ /BIT DEFECTIVE OR BOARD FIRMWARE ERROR. /
006560 EM0303: ASCIZ /MR BIT WENT CLEAR 1/3 TO 5 SECONDS AFTER RESET. /
006642 ASCIZ /SELFTEST DID NOT GET SKIPPED (SHOULD HAVE BEEN SKIPPED). /
006737 EM0401: ASCIZ /RBUF REGISTER RX CHARACTER FIELD TEST FAILED. /
007015 EM0402: ASCIZ /IMPROPER CODE FOUND IN RX FIFO AFTER DOT RESET. /
007077 EM0501: ASCIZ /EXPECTED: SELFTEST CODE, ACTUAL: IMPROPER CODE. /
007165 EM0505: ASCIZ /RBUF REGISTER ERROR FLAGS FIELD TEST FAILED. /
007241 EM0502: ASCIZ /RX ERROR FLAG(S) FOUND CLEAR ON SELFTEST CODE. /
007322 EM0502: ASCIZ /EXPECTED: ALL ERROR FLAGS SET, ACTUAL: FLAG(S) CLEAR. /
007415 EM0525: ASCIZ /RX INTERRUPT(S) RECEIVED WITH RX INTERRUPTS DISABLED. /
007505 EM0526: ASCIZ /TX INTERRUPT(S) RECEIVED WITH TX INTERRUPTS DISABLED. /
007575 EM0601: ASCIZ /CSR RX,DATA,AVAIL BIT TEST FAILED. /
007637 EM0602: ASCIZ /RX ,DATA,AVAIL BIT FOUND CLEAR AFTER RESET COMPLETION. /
007727 EM0603: ASCIZ /EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO. /
008007 EM0609: ASCIZ /RX ,DATA,AVAIL BIT COULD NOT BE CLEARED BY PURGING FIFO. /
008111 EM0701: ASCIZ /RBUF RX,DATA,VALID BIT TEST FAILED. /
008202 EM0702: ASCIZ /RX ,DATA,VALID BIT FOUND CLEAR AFTER RESET COMPLETION. /
008245 EM0703: ASCIZ /EXPECTED BIT TO BE SET FROM SELFTEST CODES IN FIFO. /
008335 EM0704: ASCIZ /RX ,DATA,VALID BIT COULD NOT BE CLEARED BY PURGING FIFO. /
008517 EM0705: ASCIZ /600 CHAR READ FROM FIFO WITHOUT R.D.A BIT CLEARED. /
008610 EM0801: ASCIZ /RBUF RX,LINE,NUMBER FIELD TEST FAILED. /
008656 EM0802: ASCIZ /LINE NUMBER WRONG ON A SELFTEST CODE. /
008726 EM0901: ASCIZ /CHECK FOR BMP CODES TEST FAILED. /
008766 EM0902: ASCIZ /UNEXPECTED BMP CODES FOUND. /
008795 EM1001: ASCIZ /SKIT SELF-TEST TEST FAILED. /
008802 EM1002: ASCIZ /SKIT SELF-TEST TOOK TOO LONG TO COMPLETE, > 50 MS. /
008811 EM1102: ASCIZ /SKIP SELF-TEST COMPLETED TOO SOON, < 10 MS. /
008812 EM1103: ASCIZ /DIAGNOS FAIL (SKP SELFTEST) TEST FAILED. /
008813 EM1201: ASCIZ /SELF-TEST FAILED. /
008814 EM1202: ASCIZ /SELF-TEST TOOK TOO LONG TO COMPLETE, > 5 SECONDS. /
008815 EM1203: ASCIZ /SELF-TEST COMPLETED TOO SOON, < 1/2 SECOND. /
008816 EM1204: ASCIZ /SELF-TEST DID NOT EXECUTE. /
1767 011517  EM1205::  .ASCIZ / DIAG FAIL BIT BAD/
1768 011543  EM1301::  .ASCIZ /FAIL SEL TEST TEST FAILED/
1769 011576  EM1302::  .ASCIZ /SELF-TEST ERROR REPORTING BAD/
1770 011635  EM1401::  .ASCIZ /ROM VERSION NUMBER TEST FAILED/
1771 011674  EM1402::  .ASCIZ /FIFO EMPTY, ONE OR MORE ROM VERSION NUMBERS MISSING/
1772 011762  EM1403::  .ASCIZ /ROM VERSION NUMBER FOUND OUT OF SEQUENCE/
1773 012035  EM1404::  .ASCIZ /ONE OR MORE ROM VERSION NUMBERS MISSING/
1774 012107  EM1405::  .ASCIZ /PROC_1/
1775 012122  EM1406::  .ASCIZ /PROC_2/
1776 012135  EM1407::  .ASCIZ /NOT FOUND/
1777 012147  EM1408::  .ASCIZ /FOUND/
1778 012155  EM1501::  .ASCIZ /CSR BIT 4 TEST FAILED/
1779 012203  EM1502::  .ASCIZ /CSR BIT 4 BAD/
1780 012221  EM1601::  .ASCIZ /TIMEOUT DORMANT WAITING FOR MASTER RESET TO CLEAR/
1781 012504  EM1604::  .ASCIZ /DEVICE REGISTER WORD READ/WRITE TEST FAILED/
1782 012560  EM1701::  .ASCIZ /DEVICE REGISTER WORD READ/MODIFY/WRITE TEST FAILED/
1783 012443  EM1801::  .ASCIZ /DEVICE REGISTER BYTE READ/WRITE TEST FAILED/
1784 012517  EM1901::  .ASCIZ /DEVICE REGISTER BYTE READ/MODIFY/WRITE TEST FAILED/
1785 012602  EM2001::  .ASCIZ /DEVICE STAT REGISTER ID BIT TEST FAILED/
1786 012652  EM2002::  .ASCIZ /ID BIT BAD. EXPECTED: SET, ACTUAL: CLEAR. /
1787 012725  EM2301::  .ASCIZ /TX_ENABLE (INACTIVE) BIT TEST FAILED/
1788 012772  EM2302::  .ASCIZ /TX_ENABLE BIT BAD ON LINE: /
1789 013030  EM2401::  .ASCIZ /TX_ENABLE (ACTIVE) BIT TEST FAILED/
1790 013073  EM2601::  .ASCIZ /RECEIVE INTERRUPT TEST FAILED/
1791 013131  EM2602::  .ASCIZ /NO RX INT GENERATED (DATA,VALID SET, RX INTS ENABLED)/
1792 013222  EM2603::  .ASCIZ /NO RX INT GENERATED (NO CODES IN FIFO AFTER RESET)/
1793 013310  EM2604::  .ASCIZ /NO RX INT GENERATED (RX_DATA_AVAILABLE, RX INTS ENABLED)/
1794 013443  EM2605::  .ASCIZ /RX INTERRUPT GENERATED WITH RX_DATA_AVAIL CLEAR/
1795 013467  EM2606::  .ASCIZ /TRANSMIT INTERRUPT TEST ERROR/
1796 013526  EM2607::  .ASCIZ /TX_ACTION SET REPEATEDLY AFTER BOARD RESET, NO DATA SENT/
1797 013622  EM2608::  .ASCIZ /TX_ACTION STUCK SET AFTER BOARD RESET/
1798 013673  EM2609::  .ASCIZ /TX INTERRUPT GENERATED WITH TX_ACTION CLEAR/
1799 013752  EM2610::  .ASCIZ /NO TX INTERRUPT WITH TX_ACTION SET AND TX INTS ENABLED/
1800 014044  EM2611::  .ASCIZ /TX_ACTION NOT SET AFTER CHAR SENT ON ALL LINES/
1801 014127  EM2612::  .ASCIZ /NO RX INT GENERATED (RX_DATA_AVAIL, RX INTS ENABLED)/
1802 014223  EM3001::  .ASCIZ /INTERCEPT BR LEVEL TEST FAILED/
1803 014262  EM3002::  .ASCIZ /NO RX_DATA_AVAILABLE FROM SELFTEST CODES IN FIFO AFTER RESET/
1804 014356  EM3003::  .ASCIZ /TX INTERRUPT GENERATED AT WRONG BR LEVEL/
1805 014432  EM3004::  .ASCIZ /RX INTERRUPT GENERATED AT WRONG BR LEVEL/
1806 014506  EM3005::  .ASCIZ /TX INTERRUPT GIVEN PRECEDENCE OVER SIMULTANEOUS RX INT. /
1807 014600  EM3101::  .ASCIZ /DIAGNOSTIC FIELD (BMP) TEST FAILED/
1808 014643  EM3102::  .ASCIZ /DIAGNOSTIC FIELD (BMP REQUEST) BAD ON LINE:
1809 014722  EM9014::  .ASCIZ /SUMMARY REPORTS FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:
1810 015016  EM9017::  .ASCIZ /FIFO WILL NOT PURGE (DATA,VALID STUCK SET)/
1811 015073  EM9201::  .ASCIZ /REMAINDER OF TEST SKIPPED/
1812 015127  EM9202::  .ASCIZ /NO CODE/
1813 015137  EM9203::  .ASCIZ /NON-SELFTEST/
1814 015154  EM9204::  .ASCIZ /SELFTEST ERROR CODE/
1815 015200  EM9205::  .ASCIZ /DATA CHARACTER/
1816 015217  EM9206::  .ASCIZ /MODEM STATUS CODE/
1817 015241  EM9207::  .ASCIZ /SELFTEST CODE/
1818 015257  EM9301::  .ASCIZ /BMP CODES WERE REPORTED DURING THIS DIAGNOSTIC/
1819 015336  EM9302::  .ASCIZ /BMP CODE FOUND IN TEST/
1820 015366  EM9303::  .ASCIZ /THE LAST BMP CODE WAS FOUND IN TEST /
1821 015433  EM9304::  .ASCIZ /UNEXPECTED BMP CODES FOUND DURING THIS PASS/
1822 015463  EM9305::  .ASCIZ /AN unexpected number of BMPs was found in this pass/
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

Это ошибка в процедуре отчета об ошибках и содержит дополнительную информацию.

**INPUTS:**
- R5 - Флаг ошибки.
  - Если бит 0 установлен, произошла ошибка чтения.
  - Если бит 1 установлен, произошла ошибка записи.

**OUTPUTS:**
- Сообщения печатаются на панели оператора.

**CALLING SEQUENCE:**
- Включите метку "ERO101" в качестве маркера обращения.
- Параметр в ошибках отчета процедуры DRS.

**COMMENTS:**
- Субпроцедуры, используемые: None.

**SOURCE CODE:**

```
BGMSG ERO101
SAVE JSR R5,PREG5 ;CALL REGISTER SAVE SUBRT.

MOV @BIT0,RO ;SET-UP BIT MAP FOR REPORT EXT'D ERROR INFO
BIC OPTION,RO ;TRY AND CLEAR THE FLAG.
BNE 61 ;EXIT IF OPTION NOT SELECTED.

61: REPORT EXTENDED ERROR INFORMATION

BIT #BIT0,R5 ;TEST FOR READ ERROR.
BEQ 21 ;SKIP READ ERROR MSG IF NO READ ERROR.
PRINTB #MSG1 ;PRINT READ ERROR MESSAGE.

MOV #MSG1,-(SP)
MOV @1,(SP)
MOV SP,R0
TRAP C1PNTB
ADD #4,SP

21: BIT #BIT1,R5 ;TEST FOR WRITE ERROR.
BEQ 41 ;SKIP WRITE ERROR MSG IF NO WRITE ERROR.
PRINTB #MSG2 ;PRINT WRITE ERROR MESSAGE.

MOV #MSG2,-(SP)
MOV @1,(SP)
MOV SP,R0
TRAP C1PNTB
ADD #4,SP

41: PRINTX #MSG3 ;REPORT THAT DHU MAY BE AT WRONG ADDRESS.

MOV #MSG3,-(SP)
MOV @1,(SP)
MOV SP,R0
TRAP C1PNTX
ADD #4,SP
```
61: PASS
ENDMSG
JSR PC,(SP),
L10002: TRAP C+MSG

015622 004 045 101 102 MSG1: ASCIZ /#ABUS TIME OUT TRAP CAUSED BY READ ATTEMPT.
015624 004 045 125 040
015624 005 125 123 115
015624 006 125 095 117
015624 007 125 040 040
015624 008 125 122 101
015624 009 120 040 103
015624 010 101 125 123
015624 011 105 104 040
015624 012 102 131 040
015624 013 122 105 101
015624 014 104 040 101
015624 015 124 124 105
015624 016 115 120 124
015624 017 056 045 116
015703 000 000

015626 045 101 102
015631 125 123 040
015634 124 111 115
015637 105 055 117
015642 125 124 040
015645 124 122 101
015650 120 040 103
015653 101 125 123
015656 105 104 040
015661 102 131 040
015664 122 105 101
015667 104 040 101
015672 124 124 105
015675 115 120 124
015700 056 045 116

015704 045 101 102 MSG2: ASCIZ /#ABUS TIME-OUT TRAP CAUSED BY WRITE ATTEMPT.
015707 125 123 040
015712 124 111 115
015715 105 055 117
015720 125 124 040
015723 124 122 101
015726 120 040 103
015731 101 125 123
015734 105 104 040
015737 102 131 040
015742 127 122 111
015745 124 105 040
015750 101 124 124
015753 105 115 120
015756 124 056 045
015761 116 000

015763 045 101 104 MSG3: ASCIZ /#ADHU MAY BE AT THE WRONG UNIBUS ADDRESS.
015766 110 125 040
015771 115 101 131
015774 040 102 105
015777 040 101 124
016002 040 124 110
016005 105 040 127
016010 122 117 116
016013 107 040 125
016016 116 111 102
016021 125 123 040
016024 101 104 104
016027 122 105 123
016032 123 056 045
016035 116 045 116
016040 000

01889 .EVEN
GLOBAL ERROR REPORTING ROUTINE

**SBJTL** GLOBAL ERROR REPORTING ROUTINE

* THIS IS AN ERROR REPORTING SUBROUTINE WHICH PRINTS 2 CONTIGUOUS
* ASCII ERROR MESSAGES. THE ADDRESS OF THE FIRST MESSAGE IS PASSED
* AS AN INPUT PARAMETER AND THE ADDRESS OF THE SECOND IS FOUND BY
* SEARCHING FOR THE END OF THE FIRST MESSAGE. THE MESSAGES ARE ONLY
* PRINTED IF EXT'D ERROR REPORTING HAS BEEN REQUESTED.
* INPUTS: R1 ADDRESS OF THE FIRST MESSAGE TO PRINT.
* OUTPUTS: A MESSAGES IS PRINTED AT THE OPERATOR CONSOLE.
* INCLUDE THE LABEL "ERO201" AS THE MESSAGE POINTER
* PARAMETER IN THE DIAG SUPER ERROR REPORT MACRO CALL.
* COMMENTS: THE MESSAGE IS PRINTED AS BASIC ERROR INFORMATION.
* THE SECOND MESSAGE SHOULD FOLLOW THE FIRST ONE IN THE PROGRAM
* MEMORY. EACH MESSAGE SHOULD BE DEFINED USING .ASCIIT
* SUBORDINATE ROUTINES USED: NONE.

BGNMSG ERO201

SAVE
JSR R5,PREG05 CALL REGISTER SAVE SUBRT.
MOV @BIT06,RO TRY TO CLEAR THE
BIC OPTION,RO EXT'D ERROR REPORTING FLAG
BNE 4# EXIT IF FLAG NOT SET.
MOV R1,R2 CHECK FOR A ZERO BYTE (END OF MESSAGE).
BNE 2# LOOP UNLST NEXT MESSAGE IS FOUND.
PRINTB @EOF0503,R1 PRINT THE FIRST MESSAGE.
PRINTB @EOF0503,R2 PRINT THE SECOND MESSAGE.
4#: PASS JSR R2,(SP) RESTORE THE GPR CONTENTS.
ENDMSG RET TURN TO PREG05 SUBRT.
MOV R2,(SP)
MOV @EOF0503,(SP)
MOV @SP,(SP)
MOV SP,R0
TRAP CAPN18
ADD @6,SP
TRAP CAPN18
ADD @6,SP
L10003: TRAP C1MSG
GLOBAL ERROR REPORTING ROUTINE

**SBTTL** GLOBAL ERROR REPORTING ROUTINE

****

**THIS IS AN ERROR REPORTING SUBROUTINE WHICH PRINTS AN ADDITIONAL ERROR**

**MESSAGE WHOSE ADDRESS IS PASSED AS AN INPUT PARAMETER, PROVIDED**

**EXTENDED ERROR REPORTING HAS BEEN REQUESTED.**

**INPUTS:**

**R1 ADDRESS OF THE MESSAGE TO PRINT.**

**OUTPUTS:**

**A MESSAGES IS PRINTED AT THE OPERATOR CONSOLE.**

**CALLING SEQUENCE:**

**LOAD THE ADDRESS OF THE MESSAGE IN R1.**

**INCLUDE THE LABEL "ERO503" AS THE MESSAGE POINTER**

**PARAMETER IN THE DIAG SUPER ERROR REPORT MACRO CALL.**

**COMMENTS:**

**THE MESSAGE IS PRINTED AS BASIC ERROR INFORMATION.**

**SUBORDINATE ROUTINES USED: NONE.**

**------------------------------------------------------------------------**

```
     BGNMSG ERO503
     MOV #BIT06,RO  ;TRY TO CLEAR THE
     BIC OPTION,RO  ;EXIT'D ERROR REPORTING FLAG
     BNE 2$  ;EXIT IF FLAG NOT SET.
     PRINTB #EF0503,91  ;PRINT THE MESSAGE.
     MOV  R1, (SP)
     MOV  #EF0503,(SP)
     MOV  #R2,(SP)
     MOV  SP,RO
     TRAP  C#PNTB
     ADD  #6,SP
     ENDMSSG
     L10004:
     TRAP  C#MSG
```

**------------------------------------------------------------------------**
GLOBAL ERROR REPORTING ROUTINE

*:SBTIL GLOBAL ERROR REPORTING ROUTINE
*:E0504
*:--------------------------------------------------------
*: THIS IS AN ERROR REPORTING SUBROUTINE WHICH PRINTS ADDITIONAL ERROR
*: MESSAGES WHEN ILLEGAL INTERRUPTS ARE RECEIVED.
*: INPUTS:
*: R1 ADDRESS OF THE MESSAGE TO PRINT.
*: R2 NUMBER OF ILLEGAL INTERRUPTS RECEIVED.
*: OUTPUTS:
*: MESSAGES ARE PRINTED AT THE OPERATOR CONSOLE.
*: CALLING SEQUENCE:
*: LOAD THE NUMBER OF ILLEGAL INTS IN R2.
*: INCLUDE THE LABEL "E0504" AS THE MESSAGE POINTER
*: PARAMETER IN THE DIAG SUPER ERROR REPORT MACRO CALL.
:* COMMENTS:
:* SUBORDINATE ROUTINES USED: NONE.

1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982 016174
1983 016174
1984 016174 012700 000100
1985 016200 046700 164020
1986 016204 001022
1987
1988 016206
1989 016230
1990 016252
1991 016252
104423

;BGMSG E0504

ER0504::

MOV @BIT06,RO ;TRY TO CLEAR THE
BNE OPTION,RO ;EXIT ERROR REPORTING FLAG
$2 ;EXIT IF FLAG NOT SET.

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

MOV R1, -(SP)
MOV @EF0503, -(SP)
MOV @2, -(SP)
MOV SP, R0
TRAP C#PNTX
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

2#: ENDMGS

L10005:
TRAP C#MSG
GLOBAL ERROR REPORTING ROUTINE

**SBTL: GLOBAL ERROR REPORTING ROUTINE**

**ER1401**

- **INPUTS:**
  - R1 - CONTAINS THE ADDRESS OF THE FIRST MESSAGE TO BE REPORTED.
  - R3 - CONTAINS THE ROM VERSION NUMBER OF PROC.1.
  - R4 - CONTAINS THE ROM VERSION NUMBER OF PROC.2.

- **OUTPUTS:**
  - BASIC AND EXTENDED ERROR MESSAGES ARE REPORTED AT THE OPERATORS CONSOLE.

- **CALLING SEQUENCE:**
  - INCLUDE THE LABEL "ER1401" AS THE MESSAGE POINTER PARAMETER IN THE DRS ERROR REPORT MACRO CALL.

- **COMMENTS:**

- **SUBORDINATE ROUTINES USED:** NONE.

---

```
BGNMSG ER1401

ER1401:

MOV @B106,RO ;TRY TO CLEAR THE
BIC OPTION,RO ;EXIT IF ERROR REPORTING FLAG
BNE 601 ;EXIT IF FLAG NOT SET.

PRINTB @EF0503,R1 ;REPORT THE ERROR MESSAGE PASSED IN.

MOV R1,-(SP)
MOV @EF0503,-(SP)
MOV @R2,-(SP)
MOV SP,RO
TRAP CPIVTB
ADD @R6,SP

/* DETERMINE WHICH ROM VERSION NUMBER(S) ARE MISSING. */

MOV @R99,R5 ;GET INVALID ROM NUMBER.
MOV @EM1405,R1 ;SELECT PROC.1 MESSAGE.
CMPB R3,R5 ;CHECK PROC.1 ROM VERSION NUMBER.
BEQ 2$: ;GO REPORT PROC.1 CODE NOT FOUND.
MOV @EM1407,R2 ;SELECT "NOT FOUND" MESSAGE.
2$: JSR PC,50$ ;GO REPORT MESSAGE.

MOV @EM1406,R1 ;SELECT PROC.2 MESSAGE.
MOV @EM1407,R2 ;SELECT THE "NOT FOUND" MESSAGE.
CMPB R4,R5 ;CHECK PROC.2 ROM VERSION NUMBER.
BEQ 4$: ;GO REPORT PROC.2 CODE NOT FOUND.
MOV @EM1408,R2 ;SELECT "FOUND" MESSAGE.
4$: JSR PC,50$ ;GO REPORT THE MESSAGE.

BR 60$ ;EXIT.
```
50$:  PRINTX  @EF1402,R1,R2  ;REPORT THE MESSAGE.
    MOV    R2, (SP)
    MOV    R1, (SP)
    MOV    @EF1402, (SP)
    MOV    @3, (SP)
    MOV    SP, R0
    TRAP   C$PRINTX
    ADD    @10, SP

60$:  ENDM SG
    RTS    PC
    RETURN.

L10006:  TRAP   C$MSG
GLOBAL ERROR REPORTING ROUTINE

**DESCRIPTION:**

This is an error reporting subroutine which prints additional error information if an error is detected in one of the device registers. Access tests, provided extended error reporting has been requested.

**INPUTS:**
- R1 - actual data (unused bits set to 0).
- R2 - expected data (unused bits set to 0).
- R3 - offset (in bytes) to the register being tested.
- R5 - line number of register being tested.
- RMatbb - label at base of register message address table.

**OUTPUTS:**
- Messages are printed at the operator's console.

**CALLING SEQUENCE:**
- Include the label `ER1601` as the message pointer parameter in the DRS error report macro call.

**COMMENTS:**
- None

**SUBORDINATE ROUTINES CALLED:**
- None

```
2048 .SBTTL GLOBAL ERROR REPORTING ROUTINE
2049
2050 .GLOBAL ER1601
2051
2052 .** THIS AN ERROR REPORTING SUBROUTINE WHICH PRINTS ADDITIONAL ERROR
2053 .** INFORMATION IF AN ERROR IS DETECTED IN ONE OF THE DEVICE REGISTER
2054 .** ACCESS TESTS, PROVIDED EXTENDED ERROR REPORTING HAS BEEN REQUESTED.
2055 .** THIS SUBROUTINE REPORTS THE ACTUAL AND EXPECTED FROM THE DEVICE
2056 .** REGISTER(S) WHICH IS ARE IN FAULTY.
2057
2058 .** INPUTS: R1 - ACTUAL DATA (UNUSED BITS SET TO 0).
2059 .** R2 - EXPECTED DATA (UNUSED BITS SET TO 0).
2060 .** R3 - OFFSET (IN BYTES) TO THE REGISTER BEING TESTED.
2061 .** R5 - LINE NUMBER OF REGISTER BEING TESTED.
2062 .** RMATBB - LABEL AT BASE OF REGISTER MESSAGE ADDRESS TABLE.
2063
2064 .** OUTPUTS: MESSAGES ARE PRINTED AT THE OPERATORS CONSOLE.
2065
2066 .** CALLING SEQUENCE: INCLUDE THE LABEL "ER1601" AS THE MESSAGE POINTER
2067 .** PARAMETER IN THE DRS ERROR REPORT MACRO CALL.
2068
2069 .** COMMENTS: NONE
2070
2071 016416
2072 016416 012700 000100
2073 016422 046700 163576
2074 016426 001036
2075 016426 000000
2076 016430 016304 002262
2077 016434 0010546
2078 016436 0010446
2079 016440 012746 004451
2080 016444 012746 000003
2081 016450 010600
2082 016452 104414
2083 016454 062706 000010
2084 016460 010246
2085 016462 012746 004345
2086 016466 012746 000002
2087 016472 010600
2088 016474 104415
2089 016476 062706 000006
2090 016480 010146
2091 016482 012746 004407
2092 016486 012746 000002
2093 016490 010600
2094 016492 104415
2095 016494 062706 000006
2096 016526 016524 016524
2097 016524 010146
2098 016524 012746 004407
2099 016526 012746 000002
2010 016530 010600
2011 016532 104415
2012 016534 062706 000006
2013 21 ENDMSG
2014 L10007:
```

```
C:\CZD\M80\F\M120015\Mar 84 09:15 PAGE 45

GLOBAL ERROR REPORTING ROUTINE

*SBTTL GLOBAL ERROR REPORTING ROUTINE
*ER1603
*------------------------------------------------------------
* THIS ERROR REPORTING ROUTINE IS USED TO PRINT A BASIC ERROR
* MESSAGE, ALONG WITH A MESSAGE INFORMING THE OPERATOR WHICH TEST IS
* ABOUT TO BE ABORTED, PROVIDED EXTENDED ERROR INFORMATION HAS BEEN
* REQUESTED, OTHERWISE ONLY A "TEST FAILURE" MESSAGE WILL BE PRINTED.
*
* INPUTS: R1 CONTAINS THE ADDRESS OF THE MESSAGE TO BE PRINTED.
* ERRMGRX CONTAINS THE ADDRESS OF THE MESSAGE THAT INDICATES
* THE TEST THAT IS BEING PERFORMED, CG DMA, BREAK ETC.
* OUTPUTS: MESSAGES ARE PRINTED AT THE OPERATORS CONSOLE.
* "TESTNAME TEST ABORTED"
* CALLING SEQUENCE: INCLUDE THE LABEL "ER1603" AS THE MESSAGE POINTER
* PARAMETER IN THE DRS ERROR REPORT MACRO CALL.
* COMMENTS:
* SUBORDINATE ROUTINES CALLED: NONE.
*
BGMSGR ER1603
SAVE
JSR R5,PREGS
;CALL REGISTER SAVE SUBRT.

MOV @BIT06,RO
;TRY TO CLEAR THE
BIC OPTION,RO
;EXT. ERROR REPORTING FLAG
BNE 2:
;EXIT IF FLAG NOT SET.

PRINTB @EF0503,R1
;PRINT BASIC MESSAGE ON OPERATORS CONSOLE.

MOV R1,(SP)
MOV @EF0503,(SP)
MOV #2,(SP)
MOV SP,RO
TRAP CIPTNB
ADD #6,SP

MOV ERMSGR,R2
;GET THE "TEST MESSAGE"
PRINTB @EF1601,R2
;PRINT "TEST ABORTED" MESSAGE.

MOV R2,(SP)
MOV @EF1601,(SP)
MOV #2,(SP)
MOV SP,RO
TRAP CIPTNB
ADD #6,SP

2:
PASS

JSR PC, @B,(SP)

ENDMSGR

L10010:
TRAP CIUSSG
Global Error Reporting Routine

This is an error reporting subroutine which is intended for use in the interrupt BR level test. It reports additional information when an interrupt has occurred at the wrong BR level. Unless extended error reporting has been requested, only the test fail message will be printed.

- **Inputs:**
  - R1 - Address of message to print first.
  - R4 - BR level at which the INT request occurred.
  - R5 - Expected or correct BR level for the DUT.

- **Outputs:**
  - A message is printed at the operator console.

- **Calling Sequence:** Include the label "ER3001" as the message pointer parameter in the diag super error report macro call.

- **Comments:** The message is printed as basic and extended error information.

- **Subordinate Routines Used:** None.

---

BGNMSG ER3001

ER3001:

M0V @BITO8,RO ; Try to clear the
BIC OPTION,RO ; Exit: D error reporting flag
BNE 2$ ; Exit if flag not set.

PRINTB @EF0503,R1 ; Print the first line of the message.

M0V R1,(SP)
M0V @EF0503, (SP)
M0V @EF0503, (SP)
M0V SP,R0
TRAP CIPNT8
ADD #6,SP

PRINTX @EF3001,R5 ; Report expected BR level.

M0V R5,(SP)
M0V @EF3001, (SP)
M0V @EF3001, (SP)
M0V SP,R0
TRAP CIPNTX
ADD #6,SP

PRINTX @EF3002,R4 ; Report actual BR level.

M0V R4,(SP)
M0V @EF3002, (SP)
M0V @EF3002, (SP)
M0V SP,R0
TRAP CIPNTX
ADD #6,SP

ENDMSG

L10011:

TRAP CIPNTG
GLOBAL ERROR REPORTING ROUTINE

**.*BTTL GLOBAL ERROR REPORTING ROUTINE ER9004
**** THIS IS AN ERROR REPORTING SUBROUTINE WHICH REPORTS ERROR SUMMARIES
**** FOR LINES WHICH HAVE EXCEEDED THE SPECIFIED MAXIMUM NUMBER OF
**** INDIVIDUAL RECEPTION ERRORS, PROVIDED EXTENDED ERROR REPORTING HAS
**** BEEN REQUESTED BY THE OPERATOR.
****
**** INPUTS: 
**** R1 - ADDRESS OF MESSAGE TO PRINT FIRST.
**** ERCNB - LABEL AT BASE OF LINE ERROR COUNTERS TABLE.
**** ERSRF - "REPORT ERROR SUMMARY FOR LINE" FLAGS.
****
**** OUTPUTS: 
**** A MESSAGE IS PRINTED AT THE OPERATOR CONSOLE.
****
**** CALLING SEQUENCE: 
**** INCLUDE THE LABEL "ER9004" AS THE MESSAGE POINTER
**** PARAMETER IN THE DIAG SUPER ERROR REPORT MACRO CALL.
****
**** COMMENTS: 
**** THE MESSAGE IS PRINTED AS BASIC AND EXTENDED ERROR INFORMATION.
**** THE CONTENTS OF GPR'S R2, R3, R4, AND R5 ARE DESTROYED.
****
**** SUBORDINATE ROUTINES USED: NONE.
****

BGNMSG ER9004

ER9004:

MOV #BIT06,RO ;TRY TO CLEAR THE
BIC OPTION,RO ;EXIT 'D ERROR REPORTING FLAG
BNE 6$ ;EXIT IF FLAG NOT SET.

PRINTB #EF0503,#EM9014 ;REPORT THE SECONDARY ERROR MESSAGE.

MOV #EM9014,-(SP)
MOV #EF0503,-(SP)
MOV #R2,-(SP)
MOV SP,RO
TRAP C1PNTB
ADD #6,SP

CLR R2 ;CLEAR THE LINE COUNTER.

MOV ERSRF,R3 ;GET THE ERROR SUMMARY FLAGS.

CLR R4 ;CLEAR "LINE COUNTER TIMES 2" OFFSET.

CLR ;CLEAR THE CARRY FOR THE FOLLOWING ROTATE.

ROR R3 ;SHIFT ANOTHER ERROR SUMMARY FLAG INTO CARRY.

BCC 4$ ;SKIP PRINTING MESSAGE IF FLAG FOR LINE CLEAR.

PRINTX #EF9010,R2,ERCNB(R4)

MOV ERCNB(R4),-(SP)
MOV R2,-(SP)
MOV #EF9010,-(SP)
MOV #R3,(SP)
MOV SP,RO
TRAP C1PNTX
ADD #10,SP

4$: MOV (R4),R5 ;INCREMENT THE LINE OFFSET BY 2.

INC R2 ;INCREMENT THE LINE COUNTER.

TST R3 ;CHECK THE ERROR SUMMARY FLAGS.

BNE 2$ ;IF MORE FLAGS SET, LOOP TO DO OTHER LINES.

L10012: TRAP C1MSG
GLOBAL ERROR REPORTING ROUTINE

**SBTL** GLOBAL ERROR REPORTING ROUTINE

:**********:*:

: THIS IS AN ERROR REPORTING SUBROUTINE WHICH IS USED TO REPORT THAT:
: SOMETHING OTHER THAN A SELFTEST CODE WAS FOUND IN A SELFTEST CODE:
: FIFO SLOT DURING THE REMOVAL OF THE SELFTEST CODES FROM THE FIFO.
: THIS ROUTINE IS USED BY THE RSTPT ROUTINE. EXTENDED ERROR INFORMATION:
: IS GIVEN ONLY WHEN REQUESTED IN THE SOFTWARE QUESTIONS.

::INPUTS::
: R1 - ADDRESS OF ERROR MESSAGE QUALIFIER STRING.
: R2 - INCORRECT CODE AS READ FROM THE SELFTEST CODE FIFO SLOT.
: R3 - LINE NUMBER ASSOCIATED WITH THE SELFTEST FIFO SLOT.

::OUTPUTS::
: A MESSAGE IS PRINTED AT THE OPERATOR CONSOLE.

::CALLING SEQUENCE::
: INCLUDE THE LABEL "ER9007" AS THE MESSAGE POINTER
: PARAMETER IN THE DIAG SUPER ERROR REPORT MACRO CALL.

::COMMENTS::
: THE MESSAGE IS PRINTED AS BASIC AND EXTENDED ERROR INFORMATION.

::SUBORDINATE ROUTINES USED::
: NONE.

:**********:*:

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Address</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>017036</td>
<td>BGNSRG</td>
<td>ER9007</td>
<td></td>
</tr>
<tr>
<td></td>
<td>017036</td>
<td></td>
<td></td>
</tr>
<tr>
<td>017042</td>
<td>MOV</td>
<td>@BIT06,RO</td>
<td></td>
</tr>
<tr>
<td>017046</td>
<td>BIC</td>
<td>OPTION,RO</td>
<td></td>
</tr>
<tr>
<td>017054</td>
<td>BNE</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>017058</td>
<td>BIC</td>
<td>@177760,R3</td>
<td></td>
</tr>
<tr>
<td>017064</td>
<td>PRINTB</td>
<td>@EF9018,R1,R3</td>
<td></td>
</tr>
<tr>
<td>017068</td>
<td>PRINTX</td>
<td>@EF9017,R1,R2</td>
<td></td>
</tr>
<tr>
<td>017070</td>
<td>ADD</td>
<td>@010,SP</td>
<td></td>
</tr>
<tr>
<td>017072</td>
<td>ADD</td>
<td>@010,SP</td>
<td></td>
</tr>
<tr>
<td>017074</td>
<td>TRAP</td>
<td>C1PMTB</td>
<td></td>
</tr>
<tr>
<td>017076</td>
<td>ADD</td>
<td>@010,SP</td>
<td></td>
</tr>
<tr>
<td>017080</td>
<td>MOV</td>
<td>$8,-(SP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MOV</td>
<td>$R1,-(SP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MOV</td>
<td>@EF9018,-(SP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MOV</td>
<td>@EF9017,-(SP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MOV</td>
<td>05074</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MOV</td>
<td>@010,SP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRAP</td>
<td>C1PMTX</td>
<td></td>
</tr>
<tr>
<td>017100</td>
<td>ENDMSG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>017104</td>
<td></td>
<td>104423</td>
<td></td>
</tr>
<tr>
<td>017108</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017112</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017114</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017116</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>017120</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
.SBIT LGLOBAL ERROR REPORTING ROUTINE

**THIS IS AN ERROR REPORTING SUBROUTINE WHICH IS USED TO REPORT THAT**
**AN UNSPECTED CODE OR CHARACTER HAS BEEN FOUND IN THE INPUT RECEIVED**
**CHARACTER FIFO. THE ADDITIONAL ERROR IS REPORTED ONLY IF REQUESTED**
**DURING THE SOFTWARE QUESTIONS.**

**INPUTS:**
R1 ADDRESS OF PARTIAL ERROR MESSAGE STRING.
R2 INCORRECT CODE AS READ FROM THE SELFTEST CODE FIFO SLOT.

**OUTPUTS:**
A MESSAGE IS PRINTED AT THE OPERATOR CONSOLE.

**CALLING SEQUENCE:**
INCLUDE THE LABEL "ER9008" AS THE MESSAGE POINTER.
PARAMETER IN THE DIAG SUPER ERROR REPORT MACRO CALL.

**COMMENTS:**
THE MESSAGE IS PRINTED AS BASIC AND EXTENDED ERROR INFORMATION.

**SUBORDINATE ROUTINES USED:** NONE.

```
BGNMSG ER9008

ER9008:

MOV @01106,RO ;TRY TO CLEAR THE
BIC OPTION,RO ;EXT.D ERROR REPORTING FLAG
BNE 2$ ;EXIT IF FLAG NOT SET.

;EXTRACT THE LINE NUMBER FROM THE INCORRECT CODE OR CHARACTER WHICH WAS READ
;FROM THE SELFTEST CODE FIFO SLOT.

MOV R2,R3
SWAB R3
BIC 017760,R3 ;CALCULATE LINE NUMBER OF CODE FOUND.
PRINTB @0F9016,R1,R3 ;REPORT TYPE OF INCORRECT CODE FOUND.

PRINTX @0F9017,R1,R2 ;REPORT THE ACTUAL INCORRECT CODE.
```

L10014: TRAP CMMSG
.SBTL GLOBAL ERROR REPORTING ROUTINE
**THIS IS A GENERAL ERROR REPORTING SUBROUTINE WHICH REPORTS A MESSAGE**
**WHICH TAKES A SINGLE, 2 DIGIT DECIMAL ARGUMENT AFTER THE END OF AN**
**ASCII MESSAGE.**
**INPUTS: R1 VALUE TO BE PRINTED AFTER MSG AS 2 DECIMAL DIGITS.**
**H2 ADDRESS OF MESSAGE TO PRINT FIRST.**
**OUTPUTS: A MESSAGES IS PRINTED AT THE OPERATOR CONSOLE.**
**CALLING SEQUENCE: INCLUDE THE LABEL "ER9101" AS THE MESSAGE POINTER**
**PARAMETER IN THE DIAG SUPER ERROR REPORT MACRO CALL.**
**COMMENTS: THE MESSAGE IS PRINTED AS BASIC ERROR INFORMATION.**
**SUBORDINATE ROUTINES USED: NONE.**

;BEGIN ROUTINE ER9101

BGNMSG ER9101

ER9101::

MOV @81106,RO ;TRY TO CLEAR THE
BIC OPTION,RO ;EXT'D ERROR REPORTING FLAG
BNE 21 ;EXIT IF FLAG NOT SET.

PRINTB @EF9006,R2,R1 ;REPORT THE STRING FOLLOWED BY THE NUMBER.

MOV R1, (SP)
MOV R2, -(SP)
MOV @EF9006, -(SP)
MOV @3, -(SP)
MOV SP, RO
TRAP C$PRINT
ADD @10, SP

ENDMSG 21:

L10015: TRAP C$MSG
GLOBAL ERROR REPORTING ROUTINE

.SBTIL  GLOBAL ERROR REPORTING ROUTINE  ER9301.

******************************************************************************************
**  THIS IS AN ERROR REPORTING SUBROUTINE WHICH PRINTS ANY BMP CODES               **
**  THAT ARE FOUND IN THE BMP CODE QUEUE, TOGETHER WITH THE THE NUMBER OF         **
**  THE TEST THAT WAS EXECUTING AT THE TIME THE BMP CODE WAS LOGGED.              **
**  PROVIDED EXTENDED ERROR REPORTING HAS BEEN ENABLED.                           **
**                                                                                   **
**  INPUTS:  R1  THE ADDRESS OF THE FIRST MESSAGE TO BE REPORTED.                **
**  R2  THE ADDRESS OF THE NEXT EMPTY CELL IN THE QUEUE.                         **
**                                                                                   **
**  OUTPUTS:  THE TEST NUMBER FOLLOWED BY THE BMP CODE ARE PRINTED AT THE        **
**  OPERATOR CONSOLE.                                                             **
**                                                                                   **
**  CALLING SEQUENCE:  INCLUDE THE LABEL "ER9301" AS THE MESSAGE POINTER          **
**  PARAMETER IN THE DIAG SERROR REPORT MACRO CALL.                             **
**                                                                                   **
**  COMMENTS:  THE MESSAGE IS PRINTED AS BASIC ERROR INFORMATION.                **
**                                                                                   **
**  SUBORDINATE ROUTINES USED: NONE.                                              **
******************************************************************************************

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

MOV  @BIT06.R0 ;TRY TO CLEAR THE
BIC  OPTION.R0 ;EXIT IF ERROR REPORTING FLAG
BNE  601 ;EXIT IF FLAG NOT SET.

PRINTB  @EF0503.R1 ;REPORT UNEXPECTED BMP CODES FOUND.

MOV  R2,(SP)
MOV  @EF0503,(SP)
MOV  SP,R0
TRAP  CPNPTR
ADD  #$6,SP

MOV  @BMPQCR.R5 ;GET THE START ADDRESS OF THE BMP CODE QUEUE.

MOV  @EM9302.R5 ;GET THE MESSAGE TO BE REPORTED.

MOV  (R3),R1 ;GET THE NUMBER OF THE TEST THAT WAS EXECUTING.

MOV  (R3),R4 ;GET BMP CODE THAT WAS REPORTED OFF THE QUEUE.

JSR  PC,501 ;GO REPORT BMP CODE.

CMP  R3,R2 ;CHECK IF ALL CODES HAVE BEEN REPORTED.
BLD  2$ ;IF IT IS NOT THE LAST BMP CODE THEN LOOP.

; CHECK IF OVERFLOW HAS OCCURRED.
; THE CONDITIONS FOR OVERFLOW ARE: THE POINTER CONTAINS THE ADDRESS OF THE
; LAST CELL IN THE QUEUE, AND A BMP CODE HAS ALREADY BEEN WRITTEN INTO THAT
; CELL.

CMP  R2,@BMPQCR.E-4 ;CHECK IF THE POINTER IS AT THE LAST LOCATION.
BNE  601 ;EXIT IF NOT AT THE LAST LOCATION.
TST  2(R2) ;CHECK FOR A BMP CODE IN THE LAST CELL
BEQ  601 ;EXIT IF NO OVERFLOW HAS OCCURRED, CELL EMPT.

MOV  (R3),R1 ;GET THE TEST NUMBER OFF THE QUEUE.
MOV  (R3),R4 ;GET BMP CODE OFF THE QUEUE.
MOV  @EM9303,R5 ;SELECT THE MESSAGE TO BE REPORTED.
PRINTX @EF9302 ;REPORT OVERFLOW CONDITION.
MOV @EF9302. (SP)
MOV @1. (SP)
MOV SP, RO
TRAP CIPNTY
ADD #4, SP
JSR PC, 50:
;REPORT THE LAST BMP CODE PLACED ON THE QUEUE.
BR 60:
;EXIT.

PRINTX @EF9301, R5, R1, R4 ;PRINT THE MESSAGE.
MOV R4.-(SP)
MOV R1.-(SP)
MOV R5.-(SP)
MOV @EF9301.-(SP)
MOV #4.-(SP)
MOV SP, RO
TRAP CIPNTX
ADD #12, SP
PASS PC ;RETURN.
JSR PC, @SP; ;RESTORE THE GPR CONTENTS.

ENDMSG

L10016: TRAP CMSG
GLOBAL SUBROUTINES SECTION

."SBITL  GLOBAL SUBROUTINES SECTION

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

**.SBTTL GLOBAL SUBROUTINE**

**.ALITLD**

; ALTER DEVICE REGISTER FIELDS ROUTINE

; THIS SUBROUTINE ALTERS THE SPECIFIED FIELD OF THE SPECIFIED DEVICE

; REGISTER FOR THE SPECIFIED LINES. THIS ROUTINE CAN BE USED TO SET

; OR CLEAR BITS WITHIN SELECTED FIELDS OF SELECTED REGISTERS.

; USE EXAMPLES: SET RX.BAUD.RATE FIELDS ON LINES 3 AND 6.

; CLEAR TX.DMA BITS ON ALL LINES.

; INPUTS:

; R1 - ADDRESS OF THE REGISTERS TO ALTER.

; R2 - BIT FIELDS SET TO DESIRED STATES.

; R3 - BIT MAP OF LINES FOR WHICH TO ALTER REGISTER.

; R4 - MASK OF BITS TO ALTER (1 INDICATES CHANGE BIT).

; CSA - CONTAINS THE ADDRESS OF THE DEVICE CSA.

; ISTAT - SAVED STATES OF THE INTERRUPT ENABLE BITS.

; OUTPUTS:

; DEVICE REGISTERS - SPECIFIED REGISTER FIELDS ALTERED.

; CSR IND.RDR.REG FIELD - DESTROYED.

; CALLING SEQUENCE: JSR PC, ALITLD

; COMMENTS:

; THIS ROUTINE READS THE SPECIFIED REGISTERS FOR ALL LINES

; WITH NUMBERS LOWER THAN THE HIGHER SPECIFIED LINE.

; THIS ROUTINE DOES NOT READ THE CSR.

; SUBRoutines CALLED: NONE.

ALITLD: SAVE

; SAVE CONTENTS OF GPDR R0 THRU R5.

JSR R5, PREG05 ; CALL REGISTER SAVE SUBRT.

SET UP TO LOOP FOR EACH LINE:

; PREPARE THE WORD TO BE ORED INTO THE REGISTER CONTENTS.

; SET UP THE WORD TO WRITE INTO THE IND.ADR.REG FIELD OF THE CSR.

; MOV R4, R0 ; CALCULATE THE NEW CONTENTS OF THE

; COM R0 ; REGISTER FIELDS WHICH ARE TO BE

; BIC R0, R2 ; ALTERED BY THIS ROUTINE.

; MOV ISTAT, R5 ; SET UP TO WRITE IND.ADR.REG FIELD TO 0.

; LOOP ONCE FOR EACH LINE, ALTERING THE SPECIFIED FIELD IN THE SPECIFIED

; REGISTER IF THE LINE HAS BEEN SELECTED FOR ALTERING.

; EXIT THE LOOP IF NO MORE LINES TO ALTER, OR IF WE HAVE ALTERED THE MAX

; ALLOWABLE NUMBER OF LINES (AS SPECIFIED BY NUMLNS).

; CLC

; PREPARE FOR ROTATE. "TST R5" DOES THIS BELOW.

2: ROR R3 ; GET THE LINE SELECT BIT FOR THIS LINE.

BCC 4: ; SKIP SETUP IF LINE IS NOT SELECTED.

MOV R5, GCSRA ; SET OUT CIR IND.ADR.REG FIELD TO THIS LINE.

MOV (R1).RO ; GET THE PRESENT CONTENTS OF THE REG TO ALTER.

BIC R4, R0 ; CLEAR THE BIT FIELDS WE ARE TO ALTER.

BIS R2, R0 ; FOR IN THE NEW STATE OF THE FIELDS.

MOV R0, (R1) ; WRITE THE NEW REGISTER CONTENTS TO THE REG.

INC R5 ; SET LINE NUMBER TO THE NEXT LINE.

TST R3 ; CHECK FOR UNHANDLED LINES. CLEAR CARRY FLAG.

BNE 2: ; LOOP IF SELECTED LINES(5) IS NOT HANDLED.
604: PASS  JSR  ;RESTORE GPR5.
   RTS  PC  ;RETURN TO PREGOS SUBRT.
   JSR  ;RETURN TO CALLING ROUTE.
**SBITL** GLOBAL SUBROUTINE

*CALMSL*

**CALMSL**

---

CALMSL:: SAVE R5,R5,PREG05 ;CALL REGISTER SAVE SUBRT.

CLR 62h ;CLEAR THE 2ND TIME FLAG.

; SYNCHRONIZE WITH THE LTC.

21h: MOV 01,R5 ;SET OUTER LOOP COUNTER TO 1 LOOP.

;INCREASE THE VALUE LOADED INTO THIS COUNTER IF THE ***
;FOLLOWINGLOOP FAILS ON FUTURE, FASTER PROCESORS. ***

CLR R0 ;CLEAR THE WAIT FOR CLOCK INT COUNTER.

MOV 01,TIMER1 ;SET UP COUNTER TO 1 TO SYNCH WITH LIC.

41h: TST TIMER1 ;CHECK FOR COUNTER HAVING GONE TO ZERO.

BEQ 63h ;JUMP OUT OF LOOP IF LTC HAS INTERRUPTED.

INC R0 ;COUNT THIS ITERATION OF THE INNER LOOP.

BNE 41h ;LOOP IF COUNTER HAS NOT TURNED OVER.

DEC R5 ;DECREMENT THE INNER LOOP COUNTER.

BGT 41h ;LOOP IF OUTER LOOP COUNT NOT UP.

; IF WE GO NO LTC INTERRUPT, INDICATE THAT THERE IS NO LTC AVAILABLE.

; LTC MUST BE FLAKY, OR NOT REALLY AN LTC AT ALL.

CLR CLKH2 ;CLEAR LTC FREQUENCY WORD TO INDICATE NO LTC.

CLC ;INDICATE FAILURE FOR RETURN.

BA 60h ;BYPASS THE FOLLOWING CALIBRATION PROCEDURES.

; WE ARE NOW SYNCHRONIZED WITH THE LTC.

; SET UP FOR THE CALIBRATION LOOP.

61h: MOV 0TIMER1,R4 ;WILL TEST TIMER1 IN THE LOOP BELOW.
CLR R1  ; CLEAR THE OUTER LOOP COUNTER.
CLR R2  ; INDICATE TO CHECK ALL BITS OF TIMER1.
CLR R3  ; INDICATE TO CHECK FOR TIMER1 CLEAR.
MOV #1,(R4) ; LOAD TIMER1 WITH COUNT OF 1.

81: MOV MSLCNT,R5  ; LOAD MS LOOP COUNT.
109: MOV (R4),R0  ; GET THE TIMER1 VALUE.
249: MOV R0,64  ; SAVE WORD (LIKE IN THE REAL LOOP).
264: BIC R2,R0  ; LEAVE ALL THE BITS.
312: CMP R0,R3  ; COMPARE AGAINST ZERO.
370: SEC  ; SET CARRY IN CASE OF SUCCESS.
400: BEQ 12$ ; EXIT LOOP IF TIMER1 HAS CLEARED.
418: DEC R5  ; COUNT DOWN THE INSIDE MS LOOP COUNT.
528: BNE 8$  ; LOOP IF MS NOT UP.
544: DEC R1  ; DECREMENT THE MS TIME COUNT.
604: BNE 81  ; KEEP LOOPING.
644: JSR PC,LOOPS  ; WE OVERFLOWED, SOMETHING IS WRONG, ABORT.

506: ; WE HAVE NOW HAVE LOOP COUNT INFORMATION FOR ONE CLOCK TICK.
507: ; WE HAVE NEGATIVE OF NUMBER OF OUTER LOOPS IN R1, EACH IS MSLCNT INNER LOOPS.
508: ; WE HAVE THE PORTION OF THE LAST OUTER LOOP NOT EXECUTED, IN R5.
510: ; NOW WE CALCULATE THE TOTAL NUMBER OF INNER LOOPS EXECUTED.

12$: NEG R1  ; GET NUMBER OF OUTER LOOPS.
123: MOV MSLCNT,R2  ; COPY NUMBER OF LOOPS FOR MULTIPLY.
141: MOV R2,R4  ; CALC # OF INNER LOOPS DONE IN LAST OUTER LOOP
158: MOV R2,R4  ; AND ADD TO ACCUMULATOR LSWORD.
175: CLR R5  ; CLEAR ACCUMULATOR MSWORD.

144: DEC R1  ; CHECK R1 FOR 0 CONDITION
151: BMI 16$  ; SKIP MULTIPLICATION IF ZERO
168: ADD R3,R4  ; MULTIPLY NUMBER OF INNER
224: ADD R5  ; LOOPS PER OUTER LOOP BY

14$ ; NUMBER OF OUTER LOOPS PERFORMED.

232:  ; DIVIDE THE TOTAL NUMBER OF INNER LOOPS BY THE NUMBER OF MS PER LITC TICK.
249:  ; # OF MS PER LITC TICK IS DIVISOR.

16$: MOV MSTICK,R1  ; LSWORD OF LOOP COUNT IS LSWORD OF DIVIDEND.
226: MOV R4,R3  ; MSWORD OF LOOP COUNT IS MSWORD OF DIVIDEND.
243: JSR PC,UNSDIV  ; DIVIDE NUMBER OF LOOPS BY MS PER LITC TICK.
260: BCS 18$  ; BYPASS OOPS IF WE'RE OK.
277: JSR PC,LOOPS  ; CLOCK ROUTINES ARE NOT LONG ENOUGH, OR BUG.
303: MOV R1,MSLCNT  ; SET NEW VALUE FOR MS LOOP COUNT.
320: CMP 63$  ; SET THE 2ND ITERATION FLAGS IF 1ST ITERATION.
337: BNE 2$  ; BRANCH IF ONLY ONE ITERATION DONE.
354: BNE 2$  ; SET THE SUCCESS FLAG FOR EXIT.
371:  ; RESTORE GPR'S.
388: RTS PC  ; RETURN TO "MSGOS SUBR".
395: JSR PC,(RSP)  ; CARRY SUCCESS FLAG, SET IF SUC.

583:  ; DUMMY WORD FOR STORAGE OF THE READ WORD.
GLOBAL SUBROUTINE

SBTL     GLOBAL SUBROUTINE

**CHECK TRAP ROUTINE**

**THIS SUBROUTINE IS USED TO CHECK FOR A BUS TIME-OUT TRAP (004 TRAP)
WHICH IS CAUSED BY AN ACCESS TO A NON EXISTENT MEMORY OR I/O LOCATION.
IF THE TRAP DOES NOT OCCUR, THIS ROUTINE RETURNS A SUCCESS INDICATION.

**INPUTS:**

- RO - SOURCE ADDRESS FOR MOVE.
- R1 - DESTINATION ADDRESS FOR MOVE.
- (RO) - SOURCE FOR THE MOVE.

**OUTPUTS:**

- (R1) - WRITTEN TO THE CONTENTS OF (RO).
- CARRY FLAG - SET ON RETURN IF NO 004 TRAP DETECTED.
- TP4FLG - NONZERO IF TRAP OCCURRED, CLEARED OTHERWISE.

**CALLING SEQUENCE:**

JSR PC,CKTRAP

**COMMENTS:**

IF THIS SUBROUTINE CAUSES A TRAP, EITHER THE ADDRESS WHICH
IS LABELED ADRPTR WILL BE THE TRAP PC ADDRESS ON THE STACK.

**SUBORDINATE ROUTINES CALLED:** NONE.

---

**CKTRAP:**

```
SAVE
CLR TP4FLG
ADRTR: TST TP4FLG
SEC
BEQ 601
CLC
INDICATE SUCCESS.
PASS
INDICATE FAILURE.
601: JSR PC,SP
RTS PC
```
GLOBAL SUBROUTINE  -  CLNRST

****************************************************************************
*:  CLEAN RESET OF THE DEVICE UNDER TEST
*:  THIS SUBROUTINE IS USED TO RESET THE DUT TO A KNOWN STATE.
*:  THE DUT'S SELF-TEST IS SKIPPED, AND THE FIFO IS PURGED OF ANY ERROR
*:  CODES, ETC.
*:  IF THE RESET DOES NOT SUCCESSFULLY COMPLETE, THEN THE CARRY BIT IS
*:  PASSED BACK TO THE CALLING ROUTINE (CLEAR).
*:  INPUTS:
*:  CSRA - CONTAINS THE ADDRESS OF THE CSR
*:  XBFCA - CONTAINS ADDRESS OF DUT DMA BUFFER COUNT REGISTER.
*:  ERRNBR - ERROR NUMBER FOR POSSIBLE ERROR REPORT.
*:  ERRBL, ERRTP, ERRNB, AND ERRMSG SET UP CORRECTLY.
*:  OUTPUTS:
*:  THE DUT PERFORMS ITS RESET FUNCTION INTO A KNOWN STATE.
*:  CARRY - CLEAR INDICATES THE TEST IS TO BE ABORTED.
*:  ERRBLK - VALUE MAY BE DESTROYED.
*:  IESTAT - TX AND RX INTERRUPT FLAGS ARE CLEARED.
*:  RX AND RX INTERRUPT ENABLE bits IN THE DUT's CSR ARE CLEARED.
*:  CALLING SEQUENCE:  JSR  PC,CLNRST
*:  COMMENTS:  THIS SUBROUTINE CAN REPORT ERRORS WITH NUMBERS ERRNBR.
*:  THIS ROUTINE DOES NOT DESTROY THE VALUE OF ERRNBR.
*:  SUBORDINATE ROUTINES CALLED:  DELAY, MSLGET, PUFIFO, RESETT.
*****************************************************************************

CLNRST:  SAVE
  JSR  RS5, PREG05  ; CALL REGISTER SAVE SUBRT.
  JSR  PC, RESETT  ; RESET THE DUT.
  JSR  PC, RESETT  ; THIS ROUTINE REPORTS ERRORS WITH NUMBERS FROM ERRNBR THRU ERRNBR+2.
  BCC  601  ; EXIT ROUTINE WITH ABORT TEST INDICATOR.
  JSR  PC, PUFIFO  ; PURGE THE FIFO OF ERROR CODES, SAVE ANY BMP CODES FOUND.
  JSR  PC, PREG5  ; RESTORE GPRS, PASS THE FOLLOWING INTACT:
  JSR  PC, BCS(5)  ; RETURN TO PREG05 SUBRT.
  RTS  PC

601:  PASS
  JSR  PC, BCS(5)  ; CARRY BIT: IF CLEAR, THEN ABORT THE TEST.
.SBTIL GLOBAL SUBROUTINE CLR16W
;*****************************************************************************
;*             CLEAR SIXTEEN WORDS ROUTINE
;*****************************************************************************
;* INPUTS:    RO - ADDRESS OF THE FIRST WORD TO CLEAR.
;* OUTPUTS:   (RO) TO (RO+15) - 16 WORDS OF MEMORY ARE CLEARED TO 0.
;* CALLING SEQUENCE:  JSR PC, CLR16W
;* COMMENTS:
;* SUBORDINATE ROUTINES CALLED:  NONE.
;*****************************************************************************

CLR16W:: SAVE JSR RS, PREGOS ;SAVE CONTENTS OF GPRS RO THRU RS.
    JSR RS, PREGOS ;CALL REGISTER SAVE SUBRT.
    MOV #16, R1 ;SET THE LOOP COUNTER TO 16.
    2I: CLR (RO) ;CLEAR A WORD OF MEMORY.
         DEC R1 ;COUNT THIS LOOP.
         BNE 2I ;LOOP IF NOT 16 WORD CLEARED.
    60I: PASS JSR PC, #(SP) ;RESTORE GPRS.
         RTS PC ;RETURN TO PREGOS SUBRT.
GLOBAL SUBROUTINE  CNTERR

- COUNT ERROR ROUTINE

THIS SUBROUTINE IS USED TO COUNT A "DATA" ERROR ON THE SPECIFIED LINE. IT CHECKS WHETHER ERROR SUMMARY REPORTING IS ACTIVE, OR SHOULDER BE MADE ACTIVE ON THIS LINE, AND ACTIVATES IT IF NEEDED.

INPUTS:
- R5: LINE NUMBER OF LINE UNDER CONSIDERATION.
- ERCTB: LABEL AT BASE OF ERROR COUNTERS TABLE.
- ERSMAF: ERROR SUMMARY FLAGS (BIT SET IF LINE IN SUMMARY MODE).
- NDERPT: NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE.

OUTPUTS:
- CARRY: SET IF LINE IS IN ERROR SUMMARY MODE.
- ERCHT: ERROR COUNTER INCREMENTED FOR SPECIFIED LINE.
- ERSMAF: BIT SET IF LINE SHOULD BE IN SUMMARY MODE.

CALLING SEQUENCE:
JSR  PC,CNTERR

COMMENTS:

SUBORDINATE ROUTINES CALLED:
NONE.

CNTERR:: SAVE
JSR  R5,PREG5
CALL REGISTER SAVE SUBRT.

COUNT THE ERROR ON THE COUNTER FOR THE SPECIFIED LINE.

ALR  R5
ASL
MOV  ERCTB(R5),R1
INC  R1
BCS  24
MOV  R1,ERCTB(R5)
TST  NDERPT
BEQ  601
CMP  R1,NDERPT
BMI  41
EQU  601
CLC
BR  601
BIT  41
CLC
BR  601
BITLBL(R5),ERSMAF
CMP  SECC
BNE  601
JSR  PC,0(BSP)
-outs

RETURN TO PREG5 SUBRT.
GLOBAL SUBROUTINE

**SUBTTL** GLOBAL SUBROUTINE DELAY

********************************************************************************************

**# THIS SUBROUTINE IS USED TO DELAY A VARIABLE NUMBER OF MILLI-SECONDS.**

**# INPUTS:** R4 CONTAINS THE NUMBER OF MS TO DELAY.

**# MSLCNT.**

**# OUTPUTS:** NONE.

**# CALLING SEQUENCE:** JSR PC,DELAY

**# COMMENTS:** IF NO HARDWARE CLOCK INTERRUPTS ARE OCCURING, CONTROL-C WILL

**# NOT BE HONORED FOR THE DURATION OF THE DELAY.**

**# SUBORDINATE ROUTINES CALLED: NONE.**

********************************************************************************************

DELAY: SAVE JSR R5,PREG05 ;SAVE CONTENTS OF GPRs RO THRU RS.

MOV R4,R1 ;SAVE NUMBER OF MS DELAY AS TIME-OUT VALUE.

MOV #1,R2 ;TELL MSLOOP ROUTINE TO CHECK ALL BITS.

CLR R3 ;TELL MSLOOP RTN TO CHECK FOR ALL BITS CLEAR.

MOV #62$ R4 ;TELL MSLOOP TO CHECK DUMMY NON ZERO WORD.

JSR PC,MSLOOP ;DELAY THE REQUESTED # OF MS.

BCC 60$ ;EXIT ROUTINE IF WE TIMED-OUT.

JSR PC,OOPS ;IF NO TIME-OUT, BAD PROGRAM OR HOST MACHINE.

60$: PASS ;RESTORE GPRs.

JSR PC,B(3P)+ ;RETURN TO PREG05 SUBRT.

RTS PC ;DUMMY, NON ZERO WORD.
GLOBAL SUBROUTINE MSLGET

******************************************************************************
*** THIS SUBROUTINE IS A GENERAL PURPOSE TEST LOOP SUBROUTINE. IT IS USED ***
*** TO VERIFY THAT A CERTAIN ACTION OCCURS BEFORE A TIME OUT PERIOD. THE ***
*** CALLING ROUTINE PASSES IN WHICH BITS SHOULD BE SET AND CLEARED FOR THE ***
*** TIME-OUT VALUE AND THE TIME-OUT VALUE IN MILLI SECONDS. ***
*** THIS ROUTINE CHECKS FOR THE DESIRED CONDITION UPON ENTRANCE INTO THE ***
*** ROUTINE AND THEN ONCE EACH MILLI-SECOND THEREAFTER. ***
*** UPON RETURN, THE LAST WORD WHICH WAS READ TO CHECK FOR THE CONDITION ***
*** IS RETURNED BY THIS SUBROUTINE.***
******************************************************************************

** INPUTS: **
** R1 - TIME-OUT VALUE IN MILLI-SECONDS (UP TO 64K MS). **
** R2 - BIT MAP OF BITS TO TEST (1 INDICATES TO TEST THE BIT). **
** R3 - DESIRED STATES OF THE INDICATED BITS IN R2. **
** R4 - ADDRESS OF THE WORD TO TEST. **
** MSCNT - MILLI SECOND SOFTWARE LOOP COUNT. **

** OUTPUTS: **
** R0 - THE LAST WORD WHICH WAS READ TO CHECK FOR THE CONDITION. **
** R1 - REMAINING NUMBER OF MS IN TIME-OUT TIME. **
** CARRY - SUCCESS FLAG (SET IF CONDITION IS MET BEFORE TIME OUT). **

** CALLING SEQUENCE: **
JSR PC,MSLGET

** COMMENTS: **
THIS ROUTINE WORKS WITH OR WITHOUT A HARDWARE CLOCK, BUT THE **
CALIBRATION IS ONLY GUARANTEED WHEN A LINE CLOCK IS AVAILABLE **
ON THE SYSTEM. **
THIS ROUTINE CAN BE USED AS A DELAY ROUTINE, BY SPECIFYING THE **
DESIRED DELAY AS THE TIME-OUT AND SPECIFYING A CONDITION TO **
LOOK FOR WHICH WILL NOT BE MET DURING THE DELAY. **
IF A TIME-OUT VALUE OF 0 IS SPECIFIED, THIS ROUTINE CHECKS FOR **
THE DESIRED CONDITION BEFORE RETURNING. IT INDICATES SUCCESS **
IF THE CONDITION IS MET, FAILURE OTHERWISE. **

** SUBORDINATE ROUTINES CALLED: **
NONE.

********************************************************************************

MSLGET:: SAVE
JR SRC, R5,PREGS ;CALL REGISTER SAVE SUBRT.

; SET UP MASK FOR REMOVING UNUSED BITS IN THE TEST WORD, AND CLEAR UNUSED
; BITS IN THE DESIRED STATE WORD TO ALLOW DIRECT COMPARISON.

COM R2
BIC R2,R3 ;MASK OUT UNUSED BITS IN DESIRED STATE WORD.

; HANDLE THE TEST AND EXIT IF WE HAVE A 0 TIME-OUT VALUE.

TST R1 ;TEST THE TIME-OUT VALUE FOR ZERO.
ONE 24 ;IF NON-ZERO TIME-OUT, GO LOOP AND TEST.
MDV (R4),R0 ;GET THE WORD TO TEST BEFORE EXITING.
MDV R0,62 ;SAVE VALUE SO WE CAN RETURN IT.
BIC R2,R0 ;MASK OUT UNTESTED BITS OF WORD.
CMP R0,R3 ;COMPARE AGAINST DESIRED STATE WORD.
SEC ;INDICATE SUCCESS IN CASE WORDS ARE EQUAL.
BEQ 64 ;EXIT WITH SUCCESS IF WORDS ARE EQUAL.
CLC                   ; INDICATE FAILURE (TIME OUT).
BR 6$                  ; EXIT WITH FAILURE, WORDS AREN'T EQUAL.

; NON ZERO TIME OUT VALUE. LOOP, WAITING FOR CONDITION OR TIME OUT.
2$: MOV  MSLCNT,R5     ; LOAD MS LOOP COUNT.
4$: MOV (R4),R0       ; GET THE WORD TO TEST.
MOV R0,62$             ; SAVE WORD IN CASE THIS IS THE LAST.
BIC R2,R0              ; MASK OUT UNTESTED BITS OF WORD.
CMP R0,R5              ; COMPARE AGAINST DESIRED STATE WORD.
SEC                   ; SET CARR) IN CASE OF SUCCESS.
BEQ 6$                 ; EXIT WITH SUCCESS IF WORDS ARE EQUAL.
DEC R5                 ; COUNT DOWN THE INSIDE MS LOOP COUNT.
BNE 4$                 ; LOOP IF MS NOT UP.
DEC R1                 ; DECREMENT THE MS TIME COUNT.
BNE 2$                 ; IF TIME NOT UP, LOOP TO COUNT ANOTHER MS.
CLC                   ; CLEAR CARRY, WE TIMED-OUT.

; HAVE EITHER FOUND CONDITION, OR TIMED OUT (POSSIBLY FROM 0 TIME OUT VALUE).
; RESTORE THE LAST CONTENTS READ FROM THE TESTWORD. EXIT ROUTINE.
6$: MOV 62$,R0         ; PASS OUT THE LAST READ WORD.
60$: PASS R0,R1       ; RESTORE GPAS, EXCEPT THE FOLLOWING:
MOV R0,ROSLOT(SP)     ; PUT RO IN STACK SLOT.
MOV R1,RISLOT(SP)     ; PUT R1 IN STACK SLOT.
JSR PC,(SP)           ; RETURN TO PREGOS SUBR.
JPCR LAST READ WORD CHECKED FOR CONDITION.
R1 REMAINING TIME (0 IF TIME OUT OCCURED).
RTS PC                ; CARRY SET IF SUCCESS, CLEAR IF TIME-OUT.

; LOCAL STORAGE.
62$: .WORD 0          ; STORAGE FOR THE LAST READ WORD.
.SBITL GLOBAL SUBROUTINE
MSLOOP

1

**TEST LOOP SUBROUTINE**

1

**THIS SUBROUTINE IS A GENERAL PURPOSE TEST LOOP SUBROUTINE. IT IS USED**

1

**TO VERIFY THAT A CERTAIN ACTION OCCURS BEFORE A TIME OUT PERIOD, THE**

1

**CALLING ROUTINE PASSES IN WHICH BITS SHOULD BE SET AND CLEAR FOR THE**

1

**DESIRED CONDITION AND THE TIME OUT VALUE IN MILLI-SECONDS.**

1

**THIS ROUTINE CHECKS FOR THE DESIRED CONDITION UPON ENTRANCE INTO THE**

1

**ROUTINE AND THEN ONCE EACH MILLI SECOND THEREAFTER.**

1

**INPUTS:**

1

**R1 - TIME-OUT VALUE IN MILLI SECONDS (UP TO 64K MS).**

1

**R2 - BIT MAP OF BITS TO TEST (1 INDICATES TO TEST THE BIT).**

1

**R3 - DESIRED STATES OF THE INDICATED FIELDS IN R2.**

1

**R4 - ADDRESS OF THE WORD TO TEST.**

1

**MSLONG - MILLI SECOND SOFTWARE LOOP COUNT.**

1

**OUTPUTS:**

1

**CARRY SUCCESS FLAG (SET IF CONDITION IS MET BEFORE TIME OUT).**

1

**CALLING SEQUENCE:**

1

**JSR PC, MSLOOP**

1

**COMMENTS:**

1

**THIS ROUTINE WORKS WITH OR WITHOUT A HARDWARE CLOCK, BUT THE**

1

**CALIBRATION IS ONLY GUARANTEED WHEN A LINE CLOCK IS AVAILABLE**

1

**ON THE SYSTEM.**

1

**THIS ROUTINE CAN BE USED AS A DELAY ROUTINE, BY SPECIFYING THE**

1

**DESIRED DELAY AS THE TIME-OUT AND SPECIFYING A CONDITION TO**

1

**LOOK FOR WHICH WILL NOT BE MET DURING THE DELAY.**

1

**IF A TIME OUT VALUE OF 0 IS SPECIFIED, THIS ROUTINE CHECKS FOR**

1

**THE DESIRED CONDITION BEFORE RETURNING, IT INDICATES SUCCESS**

1

**IF THE CONDITION IS MET, FAILURE OTHERWISE.**

1

**SUBORDINATE ROUTINES CALLED:**

1

**MSLONG.**

1

**;MSLOOP:**

1

**;SAVE CONTENTS OF GP3S RS THRU RS.**

1

**;CALL REGISTER SAVE SUBRT.**

1

**;CALLING THE MSLGET ROUTINE FROM THE MSLOOP ROUTINE ISOLATES THE CALLER OF**

1

**MSLOOP FROM THE RETURNED TEST WORD AND REMAINING TIME OUT VALUES.**

1

**;JSR PC, MSLGET ;CALL THE MULTI PURPOSE MS LOOP AND SEARCH RTN.**

1

**;PASS**

1

**;RESTORE GP3S.**

1

**;RETURN TO PREGOS SUBRT.**

1

**RTS PC ;CARRY SET IF SUCCESS, CLEAR IF TIME OUT.**
EMO102:: .ASCIZ "PROGRAM HUNG, WAITING FOR A CONTROL C. "

.EVEN
SUBROUTINE PUFIFO

- PURGE THE FIFO
- THIS ROUTINE TRIES TO REMOVE ALL THE CHARACTERS FROM THE FIFO.
- ANY BMP CODES THAT ARE FOUND ARE SAVED ON THE BMP CODE QUEUE.
- INPUTS: RBuFA CONTAINS THE ADDRESS OF THE RECEIVER.
- OUTPUTS: CARRY BIT - INDICATES THE STATE OF THE FIFO, SET: PURGED.
- BMPco - THE CONTENTS OF THE BMP CODE QUEUE MAY BE UPDATED.
- CALLING SEQUENCE: JSR PC, PUFIFO
- COMMENTS:

SUBORDINATE ROUTINES CALLED: SAVBMP.

PUFIFO:: SAVE

MOV @512, R1
MOV RBuFA, R4
BPL 6:

MOVE @70000, R0
BIC R2, R0
BNE 4:

MO'.
GLOBAL SUBROUTINE

* ***************
* READ AND VERIFY DATA PATTERN FROM DEVICE REGISTERS ROUTINE
* THIS ROUTINE READS AND VERIFIES THE ROTATED DATA PATTERN WHICH HAS
* BEEN WRITTEN BY THE WDPR SUBROUTINE.
* EACH ACTIVE LINE'S REGISTER'S CONTENTS IS READ AND COMPARED WITH THE
* WRITTEN DATA.
* AFTER THE UNUSED AND READ ONLY (RO) BITS ARE MASKED OUT, ANY ERRORS ARE
* REPORTED FROM THIS ROUTINE.
* THIS ROUTINE WILL TAKE INTO ACCOUNT THE TYPE OF WRITE OPERATION WHICH
* WAS PERFORMED BY THE WDPR SUBROUTINE.
*
* INPUTS:
* R2 - USED TO PASS IN THE DATA PATTERN TO BE ROTATED & VERIFIED.
* R3 - BYTE INDICATOR (- = LO BYTE, + = HI BYTE, 0 = BOTH).
* R4 - OPERATION TYPE INDICATOR (- = BIC, + = BIS, 0 = MOV).
* ACTLNS - BIT MAP OF ACTIVE LINES ON THE DEVICE UNDER TEST.
* CSRA - CONTAINS THE CSR ADDRESS OF THE DEVICE UNDER TEST.
* DRADRT - BASE ADDRESS OF DEVICE REGISTER ADDRESS TABLE.
* ERCNTRB - LABEL AT BASE OF ERROR COUNTERS TABLE FOR LINES.
* ERRMNG - SET UP WITH THE PROPER ERROR MESSAGE FOR THIS TEST.
* ERRNBR - SET UP WITH THE PROPER ERROR NUMBER.
* LPRR - EQUATED TO LPR REG OFFSET FROM DEVICE CSR ADDRESS.
* NUMLNS - NUMBER OF LINES ON THE DEVICE UNDER TEST.
* INDERP - NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE.
* TXBFTC - EQUATED TO TBUFF IC REG OFFSET FROM DEVICE CSR ADDRESS.
* UNBSTB - BASE ADDRESS OF THE UNUSED BIT TABLE.
*
* OUTPUTS:
* ERROR MESSAGES MAY BE PRINTED AT THE OPERATOR'S CONSOLE.
* ERCNTR - ERROR COUNTERS TABLE IS UPDATED FOR LINE UNDER TEST.
* ERSRBLK - CONTENTS DESTROYED.
* ERSNRMF - ERROR SUMMARY FLAGS BIT SET IF LINE IN SUMMARY MODE.
* UUT CSR - ALL BITS CLEARED, EXCEPT IND.ADR.REG FIELD DESTROYED.
*
* CALLING SEQUENCE: JSR PC,ROPR
* COM: FOR BYTE ACCESSES, ONLY THE SPECIFIED BYTE IS VERIFIED.
* SUBORDINATE ROUTINES CALLED: ER1601, ROA.
* -------------------------------------------------------------

ROPR: SAVE ;SAVE CONTENTS OF GPRS RO THRU RS.
        JSR #ER1601, R5, PREGS ;CALL REGISTER SAVE SUBRT.
        MOV R5, R5, PREGS ;SET UP THE ADDRESS OF THE ERROR REPORT RTN.

; DETERMINE WHETHER REGISTER DATA SHOULD BE INVERTED FROM DATA PATTERN.
; TST R4
; BPL 21 ;CHECK THE OPERAND TYPE INDICATOR.
; BIC WRITE PERFORMED? NO, USE STANDARD DATA.
        COM R2 ;YES, INVERT THE DATA PATTERN.

; SET UP OUTER LOOP.
21:  CLR R5 ;CLEAR LINE COUNTER TO SELECT LINE 0.

; THE OUTER LOOP FOLLOWS. EACH PASS THROUGH THIS LOOP READS AND COMPARES DATA
; FROM ALL OF THE DEVICE REGISTERS FOR A PARTICULAR LINE IF THE LINE IS ACTIVE.
2989 020646 010267 000222 41: MOV R2, R701 ;SAVE THE OUTER LOOP DATA PATTERN.
2990 020655 010577 161364 MOV R5, WCSR ;SET CSR IND.ADR.REG FIELD TO THIS LINE.
2991 020656 010500 R5, RO
2992 020660 006300 ASL RO
2993 020662 036061 002410 161340 BIT BIITBL(R0), ACLKNS
2994 020670 001467 BEQ 164 ;IS THE LINE ACTIVE? NO, SKIP THE LINE.
2995 020672 012703 000004 MOV @PRO, R3 ;YES, INITIALIZE REGISTER OFFSET FOR LPR.
2996 1: THE INNER LOOP FOLLOWS. EACH PASS THROUGH THIS LOOP READS AND COMPARES
2997 DATA FROM A DEVICE REGISTER.
2998 1:
2999 0000 020676 010204 64: MOV R2, R4 ;SAVE THE INNER LOOP DATA PATTERN.
3000 020700 046302 002370 BIC UNBBTBC(R3), R2 ;REMOVE UNUSED BITS FROM EXPECTED DATA.
3001 020704 016300 002242 MOV DRADTR(R3), R0
3002 020710 005766 000010 TST R5SLOT(SP)
3003 020714 001002 BNE R8 ;BYTE ACCESS? YES, GO PERFORM BYTE READ.
3004 020716 010011 MOV (R0), R1 ;NO, PERFORM WRED READ OF DEVICE REGISTER.
3005 020720 000416 BR 124
3006 020722 100410 BMI 103 ;LOW BYTE ACCESS? YES, GO PERFORM LOW BYTE READ.
3007 020724 005200 INC RO ;HIGH BYTE ADDRESS. FORM HIGH BYTE ADDRESS.
3008 020726 111001 MOVVR (R0), R1 ;READ THE HI BYTE OF THE OUT REGISTER.
3009 020730 000301 SWAB R1 ;PUT HI BYTE BACK INTO THE HI BYTE.
3010 020732 042701 000377 BIC #377, R2 ;REMOVE THE UNUSED BYTE IN ACTUAL DATA.
3011 020736 042702 000377 BIC #377, R2 ;REMOVE THE UNUSED BYTE IN EXPECTED DATA.
3012 020742 000405 BR 124
3013 020744 111001 MOVVR (R0), R1 ;READ THE LOW BYTE OF THE OUT REGISTER.
3014 020746 042701 177400 BIC #177400, R1 ;REMOVE THE UNUSED BYTE.
3015 020752 042702 177400 BIC #177400, R2 ;FORM EXPECTED LOW BYTE FOR COMPARISON.
3016 020756 046301 002370 124: BIC UNBBTBC(R3), R1 ;REMOVE UNUSED BITS FROM ACTUAL DATA.
3017 020762 020102 CMP R1, R2 ;COMPARE ACTUAL AND EXPECTED DATA.
3018 020764 001414 BEQ 141 ;ACTUAL EXPECTED? YES, SKIP ERROR.
3019 020766 004767 177054 JSR PC, CNTERR ;NO, COUNT THE ERROR, CHECK FOR ERROR SUMMARY.
3020 020772 103411 BCS 141 ;USE ERROR SUMMARY? YES, SKIP ERROR.
3021 020774 001004 1: NO, REPORT "BAD BIT(S) IN DEVICE XXXXX REGISTER FOR LINE NN (D)."
3022 020777 104460 ERROR
3023 020779 032767 000100 161220 ;EXIT THIS ROUTINE AND SET THE "EXIT ON ERROR" FLAG, IF EXTENDED ERROR
3024 REPORTING HAS NOT BEEN REQUESTED.
3025 02077A 032767 000100 161220 1: BIT #8106, OPTION ;HAS EXTENDED ERROR REPORTING BEEN REQUESTED?
3026 020781 001004 BNE 144 ;BRANCH IF IT HAS.
3027 020782 012767 000001 161270 MOV #1, EXE0RR ;SET THE EXIT ON ERROR FLAG.
3028 020784 000425 BR 60 ;EXIT THE ROUTINE.
3029 020786 032767 000100 161220 14: MOV R4, R2 ;RESTORE THE INNER LOOP DATA PATTERN.
3030 020788 032767 000100 161220 14: MOV R4, R2 ;RESTORE THE INNER LOOP DATA PATTERN.
3031 02078A 012767 000015 JSR PC, ROLDAP ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
3032 02078C 062703 000002 ADD #2, R3 ;ADD REGISTER OFFSET TO THE NEXT REGISTER.
3033 02078E 062703 000002 JMPR R3, FSL50 ;CHECK THAT THIS IS NOT THE FIFO SIZE DATA REG.
3034 020790 010002 BNE 151 ;AVOID ALTERING THE OFFSET IF IT ISN'T.
3035 020792 062703 000002 ADD #2, R3 ;POINT AT THE NEXT REGISTER.
3036 020794 012703 000002 151: JSR R3, @XBFDCO ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
3037 020796 003713 BR 61 ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
3038 020798 010402 1: BACK INTO THE OUTER LOOP, NOW SET UP FOR NEXT LINE. LOOP IF NOT DONE.
161: MOV 70!, R2
    JSR PC, RLDAP
    ; SET UP TO ROTATE THE DATA PATTERN.
    ; ROTATE THE DATA PATTERN.

3047 021060 005205
3048 021062 020527 000020
3049 021066 002667
    CMP R5, #NUMLNS
    ; COMPARE LINE COUNT WITH NUMBER OF LINES.
701: .WORD 0
    BLT 41
    ; LOOP IF SOME LINES NOT DONE.
601: PASS
    JSR PC, 2(SP)
    ; RESTORE GPRs.
    ; RETURN TO PREGOS SUBRT.
5050 021070 004736
5052 021072 000207
    RTS PC
5053 021074 000000
GLOBAL SUBROUTINE

; SUBROUTINE TO TEST THE DEVICE UNDER TEST (OUT) REGISTERS. THE USED
; BITS OF THE REGISTERS ARE EITHER ALL CLEARED OR ALL SET AND THEN THE
; DATA PATTERN IS WRITTEN AND VERIFIED USING EITHER WORD OR BYTE
; ACCESSES IN READ/WRITE OR READ/modify/WRITE MODE.

; INPUTS: R3 BYTE INDICATOR (- = LOW, - = HIGH, 0 = BOTH BYTES).
; R4 ACCESS MODE ( 1 = SET THEN BIC, 1 = CLEAR THEN BIS).
; (-2 =) SET THEM MOV, +2 CLEAR THEN MOV).
; ERRNO = SET UP WITH INITIAL ERROR NUMBER.
; OUTPUTS: GPRSO = GPR SAVE AREA 0 IS DESTROYED.
; DEVICE UNDER TEST REGISTERS ARE WRITTEN.
; ERROR MESSAGES MAY BE PRINTED AT THE OPERATORS CONSOLE.

; CALLING SEQUENCE: JSR PC,REGST

; COMMENTS: THIS ROUTINE LOOP 16 TIMES WRITING THE SAME DATA PATTERN
; ROTATED LEFT ONCE EACH ITERATION.
; THIS ROUTINE CAN REPORT ERRORS INITIAL ERROR THRU INITIAL + 2.

; SUBORDINATE ROUTINES CALLED: RDPDR, ROLPAP, SWAPO, WDPO9

; ----------------------------------------
REGSTI:: SAVE JSR $R5, PREGOS i CALL REGISTER SAVE SUBRT.
; SET UP THE GPR FOR THE WRITING OF THE DATA PATTERN.
; MDV @@6, R5 ; SET UP LOOP COUNTER TO COUNT 16 ITERATIONS.
; MDV @167410, R2 ; INITIALIZ THE DATA PATTERN.
; BIT 0, R4 ; TEST FOR R/W ACCESS.
; BNE 21 ; RW ACCESS? YES, R4 IS ALL SET UP.
; CLR R4 ; NO, INDICATE R/W ACCESS.

21:
; SET UP THE GPR FOR THE CLEARING OR SETTING OF ALL THE USED BITS.
; MDV R4, R0 ; PASS OPERATION TYPE INDICATOR AROUND SWAPO.
; JSR PC, SWAPO ; GET ALTERNATE GPR SET IN R1 THRU R5.
; MDV ERRNO, R1 ; SAVE THE INITIAL ERROR NUMBER.
; MDV R0, R4 ; SET UP OP TYPE FOR CLEARING OR SETTING.
; CMP RASLOT(0), 0 ; TEST FOR CLEAR THEN MOV TEST SEQUENCE.
; BEQ 41 ; CLEAR THEN MOV? YES, LEAVE WRITE PAT CLEAR.
; CLR R4 ; NO, SET ALL BITS OF WRITE PATTERN.
; CMP RASLOT(0), 2 ; TEST FOR SET THEN MOV TEST SEQUENCE.
; BNE 61 ; SET THEN MOV? YES, LEAVE AL BY GAR PAT CLEAR.
; CLR R0 ; NO, SET AL BY GAR EXPECTED DAT P AT TO ALL 1 S.
; JSR PC, SWAPO ; RESTORE SWAPPED GPR VALUES TO R1 THRU R5.

; START OF DATA PATTERN LOOP.

01044 021270 005100
01048 021170 010001
01052 021166 001000
01056 021160 000012 177776
01060 021156 005000
01064 021152 005102
01068 021150 001401
01072 021142 026627 000012 000002
01076 021140 005002
01080 021134 005003
01084 021130 016701 162634
01088 021124 000767 002172
01092 021122 010400
01096 021120 004567 162674
01100 021076
01104 021076 000020
01108 021102 012705 000020
01112 021106 012702 167410
; SET OR CLEAR ALL THE USED BITS OF THE DEVICE REGISTERS FOR ALL LINES.
; VERIFY THAT ALL THE BITS WERE SET OR CLEARED CORRECTLY.
; GET ALTERNATE GPRS FOR SETTING INITIAL STATES.
; GO CLEAR ALL USED REGISTER BITS, ALL LINES.
; SET UP ERROR NUMBER TO INITIAL ERRNBR.
; VERIFY ALL USED REGISTER BITS, ALL LINES.
; EXIT THIS ROUTINE IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING
; HAS BEEN REQUESTED, I.E. EXDERR IS NON ZERO.
; TST EXOERR ; HAS AN ERROR BEEN FOUND ?
; BNE 60$ ; EXIT THIS ROUTINE IF IT HAS.
; INC ERRNBR ; SET ERROR NUMBER TO INITIAL-1.
; JSR PC,SWAPD ; WRITE DATA PATTERN TO DEVICE REGISTERS.
; JSR PC,RPDOR ; VERIFY DATA PATTERN IN ALTERRED BYTE(S).
; EXIT THIS ROUTINE IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING
; HAS BEEN REQUESTED.
; TST EXOERR ; HAS AN ERROR BEEN FOUND ?
; BNE 60$ ; EXIT THIS ROUTINE IF IT HAS.
; BEQ 10$ ; CHECK THE BYTE INDICATOR.
; MOV R2,R1 ; SAVE THE DATA PATTERN.
; MOV R0,R2 ; GET THE ALTERNATE BYTE EXPECTED DATA.
; NEG R3 ; INDICATE THAT OTHER BYTE IS TO BE CHECKED.
; INC ERRNBR ; SET ERROR NUMBER TO INITIAL-2.
; JSR PC,RPDOR ; VERIFY DATA PATS IN OTHER BYTES OF REGISTERS.
; EXIT THIS ROUTINE IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING
; HAS BEEN REQUESTED.
; TST EXOERR ; HAS AN ERROR BEEN FOUND ?
; BNE 60$ ; EXIT THIS ROUTINE IF IT HAS.
; MOV R1,R2 ; RESTORE BYTE INDICATOR.
; MOV R1,R2 ; RESTORE DATA PATTERN.
; PREPARE THE NEXT DATA PATTERN AND LOOP IF NOT DONE.
; JSR PC,RLODAP ; ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
; DEC R5 ; COUNT THIS ITERATION OF THE LOOP.
; BGT 8$ ; ALL PATTERNS DONE? NO. LOOP.
; YES, RESTORE ERROR NUMBER AND EXIT.
; GET THE ERROR NUMBER FROM GPR SWAP STORAGE.
; RESTORE GPRS.
; RETURN TO PREGOS SUBRT.

3169  MOV  GPRSOB,ERANBR
     021316  016767  161126  162444  604:
3170  PASS
     021324
3171  JSR    PC,(SP)+
     004736
3172  RTS    PC
     000207
**GLOBAL SUBROUTINE**

```assembly
.SBITL GLOBAL SUBROUTINE REPSMR
.REPSMR
;;; Report Error Summary Routine

THIS SUBROUTINE REPORTS AN ERROR SUMMARY FOR THOSE LINES WHICH HAVE EXCEEDED THE NUMBER OF INDIVIDUAL ERRORS TO REPORT FOR A SINGLE LINE IN A SINGLE TEST. THIS PARAMETER CAN BE SPECIFIED BY THE OPERATOR IF HE/SHE ANSWERS THE SOFTWARE PARAMETER QUESTIONS.

INPUTS:
+ ERNCNTB - LABEL AT BASE OF LINE ERROR COUNTERS TABLE.
+ ERRMSG - ADDRESS OF PRIMARY ERROR MESSAGE.
+ ERRNBR - ERROR NUMBER OF ERRORS IN THIS ROUTINE.
+ ERRSMR - "REPORT ERROR SUMMARY FOR LINE FLAGS.

OUTPUTS:
+ ERRBLK - ADDRESS OF ERROR REPORTING ROUTINE (DESTROYED).
+ SUMMARY MESSAGES MAY BE PRINTED AT THE OPERATOR CONSOLE.

CALLING SEQUENCE:
+ JSR PC, REPSMR

COMMENTS:
+ IF NO LINES HAVE EXCEEDED THE MAXIMUM NUMBER OF INDIVIDUAL ERRORS TO REPORT, NO MESSAGES ARE PRINTED BY THIS ROUTINE.
+ ERROR SUMMARIES IN THIS ROUTINE ARE REPORTED AS ERRORS.
+ THE CONTENTS OF ERRBLK ARE DESTROYED.

SUBORDINATE ROUTINES CALLED:

REPSMR:
+ SAVE JSR R5, PREGOS ; SCURE CONTENTS OF GPRS RO THRU R5.
+ TST ERMAR ; CHECK REGISTER SAVE SUBRT.
+ BEQ 601 ; EXIT WITHOUT ACTION IF NO SUMMARY FLAGS SET.
+ ; WE HAVE SOME ERROR SUMMARIES TO REPORT.
+ MOV #ER9004, ERRBLK ; SELECT ERROR REPORTING ROUTINE.
+ ; REPORT "ERROR SUMMARY REPORT FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:";
+ ; ERROR
+ TRAP C\ERROR
+ ; RESTORE GPRS.
+ 601: PASS JSR PC, (PRP).
+ RTS PC ; RETURN TO PREGOS SUBRT.
```

**Global Subroutine**

```assembly
SBTIL GLOBAL SUBROUTINE REPSMR
.REPSMR
;; Report Error Summary Routine

THIS SUBROUTINE REPORTS AN ERROR SUMMARY FOR THOSE LINES WHICH HAVE EXCEEDED THE NUMBER OF INDIVIDUAL ERRORS TO REPORT FOR A SINGLE LINE IN A SINGLE TEST. THIS PARAMETER CAN BE SPECIFIED BY THE OPERATOR IF HE/SHE ANSWERS THE SOFTWARE PARAMETER QUESTIONS.

INPUTS:
+ ERNCNTB - LABEL AT BASE OF LINE ERROR COUNTERS TABLE.
+ ERRMSG - ADDRESS OF PRIMARY ERROR MESSAGE.
+ ERRNBR - ERROR NUMBER OF ERRORS IN THIS ROUTINE.
+ ERRSMR - "REPORT ERROR SUMMARY FOR LINE FLAGS.

OUTPUTS:
+ ERRBLK - ADDRESS OF ERROR REPORTING ROUTINE (DESTROYED).
+ SUMMARY MESSAGES MAY BE PRINTED AT THE OPERATOR CONSOLE.

CALLING SEQUENCE:
+ JSR PC, REPSMR

COMMENTS:
+ IF NO LINES HAVE EXCEEDED THE MAXIMUM NUMBER OF INDIVIDUAL ERRORS TO REPORT, NO MESSAGES ARE PRINTED BY THIS ROUTINE.
+ ERROR SUMMARIES IN THIS ROUTINE ARE REPORTED AS ERRORS.
+ THE CONTENTS OF ERRBLK ARE DESTROYED.

SUBORDINATE ROUTINES CALLED:

REPSMR:
+ SAVE JSR R5, PREGOS ; SCURE CONTENTS OF GPRS RO THRU R5.
+ TST ERMAR ; CHECK REGISTER SAVE SUBRT.
+ BEQ 601 ; EXIT WITHOUT ACTION IF NO SUMMARY FLAGS SET.
+ ; WE HAVE SOME ERROR SUMMARIES TO REPORT.
+ MOV #ER9004, ERRBLK ; SELECT ERROR REPORTING ROUTINE.
+ ; REPORT "ERROR SUMMARY REPORT FOR LINES WITH EXCESSIVE NUMBERS OF ERRORS:";
+ ; ERROR
+ TRAP C\ERROR
+ ; RESTORE GPRS.
+ 601: PASS JSR PC, (PRP).
+ RTS PC ; RETURN TO PREGOS SUBRT.
```
GLOBAL SUBROUTINE RESET

**INPUTS:**
- CSRA - CONTAINS THE ADDRESS OF THE CSR
- TXBFCA - CONTAINS ADDRESS OF DUT DMA BUFFER COUNT REGISTER.
- ERRBL, ERRTP, ERRNB, AND ERRMSG SET UP CORRECTLY.

**OUTPUTS:**
- THE DUT PERFORMS ITS RESET FUNCTION INTO A KNOWN STATE.
- CARRY - CLEAR INDICATES THE TEST IS TO BE ABORTED.
- ERRBLK - VALUE MAY BE DESTROYED.
- IESTAT - TX AND RX INTERRUPT FLAGS ARE CLEARED.
- TX AND RX INTERRUPT ENABLE BITS IN THE DUT'S CSR ARE CLEARED.

**CALLING SEQUENCE:**
- JSR PC, RESET

**COMMENTS:**
- THIS SUBROUTINE CAN REPORT ERRORS WITH NUMBERS INITIAL ERNR.
- THIS ROUTINE DOES NOT DESTROY THE VALUE OF ERNR.

**SUBORDINATE ROUTINES CALLED:**
- DELAY, MSLGET.

---

RESET: SAVE

JSR RC, R5, PREGS
CALL REGISTER SAVE SUBRT.

SET BIT MASK OF MASTER RESET BIT.

TEST THE STATE OF THE MASTER RESET BIT IN THE CSR.
IF MR IS SET THEN WAIT FOR SELF TEST TO COMPLETE.
IF TIME-OUT OCCURS, REPORT THE ERROR AND PASS OUT ABORT TEST INDICATOR.

MOV CSRA, R4
GET THE ADDRESS OF THE DUT'S CSR.
BIT R2, (R4)
CHECK STATE OF MASTER RESET BIT.
BEQ 21
DON'T DELAY IF MR IS ALREADY CLEAR.
CLR R5
SET UP DESIRED STATE OF MASTER RESET BIT.
MOV $5000, R1
PASS TIME-OUT VALUE OF 5 SECONDS.
JSR PC, MSLGET
WAIT FOR SELF-TEST TO COMPLETE, MR CLEAR.
BCC 41
GO REPORT ERROR IF TIMEOUT OCCURRED.

SET MASTER RESET BIT IN CSR. CLEAR TX AND RX ENABLE BITS, ETC.
SKIP THE SELFTEST.
TIME-OUT OF 5 SECS, JUST IN CASE THE SELF-TEST EXECUTES.

21:
MOV R2, BCSRA
SET MASTER RESET BIT, DISABLE TX AND RX INTS.
JSR PC, SKPSTS
TRY TO SKIP THE SELFTEST.

SET SELF TEST TIME-OUT OF 5 SECONDS, AND WAIT FOR MR TO CLEAR.
IF TIME-OUT OCCURS, THEN REPORT THE FATAL ERROR AND PASS-OUT THE ABORT TEST INDICATOR.

CLR R3
SET UP DESIRED STATE OF MASTER RESET BIT.
MOV $5000, R1
PASS TIME-OUT VALUE OF 5 SECONDS.
JSR PC, MSLGET
WAIT FOR SELF TEST TO COMPLETE, MR CLEAR.
BCS 41
SKIP ERROR REPORT IF MR CLEARED IN TIME.
; 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$: MOV @EM1601.R1 ; PASS ERROR MESSAGE TO REPORT.
3275 MOV @ER1603,ERABLK ; PASS ADDRESS OF ERROR HANDLING ROUTINE.
3276 REPORT ERROR "TIME OUT OCCURRED WAITING FOR MASTER RESET TO CLEAR"
3277 "TEST ABORTED"
3278 ERROR:
3279 >>>>> ERROR <<<<<
3280 BR 60$ ; EXIT THIS SUBROUTINE, ABORT TEST INDICATOR.
3281 CLC ; CLEARarsing ERROR.
3282 BR 60$ ; EXIT THIS SUBROUTINE, ABORT TEST INDICATOR.
3283 ; CLEAR FX AND RX INTERRUPT ENABLE STATUS FLAGS IN ISTAT.
3284 ; EXIT WITH CONTINUE TEST INDICATOR SET (IE,CARRY SET).
3285
3286 6$: CLR ISTAT ; CLEAR FX AND RX INTERRUPT STATUS FLAGS.
3287 SEC ; INDICATE SUCCESS, CONTINUE TEST.
3288
3289 60$: JSR PC,0(SP) ; RESTORE GPRs, PASS THE FOLLOWING INACT.
3290 ; RETURN TO PREG05 SUBRT.
3291 ; CARRY BIT: IF CLEAR, INDICATES ABORT TEST.
3292 RTS PC
3293
**GLOBAL SUBROUTINE**

*ROLDAP*

---

*ROLDAP*:

`ROLDAP: SAVE JSR R5,PREGOS ;CALL REGISTER SAVE SUBRT.
TST R2 ;SAVE CONTENTS OF GPRS RO THRU R5.
BPL 2$ ;CHECK MSB, AND CLEAR CARRY.
SEC ;SET CARRY IF MSB SET
ROL R2 ;ROTATE DATA PATTERN LEFT
2$: ;RESTORE GPRS, EXCEPT
60$: PASS R2
MOV R2,R2; SLOT(SP) ;PUT R2 IN STACK SLOT.
JSR PC,(SP) ;RETURN TO PREGOS SUBRT.
R2 : CONTAINS THE ROTATED DATA PATTERN

 COMMENTS:

CALLING SEQUENCE: JSR PC, ROLDAP

OUTPUS: R2 CONTAINS THE ROTATED DATA PATTERN

INPUTS: R2 CONTAINS THE DATA PATTERN TO BE ROTATED

SUBORDINATE ROUTINES CALLED: NONE

---

*SBITL*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:

---

*ROLDAP*:
SBTIL GLOBAL SUBROUTINE ASTRPT

This routine determines if any error codes are among the diagnostic codes reported placed in the out received character FIFO by the self test. If any non BMP error codes are found, or if other errors are encountered, appropriate errors are reported. Any BMP codes that are found, are placed on the BMP code queue to be reported later. This routine also purges the DUT FIFO looking for any characters or modem status codes. If any are found, errors are reported.

Inputs: ERMMSG address of the primary error message.
ERAME error number of first error reported by this routine.
NUMLN a number equated to the number of line on the DUT.
RBUF contains address of the DUT receiver FIFO.

Outputs: CARRY success flag (set if FIFO cleared successfully).
ERRBLK address of the error report routine (destroyed).
ERROR MESSAGE be printed at the operators console.

Calling sequence: JSR PC ASTRPT

Comments: This subroutine can report errors with numbers initial ERRNBR 1 thru initial ERRNBR 4.
This routine does not destroy the value of ERRNBR.

Subordinate routines called: ER0503, ER9007, ER9008, SAVBMP.

RSTRPT: ; SAVE ; SAVE CONTENTS OF GPRS RO THRU RS.
JSR R5, PREG05 ; CALL REGISTER SAVE SUBRT.

; READ CORRECT NUMBER (NUMBER OF LINE ODN DUT) OF CHAR FROM THE FIFO.
; VERIFY THAT EACH CHAR IS A SELFTEST SUCCESS CODE.
CLR R3 ; CLEAR THE CODE COUNTER.
MOV ERRNBR, R5 ; SAVE ERRNBR FOR RESTORATION LATER.

; READ A CHAR FROM THE DUT FIFO.
BMI 41 ; SKIP ERROR IF DATA.VALID SET FOR CHAR.

; WE EXPECT A SELFTEST CODE, BUT THIS FIFO SLOT IS EMPTY.
MOV R5, ERRNBR ; RESTORE ERROR NUMBER TO INITIAL VALUE.
MOV #ER9018, R1 ; PASS ERROR MESSAGE INFO TO ER9007 ROUTINE.
MOV #ER9007, ERRBLK ; SELECT PROPER ERROR REPORT ROUTINE.

; REPORT ERROR WITH NUMBER INITIAL ERRNBR.
; 'NO SELFTEST CODE IN SELFTEST CODE FIFO SLOT FOR LINE NN AFTER RESET.'
; ERROR ; TRAP CERROR

; EXIT THIS ROUTINE IF EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED.
; EXIT THIS ROUTINE IF EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED.
BNE 31 ; AVOID SET THE FLAG IF IT HAS.
MOV #1, EXOERF ; SET THE EXIT ON ERROR FLAG.
INSTRUCT 'SUCCESS' (BECAUSE FIFO IS PURGED), AND EXIT THIS ROUTINE.

JMP 60h ;EXIT ROUTINE.

Determine IF THIS IS NOT A SELFTEST CODE.

MOV #70001h, R0 ;GENERATE BIT MAP OF ANY CLEAR ERROR BITS OR
BIC R2, R0 ;BIT 0 WHICH ARE CLEAR.
BNE 8h ;GO TO REPORT ERROR IF THIS IS NOT A TEST CODE.

WE HAVE A TEST CODE (EITHER BMP OR SELFTEST CODE).

Determine WHAT TYPE OF CODE WE HAVE.

BIT #8171, R2 ;TEST ROM VERSION CODE INDICATOR BIT.
BEQ 10h ;SKIP ERRORS IF SELFTEST ROM VERSION CODE.
CMPB R2, 0203h ;CHECK IF SKIP SELF TEST CODE.
BEQ 10h ;SKIP ERROR REPORT IF SKIP SELF TEST CODE FOUND
CMPB R2, 0201h ;CHECK IF NULL CODE PRESENT.
BEQ 10h ;SKIP ERROR REPORT IF SELF TEST NULL CODE.
MOV #300h, R0 ;TEST CODE TYPE BITS FOR BOTH CODE
BIC R2, R0 ;TYPE BITS SET (INDICATING BMP CODE).
BNE 6h ;IF IT IS NOT A BMP CODE GO REPORT ERROR.
JSR PC, SAVBMP ;SAVE THE BMP CODE ON THE QUEUE.
BR 10h ;GO GET THE NEXT CHARACTER FROM THE FIFO.

WE HAVE A SELFTEST ERROR CODE.

MOV R5, ERRNR ;RESTORE ERROR NUMBER TO INITIAL VALUE.
INC ERRNR ;CALCULATE INITIAL ERROR NUMBER PLUS 1.
MOV R2, 0200h, R1 ;PASS ERROR MESSAGE INFO TO ER9000 ROUTINE.
MOV #ER9008, ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.

REPORT ERROR WITH NUMBER INITIAL ERRNR + 1.

"UNEXPECTED SELFTEST ERROR CODE FOR LINE NNN IN FIFO AFTER RESET:

ERROR

EXIT THIS ROUTINE IF EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED.

BIT #8106, OPTION ;HAS EXTENDED ERROR REPORTING BEEN REQUESTED?
BNE 10h ;AVOID SET THE FLAG IF IT HAS AND GO TO 
THE END OF THE LOOP.
MOV 01h, EXOERR ;SET THE "EXIT ON ERROR" FLAG
BR 50h ;EXIT THE ROUTINE WITH FAILURE SINCE THE FIFO
IS NOT PURGED.

WE HAVE A NON-SELFTEST CODE (EITHER BMP CODE OR DATA CHAR).

MOV R5, ERRNR ;RESTORE ERROR NUMBER TO INITIAL VALUE.
ADD #2, ERRNR ;CALCULATE INITIAL ERROR NUMBER PLUS 2.
MOV #ER9019, R1 ;PASS ERROR MESSAGE INFO TO ER9007 ROUTINE.
MOV #ER9007, ERRBLK ;SELECT PROPER ERROR REPORT ROUTINE.

REPORT ERROR WITH NUMBER INITIAL ERRNR + 2.
GLOBAL

SUBROUTINE

MACRO M1200

15 MAR 84 09:15 PAGE 69

GOTO SET UP FOR SELFTEST CODE IF ROM VERSION.

TEST THE ERROR TYPE BITS OF THE CODE.

SKIP THIS ERROR IF BMP CODE.

WE HAVE A SELFTEST CODE: SET UP AND GO TO REPORT ERROR.

SELECT ERROR MSG INFO FOR ER0808 ROUTINE.

GO TO REPORT THIS ERROR.

WE HAVE A BMP CODE: SAVE IT ON THE QUEUE.

SAVE THE BMP CODE ON THE QUEUE.

REPORT THE ERROR WITH ERROR NUMBER OF INITIAL ERRNR = 3.

"UNEXPECTED XXX XXX FOR LINE NA IN FIFO AFTER RESET:"

TRAP CERROR

EXIT THIS ROUTINE IF EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED.

HAS EXTENDED ERROR REPORTING BEEN REQUESTED?

THE END OF THE LOOP.

SET THE "EXIT ON ERROR" FLAG

EXIT THE ROUTINE WITH FAILURE.

END OF LOOP.

COUNT THE CHARACTER WE JUST RECEIVED, AND CHECK FOR TOO MANY RECEIVED.

COUNT THIS CHARACTER.

LOOP IF NOT TOO MANY CHARACTERS PURGED.

WE READ TOO MANY VALID CHARACTERS WHILE TRYING TO PURGE THE FIFO.

REPORT ERROR AND EXIT WITHOUT SUCCESS.

"FIFO WILL NOT PURGE (DATA,VALID STUCK SET), REMAINDER OF TEST SKIPPED."

SELECT PROPER ERROR MESSAGE.

GET INITIAL ERROR NUMBER.

CALCULATE INITIAL ERRNR = 4.

SELECT PROPER ERROR REPORT ROUTINE.

TRAP CERROR

EXIT THIS ROUTINE IF EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED.

HAS EXTENDED ERROR REPORTING BEEN REQUESTED?

AVOID (SET THE FLAG IF IT HAS).

CLEAR THE SUCCESS FLAG.

RESTORE GPRS.
RETURN TO PREGOS SUBRT.

SUCCESS FLAG (SET IF FIFO IS PURGED).
.SBTL GLOBAL SUBROUTINE RXIEO

* THIS ROUTINE IS USED TO DISABLE RECEIVER INTERRUPTS IN THE DMU11.

* INPUTS:   NONE.

* OUTPUTS:  THE RX.INT.ENBL BIT IS CLEARED IN THE DUT CSR.
           IESTST CONTAINS THE UPDATED STATUS OF THE TX AND RX INTERRUPT
           ENABLE BITS.

* CALLING SEQUENCE:  JSR PC,RXIEO

* COMMENTS:  THE CONTENTS OF THE INDIRECT ADDRESS REGISTER FIELD IN
           THE DUT CSR ARE DESTROYED.

* SUBORDINATE ROUTINES CALLED: NONE.

RXIEO:  MOV RO,-(SP) ;SAVE CONTENTS OF RO ON THE STACK.
         GETPRI -(SP) ;SAVE PROCESSOR PRIORIY ON STACK.
         TRAP CIPRI
         MOV RO,(SP)
         SETPRI @PRI07 ;IGNORE ANY INTERRUPT THAT MAY BE GENERATED.
         MOVR @PRI07,RO
         TRAP CIPRI

         BIC #137777,ISTAT ;CLEAR RX.INT.ENBL BIT IN ISTAT.
         MOV ISTAT,ICSRRA ;DISABLE RX INTERRUPTS.
         SETPRI (SP) ;ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
         MOVR (SP),RO
         TRAP CIPRI

         MOV (SP),RO ;RESTORE RO.
         RTS PC
**GLOBAL SUBROUTINE RXIE1**

RECEIVER INTERRUPT ENABLE

**INPUTS:**
NONE.

**OUTPUTS:**
- THE RX.INT.ENBL BIT IS SET IN THE DUT CSR.
- IESTST CONTAINS THE UPDATED STATUS OF THE TX AND RX INTERRUPT ENABLE BITS.

**CALLING SEQUENCE:**
JSR PC,RXIE1

**COMMENTS:**
The contents of the indirect address register field in the DUT CSR are destroyed.

**SUBORDINATE ROUTINES CALLED:**
NONE.
**GLOBAL SUBROUTINE: SAVBMP**

- **INPUTS:**
  - R2 - CONTAINS THE BMP CODE THAT IS TO BE PLACED ON THE QUEUE.
  - BMPQCP - CONTAINS ADDRESS OF NEXT LOCATION IN THE BMP QUEUE.
  - BMPQCB - LABEL AT BASE OF THE BMP CODE QUEUE.
  - BMPQCE - LABEL OF NEXT LOCATION AFTER THE END OF THE BMP QUEUE.
  - TSTNUM - CONTAINS THE NUMBER OF THE CURRENT TEST.

- **OUTPUTS:**
  - BMPQCP - INCREMENTED BY 4.
  - THE CONTENTS OF THE BMP CODE QUEUE ARE UPDATED.

- **CALLING SEQUENCE:**
  - **JSR** PC,S4VBM

- **COMMENTS:**
  - IF THE OVERFLOW OCCURS THEN THE LAST LOCATION WILL BE
    OVERWRITTEN BY ANY SUBSEQUENT ATTEMPTS TO UPDATE THE QUEUE.

- **SUBORDINATE ROUTINES CALLED:**
  - NONE.

- **SOURCE CODE:**

```
SAVBMP:: SAVE
      JSR R5,PREGO   CALL REGISTER SAVE SUBRT.
      MOV BMPQCP,R3  GET THE POINTER TO THE NEXT LOCATION IN QUEUE.
      MOV TSTNUM,(RA).  SAVE THE CURRENT TEST NUMBER ON THE QUEUE.
      INC RA       INCREMENT THE POINTER TO GIVE AN EVEN ADDRESS.
      BIC 177400,R2  CLEAR THE UNWANTED BITS FROM THE BMP CODE.
      MOV R2,(RA).  SAVE THE BMP CODE ON THE QUEUE.
      CMP R4,BMPQCE  CHECK IF OVERFLOW WILL OCCUR THE NEXT TIME.
      SUB R4,R4      GO SAVE THE POINTER IF WE WILL NOT OVERFLOW.
      MOV R4,BMPQCP  FILE THE POINTER.
      JSR R5,PREGO   RESTORE CGRS.
      RTS PC         RETURN TO PREGO SUBRT.
```
```
**SUBROUTINE**

```
; ****************************  **SKPSTS**
; |           **SKIP SEL TEST ROUTINE** -
; |  THIS SUBROUTINE IS USED TO SKIP THE SELFTEST AFTER A DUT RESET HAS BEEN
; |  INITIATED. IT MUST BE ENTERED IMMEDIATELY AFTER SETTING THE DUT MASTER
; |  OR AFTER THE EXECUTION OF A BUS RESET (BECAUSE OF TIMING
; |  CONSIDERATIONS).
; |  **INPUTS:** CSRA - CONTAINS ADDRESS OF THE DUT CSR.
; |  TXBFCA CONTAINS ADDRESS OF DUT DMA BUFFER COUNT REGISTER.
; |  **OUTPUTS:** SKIP SELFTEST CODES ARE WRITTEN TO THE DUT REGISTERS.
; |  **CALLING SEQUENCE:** JSR PC,SKPSTS
; |  **COMMENTS:**
; |  **SUBORDINATE Routines Called:** DELAY.

SKPSTS:: SAVE
JSR R5,PREG05 CALL REGISTER SAVE SUBRAT.
MOV #10,R4 PASS DELAY VALUE OF 10 MILLI-SECONDS.
JSR PC,DELAY DELAY FOR 10 MILLI-SECONDS.

; WRITE SKIP SELF-TEST CODE (52525) TO ALL THE INDEXED DUT REGISTERS.
; MOV #NUMLN5,BIT05,R1 FORM IND.ADR.REG FIELD (PLUS M.R. BIT) WORD.
; THE ABOVE INCLUSION OF THE M.R. BIT IS NECESSARY BECAUSE OF THE
; LACK OF A M.R. BIT WRITE LOCK-OUT ON THE DMU-11.
; MOV #52525,R3 INITIALISE THE SKIP SELF-TEST CODE.
; MOV CSRA,R4 GET THE ADDRESS OF THE CSR OF THE DUT.
; MOV R1,(R4) SELECT A BANK OF DUT REGISTERS.
; MOV R3,(R4) WRITE THE CODE TO A DUT REGISTER.
; CMP R4,TXBFCA COMPARE POINTER WITH LAST REGISTER ADDRESS.
; BLO 60 LOOP IF NOT ALL REGS DONE IN THIS BANK.
; BIT #17,R1 TEST FOR IND.ADR.REG FIELD DECREMENTED TO 0.
; BNE 40 LOOP UNTIL ALL REGISTERS CONTAIN THE CODE.
; PASS
; JSR PC,(&SP)^2 RETURN TO PREG05 SUBRAT.
```
GLOBAL SUBROUTINE SWAPO

;***********************************************************************
;*  ***********************************************************************
;*  SWAP GPRS WITH GPR SET 0 ROUTINE.
;*  THIS SUBROUTINE SWAPS THE PRESENT CONTENTS OF GPRS R1 THRU R5 WITH
;*  THE CONTENTS OF THE NUMBER ZERO GPR SAVE AREA. THE CONTENTS OF RO
;*  ARE NOT ALTERED BY THIS SUBROUTINE.
;*  INPUTS:  GPR CONTENTS R1 THRU R5.
;*  GPRSOB  LABEL AT BASE OF GPR SAVE AREA NUMBER ZERO.
;*  OUTPUTS: R1 THRU R5 CONTAIN THE PREVIOUS CONTENTS OF GPR SAVE AREA
;*            ZERO WORDS 1 THRU 5 RESPECTIVELY.
;*            GPRSO  GPR SAVE AREA 0 WORDS 1 THRU 5, CONTAIN PREVIOUS
;*            CONTENTS OF GPRS R1 THRU R5 RESPECTIVELY.
;*  CALLING SEQUENCE:  JSR  PC,SWAPO
;*  COMMENTS:  THE STATE OF THE CARRY FLAG IS NOT ALTERED BY THIS ROUTINE.
;*  SUBORDINATE ROUTINES CALLED:  NONE.
;***********************************************************************

SWAPO:  MOV  RO, -(SP) ;SAVE THE CONTENTS OF RO.
;*  LOAD THE STACK FROM THE GPRS.
;  MOV  R1, -(SP) ;SAVE THE CONTENTS OF R1.
;  MOV  R2, -(SP) ;SAVE THE CONTENTS OF R2.
;  MOV  R3, -(SP) ;SAVE THE CONTENTS OF R3.
;  MOV  R4, -(SP) ;SAVE THE CONTENTS OF R4.
;  MOV  R5, -(SP) ;SAVE THE CONTENTS OF R5.

;*  LOAD THE GPRS FROM THE GPR SAVE AREA 0.
;  MOV  #GPRSOB,RO ;GET THE BASE ADDRESS OF GPR SAVE AREA 0.
;  MOV  (RO), R1 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 1.
;  MOV  (RO), R2 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 2.
;  MOV  (RO), R3 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 3.
;  MOV  (RO), R4 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 4.
;  MOV  (RO), R5 ;LOAD R1 WITH GPR SAVE AREA 0 WORD 5.

;*  LOAD THE GPR SAVE AREA 0 FROM THE STACK.
;  MOV  (SP)+, (RO) ;LOAD GPR SAVE AREA 0 WORD 5 WITH SAVED R5.
;  MOV  (SP)+, (RO) ;LOAD GPR SAVE AREA 0 WORD 4 WITH SAVED R4.
;  MOV  (SP)+, (RO) ;LOAD GPR SAVE AREA 0 WORD 3 WITH SAVED R3.
;  MOV  (SP)+, (RO) ;LOAD GPR SAVE AREA 0 WORD 2 WITH SAVED R2.
;  MOV  (SP)+, (RO) ;LOAD GPR SAVE AREA 0 WORD 1 WITH SAVED R1.

;*  RESTORE THE INITIAL VALUE OF RO.
MOV  (SP)+, RO
RTS  PC
**TSABRT:**

```
TSABRT: SAVE JR, R5, PREGOS
     JSR R5, PREGOS
     ; CALL REGISTER SAVE SUBRT.
     MOV @24, R1
     ; PASS ADDRESS OF FIRST MESSAGE TO BE REPORTED.
     MOV @ER1603, ERABLK
     ; SET-UP THE ERROR REPORTING ROUTINE.
     ERROR
     ; ERROR
     TRAP CERROR
     BR 601
     .ASCIIZ "NON-RELATED TEST ERROR FOUND DURING TEST EXECUTION"
```

```
601: PASS JR, PC, (ESP)
     ; RETURN TO PREGOS SUBRT.
```
**SUBROUTINE**: TXDSBL

**TRANSMITTER DISABLE**

**THIS SUBROUTINE IS USED TO DISABLE TRANSMISSION ON SELECTED LINES BY**, CLEARING THE ASSOCIATED TX.ENABLE BIT ON THE OUT.

**INPUTS**: 
- R5: BIT'S SET CORRESPOND TO LINES ON WHICH TO CLEAR TX.ENABLE.
- CSRA: CONTAINS THE ADDRESS OF THE OUT CSR.
- IESTAT: CONTAINS THE STATE OF TXIE AND RXIE BITS IN THE CSR.
- NUMLN: EQUAL TO THE MAXIMUM NUMBER OF LINES AVAILABLE.
- TXAD2A: CONTAINS THE ADDRESS OF THE TXBUFFAD2 REGISTER.

**OUTPUTS**: 
- R5: BIT'S SET INDICATE THE INITIAL STATES OF ALL TX.ENABLE BITS.
- TXBUFFAD2: THE STATE OF THE TX.ENABLE BIT MAY BE ALTERED.
- THE CONTENTS OF THE IND.ADD.ENABLE FIELD IN THE CSR ARE DESTROYED.

**CALLING SEQUENCE**: 
- JSR PC, TXDSBL

**COMMENTS**: 
- SUBORDINATE ROUTINES CALLED: NONE.

**TXDSBL**: 

```
; SAVE REGS: R5,PC
MOV R5,R0 ; COPY CONTENTS OF R5 INTO R0
JSR SUBRT ; CALL SUBROUTINE SAVE RTS
MOV R5,R0 ; COPY BIT MAP OF LINES TO DISABLE TO R5.
MOV @BITM,R1 ; INITIALIZE THE SELECTED LINE BIT MASK.
MOV TXAD2A,R2 ; GET THE ADDRESS OF THE TXBUFFAD2 REGISTER.
INC R2 ; GET THE ADDRESS OF THE MSB BYTE OF THE TXBUFFAD2 REG.
MOV @NUMLN,R3 ; GET MAXIMUM LINE NUMBER PLUS ONE.
MOV IESTAT,R4 ; GET THE STATES OF THE INT.ENABLE BITS.
CLR R5 ; CLEAR POSSIBLE TX DISABLED ON ALL LINES.
```

**SELECT EVERY LINE IN TERN AND LOG THE STATE OF EACH TX.ENABLE BIT.**

```
MOV R4,BCSR ; WRITE TO CSR TO SELECT SINGLE LINE REGISTERS.
TSTB @R2 ; CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.
BPL 41 ; SKIP NEXT INSTRUCTION IF TX.ENABLE CLEAR.
BTS R1,R5 ; LOG TX ENABLED BIT SET FOR SELECTED LINES.
```

**CLEAR TX.ENABLE ON LINES THAT HAVE A CORRESPONDING BIT SET IN THE TX DISABLE LINE BIT MAP.**

```
BIT R1,R0 ; CHECK STATE OF DISABLED LINE BIT MAP.
BEQ 61 ; BRANCH IF THIS LINE TO REMAIN UNALTERED.
BIDC @BITM,(R2) ; CLEAR TX.ENABLE BIT ON SELECTED LINE.
```

**PREPARE TO SELECT REGISTERS FOR NEXT LINE.**

```
INC R4 ; SHIFT BIT MAP FOR NEXT LINE.
ASL R1 ; DECREMENT LINE NUMBER.
```

```
BNE 29 ; LOOP TO CHECK NEXT LINE.
```

```
PUSH R5 ; RESTORE REGS, EXCEPT
MOV @R5,SP ; R5, RX5, OT(SP)
JSR PC, @SP ; PUT R5 IN STACK SLOT.
RTS PC ; RETURN TO PREGS SBR.
```

R5: PREVIOUS STATES OF ALL TX.ENABLE BITS.
.SBTIL GLOBAL SUBROUTINE TXENBL

**TRANSMITTER ENABLE**

**THIS SUBROUTINE IS USED TO ENABLE TRANSMISSION ON SELECTED LINES BY**
**SETTING THE ASSOCIATED TX.ENABLE BIT ON THE OUT.**

**INPUTS:**
- R5: BIT'S SET CORRESPOND TO LINES ON WHICH TO SET TX.ENABLE.
- CSRA - CONTAINS THE ADDRESS OF THE DUT CSR.
- IESTAT - CONTAINS THE STATE OF TXIE AND RXIE BITS IN THE CSR.
- NUMLN - EQUATED TO BE THE MAXIMUM NUMBER OF LINES AVAILABLE.
- TXADDA - CONTAINS THE ADDRESS OF THE TBUFFAD2 REGISTER.

**OUTPUTS:**
- R5: BIT'S SET INDICATE PREVIOUSLY DISABLED LINES.
- TBUFFAD2: THE STATE OF THE TX.ENABLE BIT MAY BE ALTERED.
- THE CONTENTS OF THE INO.ADD.REG.FIELD IN THE CSR ARE DESTROYED.

**CALLING SEQUENCE:**
JSR PC, TXENBL

**COMMENTS:**

**SUBORDINATE ROUTINES CALLED:**
NONE.

---

TXENBL::

SAVE 
JSR R5,PREGOS ;CALL REGISTER SAVE SUBRT.

MOV R5,R0 
;COPY BIT MAP OF LINES TO ENABLE

MOV &LBITO.R1 
;INITIALIZE THE SELECTED LINE BIT MASK

MOV TXADDA,R2 
;GET THE ADDRESS OF THE TBUFFAD2 REGISTER

INC R2 
;GET THE ADDRESS OF THE MSBITE OF TBUFFAD2 REG.

MOV &LMSLS,R3 
;GET MAXIMUM LINE NUMBER.

MOV IESTAT,R4 
;GET THE STATES OF THE INT ENABLE BITS.

CLR R5 
;CLEAR TX.ENABLE BIT LOG OF DISABLED LINES.

;SELECT EVERY LINE IN TURN, AND LOG ANY TX.ENABLE BIT THAT IS CLEAR.

21: MOV R4, ACSRA 
;WRITE TO DUT CSR TO SELECT LINE REGISTERS.

JSR (R2) 
;CHECK STATE OF TX.ENABLE BIT ON SELECTED LINE.

BMI 41 
;SKIP NEXT INSTRUCTION IF TX.ENABLE SET.

BIS R1,R5 
;LOG TX ENABLE BIT CLEAR FOR SELECTED LINE.

;SET TX.ENABLE ON LINES THAT HAVE A CORRESPONDING BIT SET IN THE TX ENABLE
;LINE BIT MAP.

41: BIT R1,R0 
;CHECK STATE OF TX.ENABLE LINE BIT MAP.

JSR 61 
;BRANCH IF THIS LINE TO REMAIN UNALTERED.

BIS RBIT7,(R2) 
;ENABLE TRANSMISSION ON SELECTED LINE.

61: INC R4 
;PREPARE TO SELECT REGISTERS FOR NEXT LINE.

JSR 62 
;SHIFT BIT MAP FOR NEXT LINE.

ASL R1 
;DECREMENT LINE NUMBER.

DEC R5 
;LOOP TO CHECK NEXT LINE.

BNE 21

60: PASS R5 
;RESTORE GPR5,EXCEPT

JSR (RSP)

MOV R5,RSSLOT;SP
;PUT R5 IN STACK SLOT.

JSR PC
;RETURN TO PREGOS SUBRT.

R5 LINE BIT MAP CORRESPONDING TO THE
;PREVIOUS LINES THAT WERE DISABLED.

ATS PC
**GLOBAL SUBROUTINE TXIEO**

---

**THIS Routine IS USED TO DISABLE TRANSMITTER INTERRUPTS IN THE DMU11.**

**INPUTS:**

**NONE.**

**OUTPUTS:**

- **TX.INT.ENBL BIT IS CLEARED IN THE OUT CSR.**
- **IESTST -CONTAINS THE UPDATED STATUS OF THE TX AND RX INTERRUPT ENABLE BITs.**

**CALLING SEQUENCE:**

JSR PC, TXIEO

**COMMENTS:**

- **THE CONTENTS OF THE INDIRECT ADDRESS REGISTER FIELD IN THE DUT CSR ARE DESTROYED.**
- **SUBORDINATE ROUTINES CALLED:**

**NONE.**

**TXIEO:**

```assembly
MOVR, (SP) ; SAVE CONTENTS OF RO ON THE STACK.
GETPRI -(SP) ; SAVE CURRENT PROCESSOR PRIORITY ON THE STACK.
TRAP C@GPRI
MOVR, (SP) ; CLEAR TX.INT.ENBL BIT IN IESTAT.
BIC #177677, IESTAT
MOVI IESTAT, &CSRA ; CLEAR TX INTERRUPTS.
SETPRI (SP) ; ENABLE INTERRUPTS TO THE PROCESSOR AGAIN.
MOVT, (SP) ; RESTORE RO.
RTS PC
```
GLOBAL SUBROUTINE

;;;;; ... TXIE1

;; TRANSmitter INTERRUPT ENABLE
;;;;; THIS ROUTINE IS USED TO ENABLE TRANSMITTER INTERRUPTS IN THE DMU11.

;; INPUTS: NONE.

;; OUTPUTS: THE TX.INT.ENBL BIT IS SET IN THE OUT CSR.
;;;;; IESTAT - CONTAINS THE UPDATED STATUS OF THE TX AND RX INTERRUPT ENABLE BITS.

;; CALLING SEQUENCE: JSR PC, TXIE1

;; COMMENTS: THE CONTENTS OF THE INDIRECT ADDRESS REGISTER FIELD IN THE OUT CSR ARE DESTROYED.

;; SUBORDINATE ROUTINES CALLED: NONE.

; TXIE1:

023034  052767  040000  157246  BIS  #BIT14, IESTAT ; SET TX.INT.ENBL BIT IN IESTAT.
023042  042767  137677  157240  BIC  #137677, IESTAT ; CLEAR ALL BITS EXCEPT TX RX I.E BITS.
023050  016777  157234  157164  MOV  IESTAT, &CSRRA ; ENABLE TX INTERRUPTS.
023056  000207  157200  157164  RTS  PC
.SBITL GLOBAL SUBROUTINE - UNSDIV

.: THIS SUBROUTINE IS USED TO DIVIDE A 32 BIT UNSIGNED DIVIDEND BY A
.: 16 BIT UNSIGNED DIVISOR GIVING A 16 BIT QUOTIENT. ALL NUMBERS ARE
.: CONSIDERED TO BE UNSIGNED. A SUCCESS FLAG IS NOT SET ON RETURN IF
.: THE QUOTIENT WAS TOO BIG TO BE CONTAINED IN 16 BITS.
.: INPUTS: R1 - THE DIVISOR, UNSIGNED, 16 BITS.
.: R2 - MOST SIGNIFICANT WORD OF THE DIVIDEND, UNSIGNED, 16 BITS.
.: R3 - LEAST SIGNIFICANT WORD OF THE DIVIDEND, UNSIGNED, 16 BITS.
.: OUTPUTS: R1 QUOTIENT, UNSIGNED, 16 BITS (177777 IF OVERFLOW)
.: CARRY - SUCCESS FLAG, SET IF COMPLETE QUOTIENT FITS IN 16 BITS.
.: CALLING SEQUENCE: JSR PC,UNSDIV
.: COMMENTS: IF THE DIVISOR IS 0 THE QUOTIENT IS RETURNED AS ALL ONES
.: (177777) AND THE CARRY IS CLEAR REGARDLESS OF THE DIVIDEND.
.: SUBORDINATE ROUTINES CALLED: NONE.

UNSDIV:: SAVE R0 R1 R2 R3 R4 R5 R6 R7
SAVE CONTENTS OF GPR R0 THRU R5.
JSR R5,PREGOS ;CALL REGISTER SAVE SUBRT.

; CHECK FOR QUOTIENT GREATER THAN 16 BITS CONDITION.
JSR R3,PREGOS ;CALL REGISTER SAVE SUBRT.
MOV R2,R4 ;GET MSW OF DIVIDEND FOR SUBTRACT.
SUB R1,R4 ;SUBTRACT DIVISOR FROM MSW OF DIVIDEND.
BCS 2# ;IF IT DIDN'T GO, WE HAVE QUOTIENT < 16 BITS.
MOV 8-1,R1 ;SET QUOTIENT TO ALL ONES (177777).
BR 604 ;EXIT WITH CARRY CLEAR.

; SET UP COUNTERS AND VARIOUS WORKING GPRs.
CLR R4 ;CLEAR THE LSW OF THE DIVISOR.
CLC ;CLEAR CARRY FOR THE SHIFT OF THE DIVISOR.
ROR R1 ;DIVISOR BY 1.
ROR R4 ;2(UNSIGNED)
MOV 016,.RO ;SET UP INITIAL SHIFT COUNT TO 16.

; THE SUBTRACT AND SHIFT LOOP.

2#: MOV R2,(SP) ;SAVE MSWORD OF DIVIDEND.
MOV R3,(SP) ;SAVE LSWORD OF DIVIDEND.
SUB R4,R3,R1 ;SUBTRACT DIVIDEND - LSWORD OF DIVISOR.
SBC R2,R4,R1 ;MSWORD DIVIDEND - MSWORD OF DIVISOR.
BCS 6# ;IF BORROW FROM BORROW SUBTRACT, IT DIDN'T GO.
SUB R1,R2,R1 ;MSWORD DIVIDEND - MSWORD OF DIVISOR.
BCC 8# ;IF NO BORROW, IT WENT, CARRY IS CLEAR.

; I DIDN'T GO, SO WE SHIFT A 1 INTO THE QUOTIENT (COMPLEMENTED LATER).
; CARRY IS SET.

4#: MOV R2,(SP)
MOV R3,(SP)
SUB R4,R3
SBC R2,R4
BCS 6#
SUB R1,R2
BCC 8#

; I DIDN'T GO, SO WE SHIFT A 1 INTO THE QUOTIENT (COMPLEMENTED LATER).
; CARRY IS SET.

6#: MOV (SP)+,R3 ;RESTORE LSWORD OF DIVIDEND.
MOV (SP)+,R2 ;RESTORE MSWORD OF DIVIDEND.
BR 10$ ; GOTO SHIFT 1 INTO THE QUOTIENT.

IT WENT, SO WE RESTORE THE STACK AND SHIFT A 0 INTO QUOTIENT (WILL BE COMPLEMENTED LATER). CARRY IS CLEAR.

MOV (SP), (SP) ; POP THE SAVED DIVIDEND OFF THE STACK.

SHIFT THE RESULT OF THE SUBTRACT ATTEMPT INTO THE QUOTIENT SHIFT REG.

10$: ROL R5 ; SHIFT NEXT BIT INTO THE INVERTED QUOTIENT.

CLC ; DIVIDE THE
ROR R1 ; DEVISOR BY 2 (UNSIGNED).
DEC R0 ; COUNT THIS SHIFT AND SUBTRACT.
BNE 4$ ; LOOP FOR ANOTHER SHIFT & SUB IF NOT DONE.

GET QUOTIENT FROM INVERTED QUOTIENT.

NOW WE EITHER ROUND UP OR LEAVE QUOTIENT ALONE.

CLC ; CLEAR THE CARRY FOR THE SHIFT OF THE DIVIDEND.
ROL R3 ; MULTIPLY LSWORD OF DIVIDEND BY 2, MSWORD IS 0.
BNS 12$ ; IF CARRY FROM SHIFT, ROUND UP.

SUB R4, R3 ; SUBTRACT DIVISOR FROM DIVIDEND.
BNS 14$ ; IF BORROW, DON'T ROUND UP.

ROUND UP, EXTRA SUBTRACT WENT.

INC R5 ; INCREMENT THE QUOTIENT BY ONE.
BNE 14$ ; IF NO OVERFLOW, WE LEAVE THE ROUND UP.
DEC R5 ; DON'T LET ROUNING CAUSE OVERFLOW.

ALL DONE, PASS QUOTIENT AND EXIT.

MOV R5, R1 ; PASS QUOTIENT BACK IN R1.
SEC ; INDICATE NO OVERFLOW.

RESTORE GPRs, LEAVE THE FOLLOWING INTACT:

MOV R1, R1, RISLOT(SP) ; PUT R1 IN STACK SLOT.
JSR PC, BC, (SP) ; RETURN TO PREGOS SUBR.

; R1 16 BIT, UNSIGNED QUOTIENT.

; CARRY - SET INDICATES NO OVERFLOW (SUCCESS).
**SBTIL GLOBAL SUBROUTINE WAIBIC**

---

THIS SUBROUTINE WaITS FOR THE SPECIFIED BIT TO BECOME CLEAR. IF THE SPECIFIED BIT GOES TO A CLEAR STATE WITHIN THE SPECIFIED TIME OUT PERIOD A SUCCESS INDICATION IS RETURNED BY THIS ROUTINE. THE LAST VALUE WHICH IS READ LOOKING FOR THE CONDITION IS RETURNED TO ALLOW THE USE OF THIS ROUTINE TO LOOK FOR DESTRUCTIVE READ CONDITIONS.

**INPUTS:**
- R1 - TIME-OUT VALUE AND BIT NUMBER INDICATION;
- BITS 15 THRU 12 - NUMBER OF BIT TO TEST (RANGE 0 THRU 15);
- BITS 11 THRU 0 - TIME-OUT VALUE IN MILLI SECONDS (4095 MAX).
- R2 - ADDRESS OF WORD CONTAINING THE BIT TO TEST.

**OUTPUTS:**
- R2 - THE LAST WORD WHICH WAS READ TO CHECK FOR THE CONDITION.
- CARRY - SUCCESS FLAG (CARR: SET IF BIT CLR BEFORE TIME-OUT).

**CALLING SEQUENCE:**
- MOV $130040,R1 ;PASS BIT 11 (13 OCTAL) AND
- MOV $132000,R2 ;TEST BIT IN WORD AT 'LABEL'.
- JSR PC,WAIBIC ;WAIT 32 MS FOR BIT 11 TO CLR.

**COMMENTS:**
- SUBORDINATE ROUTINES CALLED: MSLGET.

---

**WAIBIC:**

- SAVE CONTENTS OF GPRS RO THRU R5.
- JSR R5,PREGOS ;CALL REGISTER SAVE SUBRT.
- MOV R2,R4 ;SET UP THE ADDRESS PARAMETER FOR MSLGET.
- MOV R1,R2
- BIC $170000,R1
- BIC $7777,R2 ;SEPARATE DELAY COUNT OUT OF PASSED PARAMETER.
- MOV R3,R2 ;SEPARATE LINE NUMBER FIELD OF PASSED PARAM.
- SWAB R2 ;PUT LINE NUMBER FIELD IN LSBYTE.
- ASR R2 ;SHIFT THE LINE NUMBER FIELD INTO THE PROPER POSITION TO USE IT AS A WORD TABLE OFFSET.
- ASR R2 ;FOR THE TABLE LOOKUP OF THE LINE BIT MAP.
- MOV BITBL(R2),R2 ;GET BIT MAP OF LINE TO TEST FROM TABLE.
- CLR R3 ;INDICATE THAT THE BIT SHOULD BE CLR.
- JSR PC,MSLGET ;WAIT FOR THE BIT TO BE CLR WITHIN TIME OUT.
- JSR PC,MSLGET ;CARRY IS CORRECT UPON MSLGET RETURN.
- JSR PC,MSLGET ;RESTORE GPRS, EXCEPT THE FOLLOWING:
- RTS PC ;R2 LAST VALUE READ LOOKING FOR CONDITION.
- CARRY SUCCESS FLAG (SET IF BIT FOUND CLR).
.SUBTIL GLOBAL SUBROUTINE WAIWIBS

********************************************************************
| * WAIT FOR BIT SET ROUTINE *
| * THIS SUBROUTINE WAITS FOR THE SPECIFIED BIT TO BECOME SET. IF THE |
| * SPECIFIED BIT GOES TO A SET STATE WITHIN THE SPECIFIED TIME OUT |
| * PERIOD A SUCCESS INDICATION IS RETURNED BY THIS ROUTINE. |
| * THE LAST VALUE WHICH IS READ LOOKING FOR THIS CONDITION IS RETURNED TO |
| * ALLOW THE USE OF THIS ROUTINE TO LOOK FOR DESTRUCTIVE READ CONDITIONS. |
| * ******************************************************************** |
| * INPUTS: R1 - TIME-OUT VALUE AND BIT NUMBER INDICATION; |
| * BITS 15 THRU 0 - NUMBER OF BIT TO TEST (RANGE 0 THRU 15); |
| * R2 - ADDRESS OF WORD CONTAINING THE BIT TO TEST. |
| * ******************************************************************** |
| * OUTPUTS: R2 - THE LAST WORD WHICH WAS READ TO CHECK FOR THE CONDITION. |
| * CARRY - SUCCESS FLAG (CARRY SET IF BIT SET BEFORE TIME-OUT). |
| * ******************************************************************** |
| * CALLING SEQUENCE: MOV #150040,R1; PASS BIT 11 (13 OCTAL) AND |
| * MOV #32 (40 OCTAL) MS DELAY; |
| * JSR PC, WAIWIBS; WAIT 32 MS FOR BIT 11 TO SET. |
| * ******************************************************************** |
| * COMMENTS: |
| * ******************************************************************** |
| * SUBORDINATE ROUTINES CALLED: MSLGET. |
| * ******************************************************************** |

WAIWIBS:

SAVE

JSR R5, PREGOS; CALL REGISTER SAVE SUBRT.

MOV R2, R4

MOV R2, R4

MOV R1, R2

BIC #7777, R2

SWAB R2

ASR R2

JSR PC, MSLGET

MOV R2, R3

MOV R2, R3

MOV B1TBL(R2), R2

MOV R0, R2

JSR PC, MSLGET

601: PASS R2

MOVR2, R2SLTOP; PUT R2 IN STACK SLOT.

JSR PC, RSP; RETURN TO PREGOS SUBRT.

RTS PC

CARRY - SUCCESS FLAG (SET IF BIT FOUND SET)
.SBTL GLOBAL SUBROUTINE - WOPDR

*** ***********************************************************************
*** WRITE DATA PATTERN TO DEVICE REGISTERS
*** THIS ROUTINE WRITES A ROTATED DATA PATTERN TO EACH OF THE 6 DEVICE
*** REGISTERS OF EACH ACTIVE LINE OF THE DEVICE UNDER TEST.
*** THE DATA PATTERN IS ROTATED ONCE AFTER EACH WRITE TO A DEVICE REGISTER
*** ON A PARTICULAR LINE. THE STARTING DATA PATTERN FOR EACH LINE
*** IS ROTATED ONCE AFTER WRITING ALL THE REGISTERS ON A PARTICULAR
*** LINE. THIS LEADS TO THE FOLLOWING DATA PATTERN:
***  LINE 0, REGISTER 0 - SHIFTED 0 BIT POSITIONS
***  LINE 0, REGISTER 1 - SHIFTED 1 BIT POSITION
***  LINE 1, REGISTER 0 - SHIFTED 1 BIT POSITIONS
***  LINE 2, REGISTER 1 - SHIFTED 2 BIT POSITIONS
*** ANY BITS FIELDS IN THE DEVICE REGISTERS THAT CANNOT BE ALTERED
*** ARE MASKED OUT OF THE DATA PATTERN BEFORE IT IS WRITTEN.
*** THIS ROUTINE WILL USE EITHER MOV, MOVB, BIS, BISB, BIC, OR BICB
*** INSTRUCTIONS. THE UPPER OR LOWER BYTE CAN BE SPECIFIED FOR WRITING.
*** Inputs:
***  R2 - USED TO PASS IN THE DATA PATTERN TO BE ROTATED & WRITTEN.
***  R3 - BYTE INDICATOR (- = LO BYTE, + = HI BYTE, 0 = BOTH).
***  R4 - OPERATION TYPE INDICATOR ( + = BIC, + = BIS, 0 = MOV).
***  AS X - BIT MAP OF THE ACTIVE LINES ON THE DEVICE UNDER TEST.
***  CSR - CONTAINS THE CSR ADDRESS OF THE DEVICE UNDER TEST.
***  DRADRT - BASE ADDRESS OF DEVICE REGISTER ADDRESS TABLE.
***  LPRO - EQUATED TO LPR REG OFFSET FROM DEVICE CSR ADDRESS.
***  NUMLNS - NUMBER OF LINES ON THE DEVICE UNDER TEST.
***  TBUFFREG - EQUATED TO TBUFF REG OFFSET FROM DEVICE CSR ADDRESS.
***  UNBTTB - BASE ADDRESS OF THE UNUSED BIT TABLE.
*** Outputs:
***  DEVICE REGISTERS ON ALL ACTIVE DEVICE LINES ARE MODIFIED.
*** Calling Sequence:
***  JSR WOPDR
*** Comments:
***  THIS ROUTINE DOES NOT WRITE DATA TO THE FOLLOWING REGISTERS,
***  RBUF, RXTIMER, STAT, FIFO SIZE, FIFO DATA.
***  THE CSR IS CLEARED EXCEPT FOR THE IND.ADR.REG FIELD.
*** Subordinate Routines Called: ROLDAP.

******************************************************************************
WOPDR: SAVE
      JSR R5, PREG05 ;CALL REGISTER SAVE SUBRT.
      SET UP OUTER LOOP WHICH WRITES THE DATA PATTERN TO EACH LINE'S REGISTERS
      CLR R5 ;CLEAR LINE COUNTER TO SELECT LINE 0.
      THE OUTER LOOP FOLLOWS. EACH PASS THROUGH THIS LOOP WRITES DATA TO ALL OF
      THE DEVICE REGISTERS FOR A PARTICULAR LINE IF IT IS ACTIVE.
      MOV R2, R4 ;SAVE THE OUTER LOOP DATA PATTERN.
GLOBAL SIMOUTINE

4165 023354 010577 156662 MOV R5,0CSRA ;SET CSR IND.ADR.REG FIELD TO THIS LINE.
4166 023360 006505 ASL R5 ;TURN LINE NUMBER INTO A WORD OFFSET.
4167 023362 036567 002410 156640 BIT BITBL(R5),ACTNF
4168 023370 001456 BEQ 20$ ;LINE ACTIVE? NO, SKIP THIS LINE.
4169 023372 012701 000004 MOV @LPRO,R1 ;YES, INITIALIZE THE REGISTER OFFSET.
4170
4171 ;THE INNER LOOP FOLLOWS. EACH PASS THROUGH THIS LOOP WRITES DATA TO A
4172 ;DEVICE REGISTER.
4173
4174 023376 010200 41: MOV R2,R0
4175 023400 046100 002370 BIC UNBITB(R1),R0 ;CLEAR BIT FIELDS FOR UNUSED REGISTER BITS.
4176 023404 016103 002242 MOV DRADE(R1),R3 ;GET THE ADDRESS OF THE DEVICE REGISTER.
4177 023410 005766 000010 TST R3SLOT(SP) ;CHECK THE OPERAND TYPE INDICATOR.
4178 023414 003040 BLE 61$ ;HIGH BYTE? NO, SKIP HIGH BYTE ADDRESS SET UP.
4179 023416 005203 INS R3 ;YES, SET THE REG ADDRESS TO THE HIGH BYTE.
4180 023420 000500 SWAB RO
4181 023422 005766 000010 61: TST R3SLOT(SP) ;CHECK THE OPERAND TYPE INDICATOR.
4182 023426 001412 BR 12$ ;WORD ACCESS? YES, GO PERFORM WORD ACCESS.
4183
4184 ;PERFORM BYTE ACCESS TO THE SPECIFIED BYTE OF THE SPECIFIED REGISTER.
4185
4186 023430 005766 000012 TST R3SLOT(SP) ;NO, CHECK THE ACCESS TYPE INDICATOR.
4187 023434 100403 BMI 81$ ;USE BIC? YES, GO PERFORM BICB INSTRUCTION.
4188 023436 001404 BEQ 10$ ;USE MOV? YES, GO PERFORM MOVB INSTRUCTION.
4189 023440 150018 BISB RO,(R3) ;NEITHER. PERFORM BISB ACCESS TO REGISTER.
4190 023442 000415 BR 18$ ;MOV RO,(R3)
4191 023444 140013 BMBC RO,(R3) ;PERFORM BICB ACCESS TO REGISTER.
4192 023446 000413 BR 18$ ;PERFORM MOVB ACCESS TO REGISTER.
4193 023450 110013 10$: MOV RO,(R3)
4194 023452 000441 BR 18$ ;PERFORM MOV ACCESS TO REGISTER.
4195
4196 ;PERFORM WORD ACCESS TO THE SPECIFIED REGISTER.
4197
4198 023454 005766 000012 12$: TST R4SLOT(SP) ;CHECK THE ACCESS TYPE INDICATOR.
4199 023460 100403 BMI 14$ ;USE BIC? YES, GO PERFORM BICB INSTRUCTION.
4200 023462 001404 BEQ 16$ ;USE MOV? YES, GO PERFORM MOVB INSTRUCTION.
4201 023464 050013 BISB RO,(R3) ;NEITHER. PERFORM BISB ACCESS TO REGISTER.
4202 023466 000403 BR 18$ ;MOV RO,(R3)
4203 023470 040013 14$: MOV RO,(R3)
4204 023472 004013 BR 18$ ;PERFORM BIC ACCESS TO REGISTER.
4205 023474 010013 16$: MOV RO,(R3)
4206 ;PERFORM MOV ACCESS TO REGISTER.
4207
4208 ;PREPARE THE DATA PATTERN AND OFFSET FOR THE NEXT REGISTER ON THIS LINE.
4209 023476 004767 175766 18$: JSR PC,ROLDAP ;ROTATE DATA PATTERN LEFT, NOT THROUGH CARRY.
4210 023502 062701 000002 18$: ADD $2R1,PC,ROLDAP ;INCREMENT OFFSET FOR NEXT REGISTER.
4211 023506 020127 000006 CMP R1,R2,ROLDAP ;CHECK IF THIS IS THE FIFOSIZE/DATE REG
4212 023512 001002 BNE 19$ ;AVOID ALTERING THE OFFSET IF IT ISN'T.
4213 023514 062701 000002 ADD #2R1,ROLDAP ;AVOID TESTING THESE REGISTERS.
4214 023520 020127 000016 CMP R1,R2,ROLDAP ;COMPARE REG OFFSET WITH OFFSET OF LAST REG.
4215 023524 003724 BLE 41$ ;LOOP IF NOT ALL REG DONE FOR THIS LINE.
4216
4217 ;BACK INTO THE OUTER LOOP. NOW SET UP FOR NEXT LINE. LOOP IF NOT DONE.
4218
4219 023526 010402 20$: MOV R4,R2 ;SET UP TO ROTATE THE DATA PATTERN.
4220 023530 004767 175734 20$: JSR PC,ROLDAP ;ROTATE THE DATA PATTERN.
4221 023534 006205 ASR R5 ;CONVERT BACK TO LINE NUMBER FROM WORD OFFSET.
GLOBAL SUBROUTINE

4222 023536 005205 INC R5, NUMLNS ; COUNT THIS LINE.
4223 023540 020527 000000 CMP R5, @NUMLNS ; COMPARE LINE COUNT WITH NUMBER OF LINES.
4224 023544 002702 BLI 2# ; LOOP IF SOME LINES NOT DONE.
4225
4226 023546 004736 60$; PASS JSR PC, @SP(); ; RESTORE UPRS.
4227 023550 000207 RTC PC ; RETURN TO PREGOS SUBR.
.SBTN: GLOBAL SUBROUTINE

This subroutine is used to set the device under test (DUT) line control registers (LNCTRL) to the specified state. Only the LNCTRLs for the specified lines are altered.

Inputs:
- RO: New line parameters.
- R5: Bit map of lines to be altered
- CSR: Contains address of the DUT CSR.
- IESTAT: Contains the current state of the TX and RX interrupt enable bits in the CSR.
- LNCTRL: Contains address of the DUT LNCTRL registers.

Outputs:
- LNCTRL: Specified OUT line control registers are altered.

Calling sequence: JSR PC, WTLNC

Comments:

Subordinate routines called: ALTFLD.

---

WTLNC:: SAVE

; SAVE CONTENTS OF GPRS RO THRU R5.
JSR RC,PREG05 ; CALL REGISTER SAVE SUBRT.

; SET UP THE PARAMETERS FOR THE CALL TO ALTFLD.

MOV LNCTRL,R1 ; SET UP THE REGISTER ADDRESS PARAMETER.
MOV R0,R2 ; SET UP THE DESIRED REGISTER CONTENTS.
MOV R5,R3 ; SET UP THE BIT MAP OF LINES TO ALTER.
MOV @1,R4 ; SELECT ALL REGISTER BITS TO BE ALTERED.

; CALL THE SUBROUTINE WHICH ALTERS THE REGISTER CONTENTS.
JSR PC,ALTFLD ; ALTER THE REGISTER CONTENTS.

60$: PASS
JSR PC,(SP) ; RESTORE GPRS.
RTS PC

---
SBTTL GLOBAL SUBROUTINE

LINE PARAMETER REGISTER SETUP ROUTINE

THIS SUBROUTINE IS USED TO SET THE DEVICE UNDER TEST (OUT) LINE
PARAMETER REGISTERS (LPR) TO THE SPECIFIED STATE. ONLY THE LPRS FOR
THE SPECIFIED LINES ARE ALTERED.

INPUTS: 
RO NEW LINE PARAMETERS.
R5 - BIT MAP OF LINES TO BE ALTERED.
CSRA - CONTAINS ADDRESS OF THE OUT CSR.
IESTAT - CONTAINS THE CURRENT STATE OF THE TX AND RX INTERRUPT
ENABLE BITS IN THE CSR.
LPRA CONTAINS ADDRESS OF THE OUT LPR.

OUTPUTS: 
LPR SPECIFIED OUT LINE PARAMETER REGISTERS ARE ALTERED.

CALLING SEQUENCE: 
JSR PC, WTLPR

COMMENTS:

SUBORDINATE ROUTINES CALLED: ALTFLD.

WTWLPR: SAVE
;SAVE CONTENTS OF GPRS RO THRU RS.
JSR RS, PREGOS ;CALL REGISTER SAVE SUBRT.

SET UP THE PARAMETERS FOR THE CALL TO ALTFLD.

MOV LPRA, R1 ;SET UP THE REGISTER ADDRESS PARAMETER.
MOV R0, R2 ;SET UP THE DESIRED REGISTER CONTENTS.
MOV R5, R3 ;SET UP THE BIT MAP OF LINES TO ALTER.
MOV @ 1, R4 ;SELECT ALL REGISTER BITS TO BE ALTERED.

CALL THE SUBROUTINE WHICH ALTERS THE REGISTER CONTENTS.

JSR PC, ALTFLD ;ALTER THE REGISTER CONTENTS.

601: PASS
;RESTORE GPRS.

JSR PC, @ (SP), ;RETURN TO PREGOS SUBRT.

RTS PC
**SBTIL** INTERRUPT SERVICE ROUTINE CACHRX

* CATCH RECEIVER INTERRUPT.
* THIS ROUTINE IS USED IN SEVERAL TESTS, TO LOG A COUNT OF THE
* NUMBER OF RECEIVER INTERRUPTS THAT OCCUR.
* INPUTS: CSRA CONTAINS THE ADDRESS OF THE CSR.
* RXINTC HOLDS THE COUNT OF THE NUMBER OF RX INTERRUPTS
* THAT OCCURRED.
* OUTPUTS: RXINTC CONTAINS THE UPDATED INTERRUPT COUNT.
* CALLING SEQUENCE: PUT THE ADDRESS OF THE LABEL CACHRX IN THE VECTOR
* LOCATION.
* COMMENTS:
* SUBORDINATE ROUTINES CALLED: NONE

CACHRX::SAVE
JSR R5,PREG05 ;CALL REGISTER SAVE SUBR.

MOV RXINTC,R1 ;GET THE RECEIVER INTERRUPT COUNT
INC R1 ;INCREMENT THE COUNT
BVC 21 ;BRANCH IF NO OVERFLOW OCCURRED
DEC R1 ;RESET THE COUNT TO 17777

21: MOV R1,RXINTC ;SAVE NEW COUNT VALUE
601: PASS
JSR PC,8(SP) ;RESTORE GPRS.

RTI ;RETURN TO PREG05 SUBR.
.SBITL INTERRUPT SERVICE ROUTINE

**CATCH TRANSMITTER INTERRUPT.**

**THIS ROUTINE IS USED IN SEVERAL TESTS TO LOG A COUNT OF THE**
**NUMBER OF TRANSMISSION INTERRUPTS THAT OCCUR.**

**INPUTS:**
CSRA CONTAINS THE ADDRESS OF THE CSR.
TXINTC HOLDS THE COUNT OF THE NUMBER OF TX INTERRUPTS THAT OCCURRED.

**OUTPUTS:**
TXINTC CONTAINS THE UPDATED INTERRUPT COUNT.

**CALLING SEQUENCE:**
PUT THE ADDRESS OF THE LABEL CACHTX IN THE VECTOR LOCATION.

**COMMENTS:**

**SUBORDINATE ROUTINES CALLED:**
NONE

CACHTX::SAVE

;SAVE CONTENTS OF GPRS RO THRU RS.
JSR R5,PREG05 ;CALL REGISTER SAVE SUBR.

MOV TXINTC,R1 ;GET THE TRANSMISSION INTERRUPT COUNT
INC R1 ;INCREMENT THE COUNT
BVC 2$ ;BRANCH IF NO OVERFLOW OCCURRED
DEC R1 ;RESET THE COUNT TO 177777
MOV R1,TXINTC ;SAVE NEW COUNT VALUE
PASS ;RESTORE GPRS.
JSR PC,(SP),* ;RETURN TO PREG05 SUBR.

RTI
// INTERRUPT SERVICE ROUTINE

; SIMPLIFY INTERRUPT SERVICE ROUTINE - CLKINT -

; \* THIS ROUTINE IS EXECUTED CLKHz TIMES PER SECOND. IT DECREMENTS THE
\* TWO TIMER COUNTERS DOWN TO ZERO.

\* INPUTS: \* TIMER1 - TIMER COUNTER 1.
\* \* TIMER2 - TIMER COUNTER 2.
\* \* TIMER3 - TIMER COUNTER FOR CALL OF BREAK MACRO.

\* OUTPUTS: \* THE 2 TIMER COUNTERS ARE DECREMENTED IF THEY ARE NOT ZERO.

\* CALLING SEQUENCE: \* PUT OCLKINT IN THE CLOCK INTERRUPT VECTOR SLOT.
\* \* PUT THE DESIRED TIME PERIOD (SECONDS TIMES CLKHz) IN
\* \* EITHER TIMER1 OR TIMER2 AND POLL THE RESPECTIVE TIMER
\* \* COUNTER TO DETECT ITS GOING TO 0 ON TIME OUT.

\* COMMENTS: \* THE 2 COUNTERS WILL NOT WRAPAROUND BUT WILL STOP AT 0. THIS
\* \* ALLOWS THE DETECTION OF A TIME-OUT ANY TIME AFTER THE TIME-OUT
\* \* HAS OCCURRED UNTIL THE TIMER COUNTER IS SET TO ANOTHER VALUE.

\* \* SUBORDINATE ROUTINES CALLED: NONE.

; ********************************************

CLKINT: \* TST TIMER1 ;CHECK FOR TIMER1 AT ZERO.
        023706 005767 156432
        BEQ 24 ;BRANCH TO LEAVE IT AT ZERO IF IT IS ZERO.
        DEC TIMER1 ;DECREMENT TIME COUNT.
        24: TST TIMER2 ;CHECK FOR TIMER2 AT ZERO.
        BEQ 44 ;BRANCH TO LEAVE IT ALONE IF IT'S ALREADY ZERO.
        DEC TIMER2 ;DECREMENT TIME COUNT.
        DEC TIMER3 ;DECREMENT THE BREAK COUNT.
        44: DEC TIMERO ;SET UP TIME TILL NEXT BREAK.
        MOV BCOUNT,TIMER3 ;SAVE CONTENTS OF RO FROM BREAK MACRO.
        MOV RO, (SP) ;CHECK FOR OPERATOR CONTROL/.
        BREAK ;TRAP CIBRK
        MOV (SP)*,RO ;RESTORE CONTENTS OF RO.
        RTI ;

4366 023706 005767 156432
4367 023712 001402
4368 023714 005367 156424
4369 023720 005767 156422
4370 023724 001402
4371 023726 005367 156414
4372 023732 005367 156412
4373 023736 001006
4374 023740 016767 156406 156402
4375 023746 010046
4376 023750
4377 023750 104422
4378 023752 012600
4379 023754 000000
4380 023800 000000
4381 023804 001402
4382 023806 005367 156424
4383 023812 005767 156422
4384 023814 001402
4385 023816 005367 156414
4386 023822 005367 156412
4387 023832 001006
4388 023836 016767 156406 156402
4389 023842 010046
4390 023846
**SBR1L INTERRUPT SERVICE ROUTINE**

- **BR IST**: RECEIVE INTERRUPT SERVICE ROUTINE
- **BR LE**: TEST. THIS ROUTINE handles receive interrupts during the interrupt
- **BR LE**: COUNTS the interrupt AND sets a flag
- **BR LE**: TO indicate that the interrupt has occurred. It also checks the
- **BR LE**: FLAG WHICH INDICATES THAT A TX INTERRUPT HAS OCCURRED. IF THE TX
- **BR LE**: INTERRUPT FLAG IS SET, THIS ROUTINE SETS AN INTERRUPT ORDER ERROR
- **BR LE**: FLAG INDICATING THAT A TRANSMIT INTERRUPT WAS SERVICED BEFORE A
- **BR LE**: SIMULTANEOUS RECEIVE INTERRUPT.

**INPUTS:**
- RXINTC - Holds the count of the number of RX interrupts.
- RXINF - RX interrupt flags.

**OUTPUTS:**
- RXINTC - Contains the updated interrupt count.
- RXINF - RX interrupt flags:
  (BIT O SET, BIT 14 SET IF TXINTF BIT 0 IS SET.)

**CALLING SEQUENCE:**
PUT THE ADDRESS OF THE LABEL RXBR1T IN THE VECTOR
LOCATION.

**COMMENTS:**
NOTE: THE FIFO IS NOT PURGED BY THIS ROUTINE.

**SUBORDINATE ROUTINES CALLED:**
NONE.

---

**RXBR1T**: SAVE (SAVE CONTENTS OF GPRS RO THRU RS.
AS PREGOS) JSR (CALL REGISTER SAVE SUBRT).

- MOV @BRAF, R0
- READ THE CHAR OUT OF THE FIFO.
- MOV RXINTC, R1
- GET THE INTERRUPT COUNT.
- INC R1
- INCREMENT THE COUNT.
- BEQ 21
- BYPASS UPDATING COUNT IF OVERFLOW OCCURRED.
- MOV R1, RXINTC
- SAVE NEW COUNT VALUE.

**RXBR1T**: 21: MOVE RXINF, R1
- GET THE RX INTERRUPT FLAGS.
- BIS #BITO, R1
- SET THE RX INTERRUPT HAS OCCURRED FLAG.
- BIT #BIT0, TXINTF
- TEST THE "TX INT HAS OCCURRED" FLAG.
- BEQ 41
- SKIP SETTING ERROR FLAG IF NO TX INT.
- BIS #BIT14, R1
- SET THE INTERRUPT ORDER ERROR FLAG.
- MOV R1, RXINF
- UPDATE THE RX INTERRUPT FLAGS.
- RESTORE GPRS.
- JSR PC, @ (SP)
- RETURN TO PREGOS SUBRT.

**RXBR1T**: 41: PASS

**RXBR1T**: 601: RTI
**INTERRUPT SERVICE ROUTINE**

**RXINTP**

---

**RECEIVE CHARACTER INPUT INTERRUPT SERVICE ROUTINE**

**FUNCTION:**

This service routine inputs a character from the DUT and loads the
char (complete with status flags) into a receive char buffer in
memory. The interrupt is also counted. The receive char buffer is
monitored to ensure that it does not overflow.

**INPUTS:**
- BUFFEND - Labels the end of the host memory buffer.
- BUFSBR - Contains address of next free buffer location.
- CSRA - Contains the address of the DUT CSR.
- RBUF - Contains the address of the RBUF DUT register.
- RXINC - Holds the count of the number of RX interrupts.
- RXINTF - RX interrupt flags.

**OUTPUTS:**
- BUFSBR - Contains updated address of next free buffer location.
- RXINC - Contains the updated interrupt count.
- RXINTF - RX INT flags (bit 15 set if RX.DATA.Avail is clear).

**CALLING SEQUENCE:**
- PUT THE ADDRESS OF THE LABEL RXINTP IN THE VECTOR LOCATION.

**COMMENTS:**
- IN CASE OF OVERFLOW OF THE MEMORY BUFFER, BUFSBR WILL BE
  MAINTAINED EQUAL TO BUFFEND AND THE WORD AT BUFSBR WILL BE
  THE LAST WORD READ FROM THE DUT FIFO.
- NOTE: THIS ROUTINE CAN DESTROY TX.ACTIONS BY READING THE CSR.

**SUBORDINATE ROUTINES CALLED:**
- NONE.

---

**RXINTP:**

SAVE

SAVE CONTENTS OF GPRS RO THRU R5.

JSR R5,PREGOS (CALL REGISTER SAVE SUBRT.

MOV @CSRA,R1 ; READ THE CONTENTS OF THE CSR.

BIT @BIT7,R1 ; TEST RX.DATA.AVAIL BIT.

BNE 2$ ; BRANCH AROUND SETTING FLAG IF BIT IS SET.

BIS @BIT15,RXINTF ; SET THE RX.DATA.AVAIL CLEAR FLAG.

2$: MOV RXINC,R1 ; GET THE INTERRUPT COUNT.

INC R1 ; INCREMENT THE COUNT.

4$: MOV R1,RXINTC ; BYPASS UPDATING COUNT IF OVERFLOW OCCURRED.

BEG 4$ ; SAVE NEW COUNT VALUE.

MOV R4,RXINTC ; GET THE POINTER TO NEXT FREE BUFFER WORD.

CMP R4,BUFFEND ; TEST FOR POINTER BEYOND END OF BUFFER.

BHS 60$ ; SKIP THE PTR UPDATE IF PTR OUT OF MEMORY.

MOV R2,BUFFPR ; UPDATE THE BUFFER POINTER.

60$: PASS ; RESTORE GPRS.

ATI

JSR PREGOS ; RETURN TO PREGOS SUBRT.
**GLOBAL TRAP SERVICE ROUTINE**

*SBTIL GLOBAL TRAP SERVICE ROUTINE TPARTN -

**BUS TIME-OUT TRAP (004 TRAP) SERVICE ROUTINE**

1. THIS ROUTINE DETERMINES IF THE 004 TRAP WAS CAUSED BY
   AN "EXPECTED" ERROR OR NOT BY EXAMINING THE RETURN PC VALUE ON THE
   STACK. IF THE TRAP IS UNEXPECTED, THIS ROUTINE JUMPS TO THE NORMAL
   DIAGNOSTIC SUPERVISOR 004 TRAP HANDLING ROUTINE.

**INPUTS:**
- SP - POINTS TO THE PC WHERE THE TRAP OCCURRED.
- ADRPTR - LABEL AT THE ADDRESS WHERE "EXPECTED" TRAPS OCCUR.
- TP4FLG - 004 TRAP FLAGS.

**OUTPUTS:**
- TP4FLG - BIT 15 IS SET IF "EXPECTED" TRAP OCCURRED.

**CALLING SEQUENCE:**
- PUT ADDRESS POINTED TO BY TPARTN IN 004 VECTOR.
- OCCURRENCE OF 004 TRAP VECTORS TO THIS ROUTINE.

**COMMENTS:**
- ANY 004 TRAP WHICH OCCURS AT AN ADDRESS OTHER THAN THAT LABELED
- ADRPTR WILL BE HANDLED BY THE NORMAL 004 TRAP SERVICE ROUTINE.

**SUBORDINATE ROUTINES CALLED:**
- NONE

**LOCATION:**
- TP4RTN:

  **Larger Memory:**
  - CMP (SP),#ADR PTR
  - *COMPARE EXPECTED ADDR AGAINST TRAP RET PC.*
  - BEQ 2:
  - *IF THEY MATCH, CONTINUE THIS ROUTINE.*

  **Spool Memory:**
  - JMP @TP4VEC
  - *IF NOT, JUMP TO NORMAL 004 TRAP SERVICE RTN.*

  **Bits:**
  - BIS @BIT15,TP4FLG
  - *SET THE 004 TRAP OCCURED FLAG.*
  - RTI
  - *ALL DONE, GO BACK TO THE TEST.*
**SBTL** INTERRUPT SERVICE ROUTINE - TXINTR

---

* TRANSMIT INTERRUPT SERVICE ROUTINE.
* THIS ROUTINE HANDLES A TRANSMIT INTERRUPT FROM THE DEVICE UNDER TEST
  (OUT) BY COUNTING THE INTERRUPT AND READING THE OUT CSR TO CLEAR THE
  INTERRUPT REQUEST. THIS ROUTINE ALSO SETS A FLAG TO INDICATE THAT
  A TX INTERRUPT HAS OCCURRED AND SETS A FLAG IF THE TX.ACTION BIT IS
  NOT SET IN THE READ CONTENTS OF THE OUT CSR.

* INPUTS:
  * CSRA - CONTAINS THE ADDRESS OF THE CSR.
  * TXINTC - MOLDS THE COUNT OF THE NUMBER OF TX INTERRUPTS.
  * TXINF - TX INTERRUPT FLAGS.

* OUTPUTS:
  * TXINTC - CONTAINS THE UPDATED TX INTERRUPT COUNT.
  * TXINF - TX INT FLAGS (BIT 0 SET, BIT 15 SET IF TX.ACTION CLR).

* CALLING SEQUENCE:
  * PUT THE ADDRESS OF THE LABEL TXINTR IN THE VECTOR LOCATION.

* COMMENTS:

* SUBORDINATE ROUTINES CALLED: NONE

---

TXINTR: SAVE

JSA  R5, PREGOS  ; CALL REGISTER SAVE SUBRT.

MOV  TXINTC, A1  ; GET THE TX INTERRUPT COUNT.
INC  R1  ; INCREMENT THE COUNT.
BVC  21  ; BRANCH IF NO OVERFLOW OCCURRED.
DEC  R1  ; RESET THE COUNT TO INITIALIZED.

21: MOV  R1, TXINTC  ; SAVE NEW COUNT VALUE.
MOV  TXINF, P3  ; GET THE TX INTERRUPT FLAGS.
MOV  @CSRA, R2  ; READ THE CSR.
BMI  41  ; SKIP SETTING OF FLAG IF TX.ACTION IS SET.

41: BIS  #BIT15, R3  ; SET THE TX.ACTION CLEAR FLAG.
BIS  #BIT0, R3  ; SET THE TX INT HAS OCCURRED FLAG.
MOV  R3, TXINF  ; UPDATE THE TX INTERRUPT FLAGS.

601: PSS  ; RESTORE GPRS.

RTI  ; RETURN TO PREGOS SUBRT.
REPORT CODING SECTION

; THE REPORT CODING SECTION CONTAINS THE
; 'PRINTS' CALLS THAT GENERATE STATISTICAL REPORTS.

BGHRPT

EXIT RPT

.EVEN

ENDORPT

L10017:

TRAP CIRPT
.SBTIL PROTECTION TABLE

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

BGNPROT

<table>
<thead>
<tr>
<th>OFFSET INTO P TABLE FOR CSR ADDRESS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>177777</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFFSET INTO P-TABLE FOR MASSBUS ADDRESS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>177777</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFFSET INTO P TABLE FOR DRIVE NUMBER</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>177777</td>
</tr>
</tbody>
</table>

ENDPROT
SBRTL  INITIALIZE SECTION

* THIS SECTION CONTAINS THE CODE WHICH IS PERFORMED AT THE BEGINNING OF
* EACH PASS OR AFTER A CONTINUE COMMAND.
* THIS CODE PERFORMS THE FOLLOWING ACTIONS:
* MOVES THE INFORMATION HELD IN THE HARDWARE P- TABLE INTO THE GLOBAL
* DATA AREA.

```
        BGNINIT
        L$INIT:
        ;SEE IF PROGRAM JUST STARTED, BR IF YES
        READEF @EF.START
        MOV @EF.START,RO
        TRAP CIREFG
        BCOMPLETE NEWSTA
        BCS NEWSTA

        ;SEE IF PROGRAM JUST RESTARTED, BR IF YES
        READEF @EF.RESTART
        MOV @EF.RESTART,RO
        TRAP CIREFG
        BCOMPLETE NEWRES
        BCS NEWRES

        ;SEE IF THIS IS A NEW PASS, BR IF YES
        READEF @EF.NEW
        MOV @EF.NEW,RO
        TRAP CIREFG
        BCOMPLETE NEWPAS
        BCS NEWPAS

        ;SEE IF PROGRAM WAS JUST CONTINUED
        READEF @EF.CONTINUE
        MOV @EF.CONTINUE,RO
        TRAP CIREFG
        BCOMPLETE GETPRM
        BCC GETPRM

        NEWSTA:
        BRESET
        ;RESET THE BUS TO PREVENT ILLEGAL INTERRUPTS.
        TRAP CIRESET

        ;SET UP FOR LINE TIME CLOCK INTERRUPTS.
        CLOCK L,RI
        ;GET THE CLOCK PARAMETERS.
        MOV @:L,RO
        TRAP CICLK
        MOV RO,RI

        MOV (R1)+,CLKCSR
        ;STORE CLOCK CSR ADDRESS.
        MOV (R1)+,CLKBRL
        ;STORE CLOCK BUS REQ INT LEVEL.
        MOV (R1)+,CLKVEC
        ;STORE CLOCK INTERRUPT VECTOR.
        MOV (R1)+,CLKHIZ
        ;STORE CLOCK FREQUENCY.
        CMP CLKHRZ,050.
        ;TEST FOR 50HZ LINE FREQUENCY.
        BNE 21
        ;BRANCH IF CLOCK IS NOT 50HZ.
```
INITIALIZATION

4669 024344 012767 000024 156002 MOV 020,MSTICK ;INDICATE 20MS PER CLOCK TICK.
4670 024352 000043 BR 41 ;
4671 024354 012767 000021 155772 21 MOV 017,MSTICK ;INDICATE 17 MS PER CLOCK TICK.
4672 024362 024362 012746 000030 243706 MOV @PR06,SP ;INITIALIZE CLOCK INTERRUPT VECTOR.
4673 024366 024372 012746 155742 243706 MOV CLKVEC,SP ;
4674 024376 012746 000003 244002 MOV 03,(SP) ;INITIALIZE THE BREAK COUNT
4675 024402 104437 ASL RO ;TO CAUSE A BREAK
4676 024404 062706 000010 244422 MOV RH,RO,BCOUNT ;EVERY 2 SECONDS
4677 024422 012700 000240 024426 104441 SETPRI #PRI05 ;ALLOW CLOCK INTERRUPTS DISABLE OTHERS.
4678 024440 062706 000010 ;MOV 00H,(SP)
4679 024444 016767 153350 155664 024450 012767 MOV 4,TP4VEC ;SAVE THE EXISTING 004 TRAP VECTOR.
4680 024456 024124 153340 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4681 024462 016767 153350 155664 MOV 4,TP4VEC ;SAVE THE EXISTING 004 TRAP VECTOR.
4682 024472 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4683 024482 012767 024124 153340 MOV 4,TP4VEC ;SAVE THE EXISTING 004 TRAP VECTOR.
4684 024484 016767 153350 155664 CLR TP4FLG ;CLEAR THE 004 TRAP FLAG.
4685 024490 012767 024124 153340 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4686 024496 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4687 024498 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4688 024490 012767 024124 153340 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4689 024496 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4690 024498 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4691 024490 012767 024124 153340 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4692 024496 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4693 024498 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4694 024500 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4695 024502 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4696 024504 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4697 024506 016767 153350 155664 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4698 024510 000402 173012 024514 016767 MOV 4,TP4VEC ;SAVE THE EXISTING 004 TRAP VECTOR.
4699 024522 012767 024124 153340 MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
4700 024530 005067 155676 024534 005067 CLR TP4FLG ;CLEAR THE 004 TRAP FLAG.
4701 024538 005067 155676 024542 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4702 024546 005067 155676 024550 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4703 024554 005067 155676 024558 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4704 024562 005067 155676 024566 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4705 024570 005067 155676 024574 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4706 024578 005067 155676 024582 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4707 024586 005067 155676 024590 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4708 024594 005067 155676 024598 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4709 024592 005067 155676 024600 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4710 024596 005067 155676 024602 005067 CLR WORD1 ;PREPARE TO CLEAR THE MEM MGT SRO REGISTER.
4711 024598 005067 155676 024604 005067 CLR MMTAB ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
4712 024600 005067 155676 024604 005067 CLR MMTAB ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
4713 024602 005067 155676 024604 005067 CLR MMTAB ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
4714 024604 005067 155676 024604 005067 CLR MMTAB ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
4715 024606 005067 155676 024606 005067 CLR MMTAB ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
4716 024608 005067 155676 024608 005067 CLR MMTAB ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
4717 024608 005067 155676 024608 005067 CLR MMTAB ;INDICATE NO MEM MGT PRESENT IN CASE IT ISN'T.
NEWRES: BRESET ;RESET THE BUS TO PREVENT ILLEGAL INTERRUPTS.
       TRAP CIHEJET

NEWPA: ;CLR COUNTER USED IN REPORTING ROM VERSION.
       MOV @ 1,UNITN ;RESET LOGICAL DEVICE TO 1

;INCREMENT THE PASS COUNTER, CORRECT FOR ANY OVERFLOW.
;THIS COUNTER IS USED IN THE ROM VERSION TEST.

INC PASCNT ;INCREMENT THE PASS COUNTER.
BNE GEIPRM ;BRANCH IF WE HAVE NOT YET OVERFLOWED.
DEC PASCNT ;SET PASS COUNT TO 17777 OCTAL.

;GET THE HARDWARE PARAMETERS FOR THIS UNIT.
GETPRM:

INC UNITN ;INCREMENT LOGICAL DEVICE NUMBER
CMP UNITN,UNIT ;SEE IF MAXIMUM UNIT NO. EXCEEDED
BGE NEWPAS ;BR IF YES

GETHARD UNITN,R1 ;GET P TABLE POINTER INTO R1
        MOV UNITN,PO
TRAP CIOPHRU
        MOV RO,R1

BCOMPLETE 301 ;BR IF DEVICE AVAILABLE
        BCS 301

;*************** HARDWARE PARAMETER MOVING CODE ***************

;STORE DMU-11 CSR ADDRESS IN DEV.REG.ADDRESS TABLE
MOV (R1),CSRA
301:

;GET THE RX INTERRUPT VECTOR ADDRESS.
MOV (R1),R2

;STORE RX INT VECTOR ADDRESS.
MOV R2,RXVECA
ADD #8,R2 ;CALCULATE TX INTERRUPT VECTOR ADDRESS.
MOV R2,TXVECA

;STORE TX INT VECTOR ADDRESS.
MOV (R1),ACILNS ;STORE DMU-11 ACTIVE LINE BIT MAP
MOV (R1),B4LEVL ;STORE DMU 11 INTERRUPT BUS REQUEST LEVEL

;CALCULATE DEVICE REGISTER ADDRESSES, AND PUT THEM IN THE
;DEVICE REGISTER ADDRESS TABLE.

;COPY CSR ADDRESS
MOV CSRA,R1
INC R1
MOV R1 ;INCREMENT CSR ADDRESS
INC R1 ;COPY BY 2.

;SET UP REGISTER COUNTER
MOV #7,R3
MOV @RELV,A,R2
MOV @RELF,A,R2
MOV @RBUF,A,R2
MOV @LRBUF,A,R2
MOV @LRBUF,A,R2
MOV @LOC,64
MOV 85,65

;LOCATION WHERE RBUF ADDRES GOES IN TABLE
MOV R1,(R2) ;STORE REGISTER ADDRESS IN TABLE
INC R1 ;INCREMENT REGISTER ADDRESS
INC R1 ;BY 2, FOR THE NEXT DEVICE REGISTER.
DEC R3 ;DECREMENT REGISTER COUNT
BNE 121 ;LOOP IF NOT DONE

;INITIALISE THE BMP CODE QUEUE.

MOV @BMPQDB,R0 ;GET THE START ADDRESS OF THE QUEUE.
MOV @BMPQDR,A1 ;GET THE END ADDRESS OF THE QUEUE.
; SET THE POINTER TO THE START OF THE QUEUE.
; CLEAR OUT THE CONTENTS OF THE QUEUE.
; CHECK IF END OF QUEUE HAS BEEN REACHED.
; LOOP IF NOT ALL DONE.
; REPORT THE UNIT NUMBER IF THE SOFTWARE P-TABLE QUESTION WAS ANSWERED YES.
; AND THE MAXIMUM UNIT NUMBER IS GREATER THAN 1.
; CHECK IF THE QUESTION WAS ANSWERED YES.
; SKIP REPORTING UNIT NUMBER IF IT IS DISABLED.
; CHECK MAXIMUM NUMBER OF UNITS SELECTED.
; DO NOT REPORT UNIT NUMBER IF MAX NUMBER < 1.
; REPORT UNIT NUMBER.
; CLR THE CTRL C TEST ABORT FLAG.
; SET THE PROCESSOR PRIORITY TO DISABLE ALL INTERRUPTS.
; SET PROCESSOR PRIORITY TO 7.
; INITIALIZE THE ASSEMBLER TEST NUMBER VARIABLE.
.SCTL AUTODROP SECTION

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

BGNAUTO

L1AUTO:

ENDAUTO

L10022:

TRAP CS_AUTO
SBITL  CLEANUP CODING SECTION

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

; BGNCLN

L$CLEAN::

TST CTRLCF
BEQ  2$i

CTRL-C FROM TEST? NO, SKIP BUS RESET.

TRAP C$IRESET

2$i:

EXIT CLN

TRAP C$IEXIT

.EVEN

ENDCLN

L10023:: TRAP C$ICLEAN
.SBITL DROP UNIT SECTION

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

; -

BEGIN

PRINT @DROP,RO ; REPORT UNIT THAT HAS BEEN DROPPED.
MOV RO,-(SP)
MOV @DROP,-(SP)
MOV @2,(SP)
MOV SP,RO
TRAP SINTF
"U0 @6,SP

BR EDROP ; BRANCH AROUND THE MESSAGE.

DROP: .ASCIZ "A UNIT DROPPED FROM FURTHER TESTING."

EDROP: .EVEN
EXIT DU

; .WORD J8JMP
.; .WORD L10024-2-.

ENDDU

L10024: TRAP SINTF
**ADD UNIT SECTION**

`; The ADD UNIT SECTION contains any code the programmer wishes
`; to be executed in conjunction with the adding of a unit back
`; to the test cycle.`

```
BGNAU

; Insert ADD code here. This code will be executed after
; an "ADD" command. The purpose of this code is to do any
; housekeeping that may be necessary after a unit has been added.
; This section is optional.
```

EXIT AU

.EVEN

ENDAU

L10025: TRAP C#AU
**HARDWARE TEST**

- **REGISTER ADDRESS TEST**

  This test verifies that the device registers will respond to the proper unibus handshaking signals when accessed, if the DPU11 does not respond to the access attempts (if the DPU11 is at the wrong address, for example). The 004 bus time-out trap is detected by this routine and an error is reported. This test is performed on line 0 only.

```assembly
BGNTST
T1: ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
000001

MOV @TNUM,TSTNUM ;SET THE TEST NUMBER.
0176777 155056
MOV $1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
155102
MOV $0101,ERRNBR ;SET THE TEST ERROR NUMBER IN THE TABLE.
156532
MOV @EMO103,ERRMSG ;SET UP THE TEST FAILURE MESSAGE IN THE TABLE.
156526
MOV @ERO101,ERRBLK ;SET UP THE ERROR ROUTINE IN THE ERROR TABLE.
156522

; SET UP TO CATCH ANY 004 TRAPS WHICH OCCUR:

MOV 4,TP4VEC ;SAVE THE EXISTING 004 TRAP VECTOR.
016767 152526 155042
MOV @TP4RTN,4 ;SET 004 TRAP VECTOR TO OUR SERVICE RTN ADR.
024124 152516
CLR R5 ;CLEAR THE ERROR FLAGS.
005005

; HERE BEGINS THE LOOP TO TEST THE REGISTERS FOR A LINE.

; FIRST TEST THE CSR AND SET THE IND.ADR.REG (I.A.R) FIELD.

MOV CSRA,R0 ;SET UP CSR AS THE CKTRAP MOVE SOURCE.
016700 154746
MOV #25,CKTRAP ;SET UP DESTINATION LOCATION FOR CKTRAP MOVE.
025466
JSR PC,CKTRAP ;MOVE AND CHECK FOR TRAP.
017246
BCS $41 ;IF NO TRAP, BYPASS ERROR.
103402
BIS $000001,R5 ;SET FATAL READ ERROR FLAGS.
100001
BIC $17,52$ ;CLEAR THE I.A.R FIELD OF THE CSR DATA.
000146
MOV R1,R0 ;USE OLD DESTINATION FOR SOURCE OF CKTRAP MOVE.
154714
MOV CSRA,R1 ;SET UP CSR AS THE CKTRAP MOVE DESTINATION.
157114
JSR PC,CKTRAP ;MOVE AND CHECK FOR TRAP.
172420
BCS $61 ;IF NO TRAP, BYPASS ERROR.
103403
BIS $000002,R5 ;SET FATAL WRITE ERROR FLAGS.
100002
BR $40$ ;EXIT AND REPORT FATAL ERROR.
000434

; NOW, WE TEST EACH REGISTER FOR THIS LINE.

MOV $8,R2 ;INIT REGISTER COUNTER TO 8.
012702 000010
MOV $8,R1 ;INITIALIZE THE REGISTER POINTER.
016767 154670 000110
MOV CSRA,R0 ;SET UP REGISTER AS THE SOURCE FOR CKTRAP MOVE.
025466
JSR #52,CKTRAP ;SET UP LOCAL STORAGE AS THE DES FOR CKTRAP.
017262
JSR PC,CKTRAP ;PERFORM THE MOVE. CHECK FOR TRAP.
172632
BCS $101$ ;IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.
10001
BIS $000001,R5 ;SET FATAL READ ERROR FLAGS.
100001
MOV R1,R0 ;USE OLD DEST AS SRC FOR CKTRAP MOVE.
10100
MOV R0,R1 ;SET UP REGISTER AS THE DEST FOR CKTRAP MOVE.
PERFORM THE MOVE. CHECK FOR TRAP.

IF NO TRAP, BYPASS THE SETTING OF ERROR FLAGS.

SET FATAL WRITE ERROR FLAGS.

INCREMENT THE REGISTER.

INCREMENT BY 2.

COUNT THE REGISTER.

LOOP TO TEST THE NEXT REGISTER ADDRESS.

DONE CHECKING DEVICE REGISTER ADDRESSES.

REPORT ANY ERRORS AND EXIT.

REPORT "DEVICE REGISTER ACCESS TEST FAILED"

ERROR

RESTORE THE NORMAL 004 TRAP VECTOR.

CHECK THE ERROR FLAGS.

EXIT ROUTINE IF NO ERRORS.

REPORT "DEVICE REGISTER ACCESS TEST FAILED"

DROP THIS UNIT FROM FUTURE TESTING.

INDICATE NO CTRL-C ABORT FROM TEST.

ABORT THIS SUB PASS.

************************** LOCAL STORAGE. **************************

STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.

STORAGE FOR THE SOURCE OR DEST OF THE CKTRAP MOVE.

END **************************** END *****************************

INDICATE THAT WE ARE NOT WITHIN A TEST.

L10026: TRAP C$ETST
.*SBTL HARDWARE TEST *.MRSTA

**MASTER RESET WITH SELFTEST TEST**

* THIS TEST VERIFIES THAT THE MASTER RESET BIT WILL CLEAR AFTER A DEVICE
  **RESET AND THE PERFORMANCE OF THE DUT ROM BASED SELFTEST.**

**SBTL**

BGNST

TNUM = TNUM + 1

INCREMENT ASSEMBLY TIME TEST COUNTER.

MOV #TNUM,ISTNUM

SET UP THE TEST NUMBER. (2)

MOV @1,CALCF

INDICATE THAT WE ARE IN A TEST.

SETPR @PR05

ALLOW LTC INTERRUPTS.

MOV @PR05,RO

TRAP C$PR05

MOV #1,ERRTYP

SET ERROR TYPE AS FATAL IN ERROR TABLE.

MOV #2001,ERMSG

SET ERROR MESSAGE ADDRESS IN ERROR TABLE.

MOV #002001,ERRBLK

SET ERROR ROUTINE ADDRESS IN ERROR TABLE.

W A I T U P T O S ECONDS FOR THE OUT MASTER RESET BIT TO CLEAR.

MOV #0000,R1

TIME-OUT VALUE IS 5.0 SECONDS.

MOV #BITS0,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,MLGET

WAIT FOR DUT CSR MR BIT TO CLEAR.

BCS 21

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,SKPSYS

TRY TO SKIP THE SELFTEST.

MOV #0000,R1

TIME-OUT VALUE IS 5.0 SECONDS.

MOV #5000,R1

WAIT FOR DUT CSR MR BIT TO CLEAR.

BCC 41

GO REPORT ERROR IF MR BIT DID NOT CLEAR.

SET THE MASTER RESET BIT AND VERIFY THAT IT CLEAR WITHIN THE PROPER TIME.

MOV #5000,R1

TIME-OUT VALUE IS 5.0 SECONDS.

MOV R2,(R4)

SET THE DUT MASTER RESET BIT.

JSR PC,MLGET

WAIT FOR DUT CSR MR BIT TO CLEAR.

BCC 41

GO REPORT ERROR IF MR BIT DID NOT CLEAR.

MOV #5000,R2

CALCULATE # OF MS FOR MR TO CLEAR.

SUB R1,R2

GO REPORT ERROR IF MR CLEAR IMMEDIATELY.

BEQ 41

CMP R2,#500.

BLT 81

GO REPORT ERROR IF MR CLEAR IN < 1/2 SECOND.

BR 601

EXIT THE TEST WITHOUT ERROR.

ERROR REPORTS:

REPORT MR BIT WOULD NOT CLEAR AFTER A DUT RESET.

MOV #002001,ERMSR

SET THE ERROR NUMBER IN ERROR TABLE.

MOV #002002,R1

SELECT ERROR MESSAGE.

ERROR

REPORT ERROR.
025654 104460  BR  60$:  EXIT THE TEST.
025656 000415
025660 012767 000312 156102 64$: MOV 0202, ERR2BR  Set the error number in Error Table.
025666 012701 006024  MOV @EM0203,R1  Select error message.
025672 ERROR  REPORT ERROR. >>>>> ERROR 0202 <<<<<
025674 104460 TRAP CIERROR
025676 000406  BR  60$:  EXIT THE TEST.
025678
025676 012767 000313 156064 64$: MOV 0203, ERR2BR  Set the error number in Error Table.
025704 012701 006167  MOV @EM0204,R1  Select error message.
025710 ERROR  REPORT ERROR. >>>>> ERROR 0203 <<<<<
025710 104460 TRAP CIERROR
025712 60$: SETPRI @PRI07  Disable all interrupts.
025712 012700 000340  MOV @PRI07,RO
025716 104441  TRAP CI$PRI
025720 005067 154362  CLR CTRLCF  Indicate that we completed the test.
025724 ENDS1  L10027:
025724 104401 TRAP CI$ETS1
.SBTL  Hardware Test  M855A

** 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 out ROM based selftest.

**----------------------------------------**

BGNST

T3:  TNUM = TNUM + 1  ; Increment the assembly time test counter.

MOV #TNUM, TSTNUM  ; Set up the test number.

MOV $1, CTRLDF  ; Indicate that we are in a test.

MOV $1, ERRTYP  ; Set error type as fatal in error table.

MOV #EM0301, ERMSG  ; Set error message address in error table.

MOV #ERO201, ERBLK  ; Set error routine address in error table.

** Wait up to 5 seconds for the out master reset bit to clear. **

MOV #5000, R1  ; Time-out value is 5.0 seconds.

MOV #BIT05, R2  ; Waiting for master reset bit.

CLPA R3  ; Waiting for bit to clear.

MOV CSR, R4  ; Bit is in the out CSR.

JSR PC, MSLGET  ; Wait for out CSR MR BIT TO CLEAR.

BCS 2;  ; Skip to reset out if MR clear.

** Out master reset bit did not go clear. Device may be stuck in some **
** odd state. Try to reset device with a bus reset. **

@RESET  ; No, try to jog device with bus reset.

TRAP C@RESET  ; Try to skip the selftest.

JSR PC, SKPSTS  ; Try to skip the selftest.

MOV #5000, R1  ; Time-out value is 5.0 seconds.

JSR PC, MSLGET  ; Wait for out CSR MR BIT TO CLEAR.

BCC 6;  ; Go report error if MR 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;  ; Time-out value is 1/5 second.

MOV #200, R1  ; Set the out master reset bit.

JSR PC, SKPSTS  ; Try to skip the selftest.

JSR PC, MSLGET  ; Wait for out CSR MR BIT TO CLEAR.

BCC 4;  ; Go find out what is wrong if MR not clear.

MOV #200, R2  ; Calculate 0 of MS for MR to clear.

SUB R1, R2  ; R1, R2.

CMP R2, #10.  ; 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;  ; Time-out value is 5 seconds minus 1/5 second.

MOV #4800, R1  ; Waiting for out CSR MR BIT TO CLEAR.

JSR PC, MSLGET  ; Go report error if MR cleared finally.
ERROR REPORTS:

; REPORT MR BIT WOULD NOT CLEAR AFTER A DUT RESET.
5157 026106 012767 000455 155564 68: MOV @0301, ERNRBR ; SET THE ERROR NUMBER IN ERROR TABLE.
5162 026114 0127F1 005651 MOV @EM0202, R1 ; SELECT ERROR MESSAGE.
5163 026120 104460 ERROR ; REPORT ERROR.
5164 026122 000415 TRAP CIERROR ; EXIT THE TEST.

; REPORT MR BIT CLEAR WITHIN 10 MS AFTER DUT RESET.
5165 026124 012767 000456 155563 68: MOV @0302, ERNRBR ; SET THE ERROR NUMBER IN ERROR TABLE.
5166 026132 012701 006420 MOV @EM0302, R1 ; SELECT ERROR MESSAGE.
5167 026136 104460 ERROR ; REPORT ERROR.
5168 026140 000406 TRAP CIERROR ; EXIT THE TEST.

; REPORT MR CLEARED BETWEEN 1/5 SECOND AND 5 SECONDS OF DUT RESET.
5169 026142 012767 000457 155620 10$: MOV @0303, ERNRBR ; SET THE ERROR NUMBER IN ERROR TABLE.
5170 026150 012701 006560 MOV @EM0303, R1 ; SELECT ERROR MESSAGE.
5171 026154 104460 ERROR ; REPORT ERROR.
5172 026158 003400 TRAP CIERROR ; EXIT THE TEST.

; SETPRI @PRI07 ; DISABLE ALL INTERRUPTS.
5173 026156 012700 000340 MOV @PPRI07, RO
5174 026162 104441 TRAP CI$PR

; CLEAR CTRLCF ; INDICATE THAT WE COMPLETED THE TEST.
5175 026164 005067 154116 CLR CTRLCF
5176 026170 ENDTST

L10030: TRAP C#ETST
;** THIS TEST VERIFIES THAT THE RX CHARACTER FIELD OF THE DUT RBUF REGISTER
;** APPEARS TO BE FUNCTIONING CORRECTLY. THIS TEST USES THE CODES WHICH
;** SHOULD BE IN THE FIFO AFTER A BOARD RESET AND SKIP SELFTEST SEQUENCE.

;*****************************************************
;** RBUF REGISTER RX CHARACTER FIELD TEST
;*****************************************************

; Set the DUT CSR MASTER RESET (MR) BIT, PERFORM THE SKIP SELFTEST SEQUENCE,
; and wait up to 5 seconds for the MR BIT to clear.

BGNST

; Set the DUT CSR MASTER RESET (MR) BIT, PERFORM THE SKIP SELFTEST SEQUENCE,
; and wait up to 5 seconds for the MR BIT to clear.

MOV #5000,R1       ; TIME OUT VALUE IS 5.0 SECONDS.
MOV #BIT05,R2      ; WAITING FOR MASTER RESET BIT.
CLR R3             ; WAITING FOR BIT TO CLEAR.
MOV #RXA,R4        ; BIT IS IN THE DUT'S CSR.
MOV R2,(R4)         ; SET THE DUT MASTER RESET BIT.
JSR PC,SKPSTS      ; SKIP THE SELFTEST.
JSR PC,MSLGET      ; WAIT FOR OUT CSR MR BIT TO CLEAR.
BCC 43              ; GO REPORT ERROR IF MR DID NOT CLEAR.

; Read 6 characters from the DUT and verify that they are valid selftest codes.

MOV (R4)+,RO       ; INCREMENT POINTER TO POINT TO DUT RBUF REGISTR.
MOV #6,R1          ; INITIALIZE THE LOOP COUNTER.
21: MOV (R4),R2    ; READ A CHARACTER FROM THE DUT RBUF REGISTER.
MOV R2,RO          ; REMOVE ALL BUT BITS SPECIFIC TO SELFTEST CODE.
CMP R0,#201        ; CHECK THAT BITS 0, 6, AND 7 ARE CORRECT.
BNE 58              ; GO REPORT ERROR IF CODE IS NOT SELFTEST CODE.
DEC R1             ; COUNT THIS LOOP ITERATION.
BNE 21              ; LOOP IF NOT ALL LINES DONE.
BR 60$              ; EXIT TEST, NO ERROR FOUND.

; Error reports:

; REPORT MR BIT WOULD NOT CLEAR AFTER A DUT RESET.

MOV #0401,ERERROR  ; SET THE ERROR NUMBER IN ERROR TABLE.
MOV #EMO202,R1     ; SELECT ERROR MESSAGE.
ERROR              ; REPORT ERROR.
       ERROR #0401 error
TRAP CERROR
BR 60$              ; EXIT THE TEST.

; REPORT IMPROPER CODE FOUND IN DUT RBUF AFTER RESET (SKIP SELFTEST).
64: MOV #0402, ERRNBR
SET THE ERROR NUMBER IN ERROR TABLE.

5235 MOV #EM0402,A1
SELECT ERROR MESSAGE.

5236 ERROR
REPORT ERROR.

5237 

5238 MOV #0402 <<<<<

5239 TRAP C\ERROR

5240 601: SEIPRI 1\PRI07
DISABLE ALL INTERRUPTS.

5241 MOV #PRI07,RO

5242 TRAP C\SPRI

5243 CLR C\TALCF
INDICATE THAT WE COMPLETED THE TEST.

5244 ENDSTST

5245 L10031: TRAP C\ETST
`** **
** THIS TEST VERIFIES THAT THE FIELD OF 3 FLAG BITS IN THE RBUF READS AS ALL ONES WHEN THE SELFTEST CODES ARE BEING READ FROM THE DUT AFTER A BOARD RESET AND SKIP SELFTEST SEQUENCE. **

```
      BGNINST
      SETPRI @PRI05 ; ALLOW LTC INTERRUPTS.
      MOV @PRI05,R0
      TRAP C$SPI

      TNUM == TNUM + 1 ; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
      MOV @TNUM,TNUM ; SET UP THE TEST NUMBER.
      MOV 0-1,CIRLCF ; INDICATE THAT WE ARE WITHIN A TEST.
      MOV #1,ERRTYP ; SET ERROR TYPE AS FAIL IN ERROR TABLE.
      MOV #E0501,ERRMSG ; SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
      MOV #E0201,ERRBLK ; SET ERROR ROUTINE ADDRESS IN ERROR TABLE.

      ; SET THE DUT CSR MASTER RESET (MR) BIT, PERFORM THE SKIP SELFTEST SEQUENCE, AND WAIT UP TO 5 SECONDS FOR THE MR BIT TO CLEAR.
      MOV #5000,R1 ; TIME-OUT VALUE IS 5.0 SECONDS.
      MOV #BIT05,R2 ; WAITING FOR MASTER RESET BIT CLR R3 ; WAITING FOR BIT TO CLEAR.
      MOV CSRA,R4 ; BIT IS IN THE DUT'S CSR.
      MOV R2,(R4) ; SET THE DUT MASTER RESET BIT.
      JSR PC,SKPSTS ; SKIP THE SELFTEST.
      JSR PC,MSLET ; WAIT FOR DUT CSR MR BIT TO CLEAR.
      BCC 41 ; GO REPORT ERROR IF MR DID NOT CLEAR.

      ; READ 8 CHARACTERS FROM THE DUT AND VERIFY THAT ALL 3 RX ERROR FLAGS ARE SET FOR EACH CHARACTERS.
      MOV (R4)+,R0 ; INCREMENT POINTER TO POINT TO DUT RBUF REGISTR.
      MOV #8,R1 ; INITIALIZE THE LOOP COUNTER.
      MOV (R4),R2 ; READ A CHARACTER FROM THE DUT RBUF REGISTER.
      MOV #70000,R0 ; CALCULATE BIT MAP OF CLEAR RX ERROR FLAGS.
      BIC R2,R0 ; GO REPORT ERROR IF NOT ALL RX ERROR FLAGS SET.
      BNE 61 ; COUNT THIS LOOP ITERATION.
      DEC R1 ; LOOP IF NOT ALL LINES DONE.
      BNE 21 ; EXIT TEST, NO ERROR FOUND.
      BR 60 ; ERROR REPORTS:
      ; REPORT MR BIT WOULD NOT CLEAR AFTER A DUT RESET.
      MOV #E0501,ERRBR ; SET THE ERROR NUMBER IN ERROR TABLE.
      MOV #E0020,R1 ; SELECT ERROR MESSAGE.
      ERROR ; REPORT ERROR.
      ERROR ; REPORT ERROR.
      TRAP C$ERROR
      BR 60 ; EXIT THE TEST.
      ; REPORT ONE OR MORE RX ERROR FLAGS FOUND CLEAR WITH SELFTEST CODE.
      MOV #E0502,ERRBR ; SET THE ERROR NUMBER IN ERROR TABLE.
```
5295 025540 012701 007241 MOV @EM0502,RI ;SELECT ERROR MESSAGE.
5296 026544 104460 ERROR ;REPORT ERROR.
5297
5298 026546 012700 000340 TRAP C*ERROR
5299 026552 104441 601: SETPRI @PRI07 ;DISABLE ALL INTERRUPTS.
5300 026554 005067 153526 MOV @PRI07,RO ;INDICATE THAT WE COMPLETED THE TEST.
5300 026560 104401 TRAP C*SPrI
5300 026560 104401 ENDTST
5300 026560 104401
.SBTIL HARDWARE TEST RDAA

;*** ***********************************************************************
;* CSR RX DATA AVAILABLE BIT TEST
;* ^ THIS TEST VERIFIES THAT THE DUT CSR RX DATA AVAILABLE BIT IS SET BY THE
;* ^ INCLUSION OF THE SELFTEST CODES IN THE DUT FIFO AND THAT THE BIT CLEARS
;* ^ AFTER THE FIFO HAS BEEN EMPTYED.

;***********************************************************************

BGNST

SETPRI @PRIO5 ;ALLOW LTC INTERRUPTS.

MOV @PRIO5,RO
TRAP C1SPRI

TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.

MOV @TNUM,TNUM ;SET UP THE TEST NUMBER.

MOV @1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.

MOV @01,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.

MOV @PM0604,Ermmsg ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.

MOV @PM0202,ERRBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.

; ; SET THE DUT CSR MASTER RESET (MR) BIT, PERFORM THE SKIP SELFTEST SEQUENCE,
; ; AND WAIT UP TO 5 SECONDS FOR THE MR BIT TO CLEAR.

MOV @5000,R1 ;TIME OUT VALUE IS 5.0 SECONDS.

MOV @BIT05,R2 ;WAITING FOR MASTER RESET BIT.

CLR R3 ;WAITING FOR BIT TO CLEAR.

MOV CSR,R4 ;BIT IS IN THE DUT'S CSR.

SET THE DUT MASTER RESET BIT.

JSR PC,SKPST ;SKIP THE SELFTEST.

JSR PC,MSLGET ;WAIT FOR DUT_CSR MR BIT TO CLEAR.

BCC 41 ;GO REPORT ERROR IF MR DID NOT CLEAR.

; ; CHECK THAT THE RX DATA AVAILABLE BIT IS SET.

BIT @BIT7,R4 ;TEST THE DUT RX(DATA,AVAL BIT.

BEQ 61 ;GO REPORT ERROR IF BIT IS NOT SET.

; ; READ CHARACTERS FROM THE DUT RX FIFO AND WAIT FOR RX,DATA,AVAL TO GO CLEAR.

MOV @5000,R5 ;ALLOW READING 500 CHAR S BEFORE ERROR.

MOV R4,R3 ;CALCULATE THE RBUF ADDRESS.

MOV R5,R0 ;READ A CHARACTER FROM THE RX FIFO.

MOV @BIT17,R4 ;TEST THE DUT RX,DATA,AVAL BIT.

BEQ 60 ;EXIT TEST WITHOUT ERROR IF RX,DATA,AVAL CLR.

DEC R5 ;COUNT THE CHARACTER JUST READ.

BNE 21 ;LOOP IF NOT TOO MANY CHAR READ FROM FIFO.

BR 81 ;GO REPORT ERROR IF RX,DATA,AVAL WOULDN'T CLR.

; ; ERROR REPORTS:

; ; REPORT MR BIT WOULD NOT CLEAR AFTER A DUT RESET.

MOV @0601,ERRNR ;SET THE ERROR NUMBER IN ERROR TABLE.

MOV @EM0202,R1 ;SELECT ERROR MESSAGE.

ERROR ;REPORT ERROR.

TRAP C1ERROR
REPORT THAT RX.DAT.AVAIL BIT WAS NOT SET AFTER A RESET COMPLETION.
MOV #0602, ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
MOV #EM0602.R1 ;SELECT ERROR MESSAGE.
REPORT ERROR. >>>>> ERROR #0602 <<<<< TRAP C#ERROR

REPORT THAT RX.DAT.AVAIL BIT COULD NOT BE CLEARED BY PURGING FIFO.
MOV #0603, ERRNBR ;SET THE ERROR NUMBER IN ERROR TABLE.
MOV #EM0603.R1 ;SELECT ERROR MESSAGE.
REPORT ERROR. >>>>> ERROR #0603 <<<<< TRAP C#ERROR

SETPRI #PRI07 ;DISABLE ALL INTERRUPTS.
MOV #PRI07, R0
TRAP C#SPRI

CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
ENDTST

L10033: TRAP C#ETST
.SBITL HARDWARE TEST - RDVA -
***
* THIS TEST VERIFIES THAT THE OUT RBUF RX DATA VALID BIT IS SET BY THE
* INCORPORATION OF THE PERMANENT CODES IN THE OUT FIFO AND THAT THE BIT CLEAR
* AFTER THE FIFO HAS BEEN EMPTYED.
***

BGNRTST

SETPRI @PR105 ;ALLOW LTRC INTERRUPTS.

TNUM := TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV @TNUM,TSTNUM ;SET THE TEST NUMBER.
MOV @1,CTRLCF ;INDICATE THAT WE ARE WITHIN A TEST.
MOV @1,ERRTYPE ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV #EMO701,ERRORMG ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
MOV #ER0201,ERRORBLK ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.

SET THE DUT CSR MASTER_RESET (MR) BIT, PERFORM THE SKIP SELFTEST SEQUENCE,
AND WAIT UP TO 5 SECONDS FOR THE MR BIT TO CLEAR.

MOV #5000,R1 ;TIME-OUT VALUE IS 5.0 SECONDS.
MOV #BIT05,R2 ;WAITING FOR MASTER_RESET BIT.
CLR R3 ;WAITING BIT TO CLEAR.
MOV CSR4,R4 ;BIT IS IN THE DUT'S CSR.
MOV R2,(R4) ;SET THE DUT MASTER_RESET BIT.
JSR PC,SKIPST ;SKIP THE SELFTEST.
JSR PC,MSLGET ;WAIT FOR OUT_CSR_MR BIT TO CLEAR.
BCC 41 ;GO REPORT ERROR IF MR DID NOT CLEAR.

CHECK THAT THE RX DATA VALID BIT IS SET.

MOV (R4),RO ;INCREMENT POINTER TO PNT TO DUT RBUF REG.
TST (R4) ;TEST THE OUT RX.DAT.A VALID BIT.
BPL 61 ;GO REPORT ERROR IF BIT IS NOT SET.

READ CHARACTERS FROM THE DUT RX FIFO AND WAIT FOR RX.DATA.VALID TO GO CLEAR.

MOV #600,R5 ;ALLOW READING 600 CHARs BEFORE ERROR.
2$:

MOV (R4),RO ;READ A CHARACTER FROM THE RX FIFO.
BPL 60$ ;EXIT TEST WITHOUT ERROR IF BIT IS CLEAR.
DEC R5 ;COUNT THE CHARACTER JUST READ.
BNE 2$ ;LOOP IF NOT TOO MANY CHARs READ FROM FIFO.
BR 8$ ;GO REPORT ERROR IF RX.DATA.VALID WOULDN'T CLR.

ERROR REPORTS:

REPORT MR BIT WOULD NOT CLEAR AFTER A DUT RESET.

MOV #027122,ERRORBR ;SET THE ERROR NUMBER IN ERROR TABLE.
MOV #EMO701,ERRORMG ;SELECT ERROR MESSAGE.
ERROR ;REPORT ERROR.
EXIT TEST.

MACRO M1200 15 MAR 84 09:15 PAGE 106
; REPORT THAT RX.DAT.A.VALID BIT WAS NOT SET AFTER A RESET COMPLETION.
5425  MOV  #0702, ERRNBR
5426  MOV  #EM0702,R1
5427  ERROR
5428  ;REPORT ERROR. >>>>> ERROR #0702 <<<<<
5429  TRAP  C#ERROR
5430
5431  ;REPORT THAT RX.DAT.A.VALID BIT COULD NOT BE CLEARED BY PURGING FIFO.
5432  MOV  #0703, ERRNBR
5433  MOV  #EM0703,R1
5434  ERROR
5435  ;SELECT ERROR MESSAGE. >>>>> ERROR #0703 <<<<<
5436  TRAP  C#ERROR
5437  ;SETPRI #PRI07 ;DISABLE ALL INTERRUPTS.
5438  CLR  CTRLCF
5439  ;INDICATE THAT WE COMPLETED THE TEST.
5440  ENDTST
5441  L10034:
5442  TRAP  C#ETST
**SBTLIL HARDWARE TEST**

**-------------------------------------------------------------------**

// THIS TEST VERIFIES THAT THE OUT RBUF RX LINE NUMBER FIELD IS WORKING
// CORRECTLY BY UTILIZING THE SELFTEST CODES WHICH ARE PUT IN THE RX
// FIFO AFTER A BOARD RESET.

**-------------------------------------------------------------------**

-- **BDNSTST**

SEIPR  #PRI05  ;ALLOW LTC INTERUPTS.

**-------------------------------------------------------------------**

TO:;

TNUM  = TNUM  + 1  ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.

MOV  @TNUM,TSTNUM  ;SET UP THE TEST NUMBER.  (8)

MOV  #6-1,CTRLCF  ;INDICATE THAT WE ARE WITHIN A TEST.

MOV  #1,ERRTYP  ;SET ERROR TYPE AS FATAL IN ERROR TABLE.

MOV  #EMOBO1,ERMSG  ;SET ERROR MESSAGE ADDRESS IN ERROR TABLE.

**-------------------------------------------------------------------**

;SET THE OUT CSR MASTER RESET (MR) BIT, PERFORM THE SKIP SELFTEST SEQUENCE,
;AND WAIT UP TO 5 SECONDS FOR THE MR BIT TO CLEAR.

;MOV  #5000,.R1  ;TIME-OUT VALUE IS 5.0 SECONDS.

MOV  #BIT05,.R2  ;WAITING FOR MASTER RESET BIT.

CLR  R3  ;WAITING FOR BIT TO CLEAR.

MOV  CSRA,.R4  ;BIT IS IN THE OUT'S CSR.

MOV  R2,.R4  ;SET THE OUT MASTER RESET BIT.

JSR  PC,SKPSST  ;SKIP THE SELFTEST.

JSR  PC,MSGET  ;WAIT FOR OUT CSR MR BIT TO CLEAR.

BCC  4$  ;GO REPORT ERROR IF MR DID NOT CLEAR.

**-------------------------------------------------------------------**

;READ CHARACTERS FROM THE OUT RX FIFO AND VERIFY THAT THE LINE NUMBERS ARE
;CORRECT.

;EIGHT CHARACTERS ARE READ FROM THE FIFO.

2$:  MOV  (R4),.R0  ;READ CHARACTER FROM THE OUT RX FIFO.

**-------------------------------------------------------------------**

;ERROR REPORTS:

;REPORT MR BIT WOULD NOT CLEAR AFTER A RESET.

MOV  #0081,.ERRNBR  ;SET THE ERROR NUMBER IN ERROR TABLE.

MOV  #ERROR03,.ERRBLK  ;SET ERROR ROUTINE ADDRESS IN ERROR TABLE.

MOV  #EMOBO2,.R1  ;SELECT ERROR MESSAGE.

ERROR  TRAP  CERROR
5493 027354 000411
5494 601: EXIT THE TEST.
5495 ;REPORT THAT RX LINE NUMBER FIELD IS WRONG FOR SELFTEST CODE.
5496 027356 012767 001442 154404
5497 027364 012767 016136 154402
5498 027372 012701 010656
5499 027376 ERROR
5500 104460
5501 027400 601: SETPRI #PRI07; DISABLE ALL INTERRUPTS.
5502 012700 000340
5503 027406 104441
5504 CLR CRCLCF
5505 152674
5506 ENDSTST
5507 L10035: TRAP C1ETST
**SBTL - HARDWARE TEST**

**BMPCHK**

---

**5505**

**5506** **BMP CHECK TEST**

**5507**

**5508** **THIS TEST IS USED TO VERIFY THAT THE OUT DOES NOT IMMEDIATELY FAIL**

**5509** **THE ON-BOARD BACKGROUND-MONITOR PROGRAM, AND HENCE INVALIDATE**

**5510** **SUCCEEDING TESTS.**

**5511** **THIS TEST LOOKS FOR BMP CODES IN THE FIFO FOR A SET PERIOD IMMEDIATELY**

**5512** **AFTER THE SELF TEST IS SKIPPED.**

**5513** **ANY BMP CODES THAT ARE FOUND ARE SAVED ON THE QUEUE AND ARE ALSO**

**5514** **REPORTED IN THIS TEST.**

**5515** **BGNIST**

**5516**

**5517** 027414

**5518** 027414 01?700 000240

**5519** 027420 104441

**5520** 000017

**5521** 027422 017267 000017 152674

**5522** 027430 017267 177777 152650

**5523** 027436 017267 000001 154322

**5524** 027444 017267 001605 154316

**5525** 027452 017267 010726 154312

**5526**

**5527**

**5528**

**5529** 027460 012701 005670

**5530** 027464 012702 000040

**5531** 027470 005003

**5532** 027472 016704 152544

**5533** 027476 004767 175462

**5534** 027502 103027

**5535**

**5536**

**5537**

**5538** 027504 010214

**5539** 027506 004767 172632

**5540**

**5541**

**5542**

**5543**

**5544** 027512 012704 000764

**5545** 027516 004767 170402

**5546** 027522 004767 171014

**5547** 027526 103015

**5548**

**5549**

**5550**

**5551**

**5552**

**5553**

**5554**

**5555**

**5556**

**5557**

**5558**

---

**ALC2**

**T9::**

**SETPRI @PRIO5**

**ALLOW LTC INTERRUPTS.**

**MOV @PRIO5,RC**

**TRAP C0SPR**

**TNUM = TNUM + 1**

**INCREMENT THE ASSEMBLY TIME TEST COUNTER.**

**MOV @TNUM,TSTNUM**

**SET UP THE TEST NUMBER.**

**(9)**

**MOV @1.CTRLF**

**INDICATE THAT WE ARE WITHIN A TEST.**

**MOV @01.ERR#F**

**SET ERROR TYPE AS TERMINAL IN ERROR TABLE.**

**MOV @0901.ERRNUM**

**SET THE ERROR NUMBER.**

**MOV @EM0901.ERRMSG**

**SET THE ERROR MESSAGE.**

**WAIT UP TO 3 SECONDS FOR THE OUT MASTER RESET BIT TO CLEAR.**

**IF TIME OUT OCCURS, THEN EXIT THIS TEST.**

**MOV @0000.R1**

**TIME-OUT VALUE IS 3.0 SECONDS.**

**MOV @0105,R2**

**WAITING FOR MASTER RESET BIT.**

**CLR R3**

**WAITING FOR BIT TO CLEAR.**

**MOV CSRA,R4**

**BIT IS IN THE OUT'S CSR.**

**JSR PC,MSLGET**

**WAIT FOR OUT CSR MR BIT TO CLEAR.**

**BCC 501**

**ABORT THE TEST IF MR DID NOT CLEAR.**

**REPEAT THE MASTER TEST.**

**MOV R2,(R4)**

**SET THE OUT MASTER RESET BIT.**

**JSR PC,SKPST**

**WRITE THE SKIP SELF TEST CODES TO THE OUT.**

**WAIT FOR MASTER RESET TO CLEAR. DELAY FOR 500 MILLISECONDS BEFORE PURGING THE FIFO.**

**MOV @500,R4**

**TIME-OUT VALUE IS 500 MILLISECONDS.**

**JSR PC,DEL**

**WAIT FOR BMP TO BEGIN EXECUTION.**

**JSR PC,PURIF**

**PURGE THE FIFO, SAVING ANY BMP CODES.**

**BCC 501**

**ABORT THE TEST IF THE FIFO DID NOT CLEAR.**

**REPORT THE ERROR IF ANY BMP CODES WERE FOUND.**

**MOV BMPGCP.R2**

**GET THE CONTENTS OF THE POINTER TO THE BMP Q.**

**MOV @BMPGCP,R3**

**GET THE START ADDRESS OF THE QUEUE.**

**CMP R2,R3**

**SEE IF THE POINTER HAS MOVED FROM THE BAST.**

**BEQ 601**

**EXIT NO CODC IN THE QUEUE.**

**REPEAT ERROR BMP CODE FOUND IN TEST NN. BMP CODE::NNNNNN**

---
MOV @EM0902,R1 ;PASS THE MESSAGE TO BE REPORTED.
ERRDF 0901,EM0901,ER9301 ; >>>>> ERROR #0901 <<<.

TRAP CIEROF
.WORD 901
.WORD EM0901
.WORD ER9301

BR 60:

MOV @#902,EMNRBR ;SET >>>>> ERROR #0902 <<<.

JSR PC,TSABAT ;REPORT NON TEST RELATED ERROR.

SETPRI #PRI07 ;DISABLE ALL INTERRUPTS.

MOV @PRI07,R0
TRAP CISPRI

CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.

ENDST

L10036:
TRAP CIETST
**HARDWARE TEST**

* SKIP SELF-TEST TEST
* THIS TEST VERIFIES THAT THE OUT SKIPS THE SELF TEST WITHIN THE
* TIME ALLOWED, AND THAT THE FIFO CONTAINS THE CORRECT CODES AFTER ITS
* COMPLETION.

```assembly
; SET PRI #PRI05 ; ALLOW LTC INTERRUPTS.
MOV #PRI05, R0
TRAP C$PRI

T0:;
INUM = INUM + 1 ; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV #INUM,TINUM
SET UP THE TEST NUMBER. (10)

MOV #0-1,CTRCLF ; INDICATE THAT WE ARE WITHIN A TEST.

MOV #1,ERRTYP ; SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV #M1001,ERRMSG ; SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
MOV #EROS03,ERABLK ; SET ERROR ROUTINE ADDRESS IN ERROR TABLE.

; WAIT UP TO 5 SECONDS FOR THE OUT MASTER RESET BIT TO CLEAR.
; IF TIME-OUT OCCURS, THEN EXIT THIS TEST.

MOV #500000,R1 ; TIME-OUT VALUE IS 5.0 SECONDS.
MOV #BIT05,R2 ; WAITING FOR MASTER RESET BIT.
CLR R3 ; WAITING FOR BIT TO CLEAR.

MOV #MSA,R4 ; BIT IS IN THE OUT'S CSA.
JSR PC,MSGET ; WAIT FOR OUT CSA MR BIT TO CLEAR.
BCC 50$ ; AVOID THE TEST IF MR DID NOT CLEAR.

; DETERMINE IF THE OUT 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 OUT MASTER RESET BIT.

JSR PC,SKPSTS ; WRITE THE SKIP SELFTEST CODES TO THE OUT.
JSR PC,MSGET ; WAIT FOR OUT CSA MR BIT TO CLEAR.
BCC 24$ ; GO REPORT ERR IF SKIPPING TEST TOOK TOO LONG.
CMP R1,#040.
BGT 41$ ; 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 OUT
; SUCCESSFULLY COMPLETED THE SELF-TEST.
; THIS SUBROUTINE REPORTS ERRORS WITH NUMBERS >>>>> 1003 THRU 1007 <<<<<.

MOV #1003,ERRNBR ; SET ERROR NUMBER TO 1003.
JSR PC,START ; CHECK SELF-TEST CODES IN THE FIFO.
BR 60$ ; EXIT TEST.

; ERROR REPORTS:
; REPORT SKIP SELF-TEST TOOK TOO LONG.
MOV #1001,ERRNBR ; SET THE ERROR NUMBER IN THE ERROR TABLE.
```
5624 027750 012701 011055 MOV @EM1002.R1 ;SELECT ERROR MESSAGE.
5625 027754 027754 ERROR ;REPORT ERROR.  >>>> ERROR 01001 <<<<
5626 027756 000414 TRAP CIERROR ;EXIT THE TEST.
5627
5628 ;REPORT SKIP SELF TEST COMPLETED TOO SOON.
5629 027760 012767 001752 154002 48. MDV @1002.ERRNBR ;SET THE ERROR NUMBER IN THE ERROR TABLE.
5630 027766 012701 011142 MDV @EM1003.R1 ;SELECT ERROR MESSAGE.
5631 027772 027772 ERROR ;REPORT ERROR.  >>>> ERROR 01002 <<<<
5632 027774 104460 TRAP CIERROR ;EXIT THE TEST.
5633
5634 027776 012767 001753 153764 50: MDV @1003.ERRNBR ;SET ERROR NUMBER.
5635 030004 004767 172462 JSR PC,TSABRT ;REPORT NON TEST RELATED ERROR.
5636
5637 030010 012700 000340 60: SETPRI #PRI07 ;DISABLE ALL INTERRUPTS.
5638 030010 012700 000340 MOV @PRI07,RO
5639 030014 104441 TRAP CI$PRI
5638 030016 005067 152264 CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5639 050022 ENDTST
5639 050022 L10037:
5639 050022 TRAP CI$EST
**SBITL HARDWARE TEST**

- **DFSKST**

```assembly
1: Diagnostic fail bit, skip self test test
**This test verifies that the diagnostic fail bit of the out, correctly
changes state as the on-board self test is skipped.
**

- **BQNTST**

```assemble
MOV @01,R1  ; SET TIME OUT OF 1 MILLI-SECOND.
BIS @BIT15,R3  ; WAIT FOR BIT TO SET
JSR PC,MSLGET  ; DOUBLE CHECK TO ELIMINATE NOISE PROBLEMS.
BCC 601  ; EXIT IF DIAG.FAIL BIT STILL CLEAR.
MOV R5,R1  ; PASS THE REMAINING TIME-OUT VALUE.
BR 21  ; LOOP TO CHECK AGAIN.

; ERROR REPORTS:

; REPORT DIAGNOSTIC FAIL BIT BAD.
MOV @1101,ERRNBR  ; SET THE ERROR NUMBER IN THE ERROR TABLE.
MOV @EM1205,R1  ; SELECT ERROR MESSAGE.
ERROR  ; REPORT ERROR.
          ; ERROR @1101
          ; TRAP C1ERROR
BR 601  ; EXIT THE TEST.

MOV @1102,ERRNBR  ; SET THE ERROR NUMBER FOR TSABRT RTN.
JSR PC,TSABRT  ; REPORT NON TEST RELATED ERROR.

SETPRI @PRIO7  ; DISABLE ALL INTERRUPTS.
MOV @PRIO7,RO  ; TRAP C*SPRI

CLR CTRLCF  ; INDICATE THAT WE COMPLETED A TEST.
ENDTST

L10040:  ; TRAP C*IETST
**SBITL HARDWARE TEST - SELFTS**

- **SELF-TEST TEST**
- THIS TEST VERIFIES THAT THE DUT'S SELF-TEST EXECUTES WITHIN THE TIME ALLOWED, AND THAT THE FIFO CONTAINS THE CORRECT CODES AFTER ITS COMPLETION.

- **GLOBAL**

- **BGTST**

```
SETPRI #PRI05 : ALLOW LTC INTERRUPTS.
MOV #PRI05,R0
TRAP C1SPRI

INUM == INUM + 1 : INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV @INUM,T,INUM : SET UP THE TEST NUMBER.

MOV 0,CTRLOM : INDICATE THAT WE ARE WITHIN A TEST.
MOV @1,ERRTYP : SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV @EM1201,ERRMSG : SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
MOV @ER0503,ERRBLK : SET ERROR ROUTINE ADDRESS IN ERROR TABLE.

; WAIT UP TO 5 SECONDS FOR THE DUT MASTER RESET BIT TO CLEAR.
; IF TIME-OUT OCCURS, THEN EXIT THIS TEST.

; DETERMINE IF THE SELF-TEST TAKES TOO SHORT OR TOO LONG A TIME TO COMPLETE.
; IF MR IS CLEAR IN LESS THAN 1/2 SECOND, OR GREATER THAN 5 SECONDS, REPORT THE ERROR.

; SELF-TEST COMPLETED WITHIN 1SEC TO 5 SECONDS.
; CHECK THE STATE OF THE DIAGNOSTIC FAIL BIT, REPORT ERROR IF IT IS SET.

; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED
```
BIT #8106, OPTION
EXIT WITH TEST FAILURE MESSAGE IF
NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
DURING THE SOFTWARE QUESTIONS.

; VERIFY THAT THE SELF TEST CODES IN THE FIFO ARE "GOOD" CODES, IE THE DUT
; SUCCESSFULLY COMPLETED THE SELF TEST.
; THIS SUBROUTINE REPORTS ERRORS WITH NUMBERS 1205 THRU 1209.

; REPORT SELF TEST TOOK TOO LONG TO COMPLETE.
MOV #1205, ERRNR
; SET ERROR NUMBER TO 1205.
JSR PC, RSTAP
; CHECK SELF-TEST CODES IN THE FIFO.
BR 601
; EXIT TEST.

; ERROR REPORTS:

; REPORT SELF-TEST DID NOT EXECUTE AFTER DUT RESET.
MOV #1201, ERRNR
; SET THE ERROR NUMBER IN THE ERROR TABLE.
MOV #EM1202, R1
; SELECT ERROR MESSAGE.
ERROR
; REPORT ERROR. ERROR NUMBER 1201.
TRAP C*ERROR

; REPORT SELF-TEST COMPLETE TOO SOON.
MOV #1203, ERRNR
; SET THE ERROR NUMBER IN THE ERROR TABLE.
MOV #EM1203, R1
; SELECT ERROR MESSAGE.
ERROR
; REPORT ERROR. ERROR NUMBER 1203.
TRAP C*ERROR

; DISABLE ALL INTERRUPTS.
MOV #PRI07, RO
TRAP C*SPRI

; INDICATE THAT WE COMPLETED THE TEST.
L10041:
ENDST
CTRLCF
5810 .SBTLI HARDWARE TEST .STFAIL .
5812
5813 SELF TEST FAIL TEST
5814
5815 THIS TEST VERIFIES THAT THE DUT WILL REPORT SELFTEST ERRORS VIA THE
5816 FIFO. AND THAT THE DIAGNOSTIC FAIL BIT WILL INDICATE THE ERROR.
5817 THIS IS ACCOMPLISHED VIA A SOFTWARE "HOOK" IN THE SELF-TEST, WHICH
5818 FORCES A "PROG TO RAM ERROR" TO BE PLACED IN THE FIFO.
5819
5820 030546
5821 030546
5822 030546 012700 000240
5823 030552 104441
5824 030554 000015
5825 030554 012767 000015 151542
5826 030562 012767 177777 151516
5827 030570 012767 000001 151370
5828 030576 012767 011543 153166
5829 030604 012767 016136 153162
5830 030612 012767 002425 153150
5831
5832 T13:
5833 030620 012701 011610
5834 030624 012702 000040
5835 030630 00003
5836 030632 016704 151404
5837 030636 004767 167322
5838 030642 103064
5839
5840
5841 030644 012777 000040 151370
5842 030652 012704 000031
5843 030656 004767 167242
5844 030662 012777 146314 151370
5845
5846 030670 005267 153074
5847 030674 012701 011610
5848 030700 012702 000040
5849 030704 005003
5850 030706 016704 151350
5851 030712 004767 167246
5852 030716 103036
5853
5854 030720 005267 153044
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5864 030724 032714 020000 BIT BIT13, (R4) ;CHECK THE STATE OF THE DIAG_FAIL BIT.
5865 030730 001425 BEQ 10$ ;GO REPORT ERROR IF DIAG_FAIL BIT CLEAR.
5866
5867
5868
5869
5870 030732 005267 153032 ;* REMOVE THE 8 SELF TEST CODES FROM THE FIFO, AND VERIFY THAT AT LEAST
5871 030736 012700 000010 ;ONE IS A PROCI TO RAM ERROR CODE (231).
5872 030742 005001
5873 030744 016704 151274
5874 030750 011402 6$:
5875 030752 100020 MOV (R4), R2 ;READ A CODE FROM THE FIFO.
5876 030754 042702 007400 BPL 50$ ;GO REPORT ERROR IF THE FIFO IS EMPTY.
5877 030760 120227 170231 BIC 07400, R2 ;REMOVE THE LINE NUMBER FROM THE CODE.
5878 030764 001001 CMPB R2, 0170231 ;IS IT THE CORRECT ERROR CODE?.
5879 030766 005201 INC R1 ;SET ERROR NUMBER TO 1304.
5880 030770 005300 MOV 68, R0 ;SET MAXIMUM READ COUNT.
5881 030772 001366 CLR R1 ;CLEAR THE CORRECT CODE COUNTER.
5882 030774 005701 MOV RBUFA, R4 ;GET ADDRESS OF THE RECEIVER BUFFER REGISTER.
5883 030776 001010 BNE 8$ ;SKIP NEXT INSTRUCTION, IF NOT A 231 CODE.
5884 031000 005267 152764 8$:
5885 031004 012701 011576 INC ERRNR ;INCREMENT COUNTER.
5886 031010 010460 MOV #EM1302.R1 ;SELECT ERROR MESSAGE.
5887 031012 000402 ERR 10$ ;REPORT ERROR.
5888 031014 004767 171452 TRAP C$ERROR ;EXIT THE TEST.
5889 031018 104460 BR 60$ ;EXIT THE TEST.
5890 031020 000340 JSR PC, TSABR ;REPORT NON-RELATED TEST ERROR.
5891 031024 104441 60$:
5892 031026 005067 151254 SETPRI #P107 ;DISABLE ALL INTERRUPTS.
5893 031032 005067 151254 CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
5894 031032 104401 ENDIST 10042:
5895
5896
**SBITIL HARDWARE TEST**

* **ROM VERSION TEST** -

This test verifies that the DUT's self-test places valid ROM version numbers in the FIFO after it has been skipped. The ROM version numbers will be reported (on the first pass only), if an affirmative answer was given to the software P table question.

```
TRAP C1SPRI
TNUM = TNUM + 1
INCRUMENT THE ASSEMBLY TIME TEST COUNTER.
MOV @TNUM,TSTNUM
SET UP THE TEST NUMBER. (14)
MOV @T1,CTRLFL
INDICATE THAT WE ARE WITHIN A TEST.
MOV @E11,E1RTYP
SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV @E14,E14ERRMSG
SET ERROR MESSAGE ADDRESS IN ERROR TABLE.
MOV @ER0503,ERRBLK
SET ERROR ROUTINE ADDRESS IN ERROR TABLE.
```

- **ACCESS FIFO**

```
IF TIME-OUT OCCURS, THEN EXIT THIS TEST.
```

```
MOV @3000,R1
TIME-OUT VALUE IS 3.0 SECONDS.
MOV @B1062,R2
WAITING FOR MASTER RESET BIT.
```

```
CLS PC,R1
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 501
ABORT THE TEST IF MR DID NOT CLEAR.
```

- **SET THE MASTER RESET BIT, AND SKIP THE SELF TEST.**

```
JSR PC,MSLGET
```

```
BCC 501
ABORT THE TEST IF MR DID NOT CLEAR.
```

```
```

- **REMOVE CHARACTERS FROM THE FIFO UNTIL EITHER:**
  - (A) THE FIFO IS PURGED, GO REPORT THE ERROR.
  - (B) THE MAXIMUM TRY COUNTER IS ZERO, GO REPORT THE ERROR.
  - (C) PROC_1'S ROM VERSION NUMBER WAS FOUND BEFORE PROC_2'S, GO REPORT ERROR.
  - (D) BOTH ROM VERSION NUMBERS HAVE BEEN FOUND.

```
MOV @E1401,ERRBR
SELECT MESSAGE TO BE REPORTED IF FIFO EMPTY.
```

```
MOV @R8BUF,R2
READ THE NEXT CHAR FROM THE FIFO.
BPL 12
GO REPORT ERROR IF FIFO EMPTY.
```

```
```

- **CHECK IF THE READ DATA IS A BMP CODE.**
5951 031176 012700 000501
5952 031202 040200
5953 031204 001003
5954 031206 004767 171064
5955 031212 000435
5956
5957
5958
5959 031214 012700 000201
5960 031220 040200
5961 031222 001431
5962
5965
5966 031224 032702 000002
5967 031230 001407
5968
5969 031232 010204
5970 031234 042704 177603
5971 031240 000241
5972 031242 006004
5973 031244 000417
5974 031250 010203
5975 031252 042701 177603
5976 031256 000241
5977 031260 006003
5978 031262 006003
5979 031264 020427 000143
5980 031270 001016
5981
5982
5983
5984
5985 031272 012701 011762
5986 031276 012767 002572 152464
5987 031304 000435
5988
5989 031306 005305
5990 031310 001327
5991 031312 012701 012035
5992 031316 012767 002573 152444
5993 031324 000423
5994
5995
5996
5997
5998 031326 032767 000001 150670
6000 031336 026727 150750 000001
6001 031344 030525
6002 031346
6003 031346 010446
6004 031350 010346
6005 031352 012746 004154
6006 031356 012746 000003
031362 010600
031364 104414
031366 062706 000010
6003 031372 000412
       BR  60$  ;EXIT THIS TEST.
6004
6005
6006
6007 031374 012767 016254 152572 12$: MOV @ER1401,ERRBLK ;SELECT THE ERROR REPORTING ROUTINE.
6008 031402 ERROR
6009 031402 ERROR
6010 031404 000405
60 0
60.1 031406 012767 002575 152554 50$: MOV @1405..ERRNBR ;SET UP ERROR NUMBER FOR TSABRT RTN.
60.2 031414 004767 171052 JSR PC,TSABRT ;REPORT NON-TEST RELATED ERROR.
60.13
60.14 031420 012700 000340 60$: SETPRI @PRI07 ;DISABLE ALL INTERRUPTS.
60.15 031424 104411 TRAP @PRI07,RO
60.16 031426 005067 150654 CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
60.17 031432 ENDTST
60.18 031432
60.19 031432 104401

MOV SP,RO
TRAP CISPNTE
ADD #010.5P
0618  6018  .SBTTL HARDWARE TEST
0619  CSAB4
0620  .********************************************************************************
0621  |
0622  CSR BIT A TEST
0623  |
0624  THIS TEST VERIFIES THAT WHEN THIS BIT IS SET (AT THE SAME TIME
0625  AS MASTER RESET) THE OUT REMAINS INACTIVE WITH THE MASTER RESET
0626  BIT SET; AND WHEN CSR BIT 4 IS SUBSEQUENTLY CLEARED, THE BOARD
0627  BECOMES ACTIVE AND REPORTS SIX SKIP SELFTEST CODES IN THE RXIFO.
0628  ANY BMP CODES FOUND IN THE FIFO ARE SAVED TO BE REPORTED LATER.
0629  |
0630  BGNTST
0631  SETPRI #PRI05
0632  ;ALLOW LTC INTERRUPTS.
0633  |
0634  T15:;
0635  MOV #PRI05,RO
0636  TRAP CSPI
0637  |
0638  TNUM = TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
0639  MOV #TNUM,TSTNUM
0640  ;SET UP THE TEST NUMBER.
0641  MOV #1,CTRCLF
0642  ;INDICATE THAT WE ARE WITHIN A TEST.
0643  MOV #1,ERRTYP
0644  ;SET ERROR TYPE AS FAULT IN ERROR TABLE.
0645  MOV #1501,ERRNR
0646  ;SET THE ERROR NUMBER TO 1501.
0647  MOV #EM1501,ERRMSG
0648  ;SET THE ERROR MESSAGE ADUR.
0649  MOV #ERCS03,ERRBLK
0650  ;SET THE ERROR REPORTING ROUTINE.
0651  |
0652  ;WAIT FOR THE MASTER RESET BIT TO CLEAR. REPORT THE ERROR IF IT FAILS
0653  ;TO CLEAR.
0654  |
0655  MOV #5000,R1
0656  ;SET THE TIME-OUT VALUE OF 5 SEC.
0657  MOV #BTO5,R2
0658  ;INDICATE TO TEST BIT 5.
0659  CLR R3
0660  ;INDICATE TO TEST BIT CLEAR.
0661  MOV CSRA.R4
0662  ;INDICATE TO TEST THE CSR REG.
0663  JSR PC,MMLSRO
0664  ;WAIT FOR THE BIT TO CLEAR
0665  BCC 504
0666  ;JUMP TO REPORT ERROR IF BIT FAILED TO CLEAR.
0667  |
0668  ;SET THE MASTER RESET AND CSR BIT 4 BIT, AND THEN WAIT 5.5 SEC TO ENSURE
0669  ;THAT THE MR BIT DOESN'T CLEAR.
0670  |
0671  INC ERRNR
0672  ;SET THE ERROR NUMBER TO 1502.
0673  MOV #60,BCSRA
0674  ;RESET THE BOARD WITH BIT 4 SET.
0675  |
0676  ;VERIFY THAT CSR BIT 4 IS SET.
0677  |
0678  MOV BCSRA.R0
0679  ;READ THE CSR.
0680  BIT #BIT04.R0
0681  ;TEST BIT 4.
0682  BEQ 401
0683  ;EXIT WITH ERROR IF THE BIT IS CLEAR.
0684  |
0685  ;WAIT 5 SECONDS FOR THE MR BIT TO CLEAR.
0686  |
0687  INC ERRNR
0688  ;SET THE ERROR NUMBER TO 1503.
0689  MOV #5500,R1
0690  ;SET THE TIME-OUT VALUE OF 5.5 SEC.
0691  JSR PC,MMLSRO
0692  ;WAIT FOR THE MASTER RESET BIT TO CLEAR.
0693  BCS 404
0694  ;REPORT THE ERROR IF THE MR BIT CLEARED.
; CLEAR CSR BIT 4 AND VERIFY THAT THE MASTFR RESET BIT ALSO CLEARS.
Gov BCSRA,R5  ; READ THE CSR.
BIC @BITO4,R5  ; CLEAR BIT 4.
MOV R5,9CSCRA  ; RESTORE THE CONTENTS OF THE CSR.

; VERIFY THAT CSR BIT 4 CLEARED.

; WAIT FOR THE MR BIT TO CLEAR.

; READ SIX CHARACTERS FROM THE RXFIFO AND VERIFY THEY ARE SKIP SELFTEST CODES. SAVE ANY BMP CODES FOUND TO BE REPORTED LATER.

; READ SIX CHARACTERS FROM THE RXFIFO AND VERIFY THEY ARE SKIP SELFTEST CODES. SAVE ANY BMP CODES FOUND TO BE REPORTED LATER.

MOV @ER9008,ERBLK  ; SET UP THE ERROR ROUTINE.
MOV @ERM024,R1  ; SET THE ERROR MESSAGE.
MOV @R6,R4  ; "IMPROPER SELFTEST CODE FOUND".

; NUMBER OF CHAR'S TO READ.

; READ A CODE FROM THE RXFIFO.

BPL 501  ; EXIT WITH ERROR IF THE FIFO IS EMPTY.
MOV R2,R3  ; COPY THE CODE.

; CLEAR THE LINE NUMBER AND ERROR FLAGS.

; CHECK IF THE CODE IS A BMP CODE.

; AVOID SAVING THE BMP CODE IF IT ISN'T.

; SAVE THE BMP CODE.

; READ MORE DATA FROM THE RXFIFO.

; IS THE CODE A SKIP SELFTEST?

; BRANCH TO AVOID THE ERROR IF IT IS.

; REPORT UNEXPECTED SELFTEST CODE FOUND.

; IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED THEN EXIT THE TEST WITH THE TEST FAILURE MESSAGE.

; HAS EXTENDED ERROR REPORTING BEEN REQUESTED?

; EXIT THE TEST IF IT HAVEN'T.

; DECREMENT THE CODE COUNT.

; BRANCH AND READ ANOTHER CODE IF NOT ALL
BR 601

; HAVE BEEN READ.
; OTHERWISE, EXIT THE TEST, THE DUT HAS PASSED
; THIS TEST.

; REPORT THE ERROR "CSR BIT 4 BAD" AND EXIT THIS TEST.

401: MOV #0EM1502,R1 ; SET UP THE EXTENDED ERROR MESSAGE AS,
    ERROR ; "CSR BIT 4 BAD".
    ; REPORT THE ERROR.
    TRAP C#ERROR
    BR 601 ; EXIT THE TEST.

; REPORT A NON-RELATED TEST ERROR.

501: JSR PC.TSABRT ; REPORT THE ERROR.

601: CLR CTRLCF ; INDICATE THAT WE ARE NOT WITHIN A TEST.

ENDTST

L10044: TRAP C#ETST
**SECTION HARDWARE TEST**

**DEVICE REGISTERS WORD ACCESS READ AND WRITE TEST**

- **THIS TEST VERIFIES THAT THE DEVICE REGISTERS CAN BE READ AND WRITTEN CORRECTLY USING WORD ACCESSES.**

**BGNST**

```assembly
SETPRI @PRI05 ; ALLOW THE LTC TO INTERRUPT.

TRAP CSPRI

INUM = INUM + 1 ; INCREMENT THE ASSEMBLY TIME TEST COUNTER.

MOV @INUM,TSTNUM ; SET UP THE TEST NUMBER. (16)

MOV @0-1,CTRLCF ; INDICATE THAT WE ARE WITHIN A TEST.

MOV @0-1,ERRORTY ; SET UP DEVICE FATAL INDICATOR IN ERROR TYPE.

MOV @0-1,ERRORNZ ; SET UP ERROR NUMBER IN THE ERROR TABLE.

CLR ERMIF ; CLEAR THE ERROR SUMMARY FLAGS.

MOV @0-1,ERRORC ; CLEAR THE ERROR COUNTER TABLE.

CLR EXGERR ; CLEAR THE "EXIT ON ERROR" FLAG

JR SR,PC,CLRCLEW

JSR PC,CLRCLEW

BCS .6 ; FATAL ERROR? NO, CONTINUE WITH TEST.

JMP 601 ; YES, EXIT THE TEST.

; VERIFY READ/WRITE CAPABILITY TO INOREC ADDRESS FIELD OF CSR

INC ERRNBR ; SET THE ERROR REPORT NUMBER TO 1602.

MOV @7,R2 ; SET LOOP COUNTER.

MOV @0-1,CSR A ; GET CSR ADDRESS.

MOV @0-1,CSR R4 ; WRITE CONTENTS OF THE CSR.

MOV @0-1,CSR R4 ; MASK OUT ALL BUT THE INO_ADDR.REG FIELD.

CMP @0-1,CSR R4 ; CHECK FOR CORRECT DATA WRITTEN/READ.

BEQ 4 ; IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.

REPORT "BAD BIT(S) IN DEVICE CSR REGISTER FOR LINE 0 (0)".

CLR R3 ; SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.

CLR R5 ; CAUSE REPORT OF LINE 0.

ERROR ; ------ ERROR # 1602 <<<< TRAP CERROR

; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED.

BIT #BIT06,OPTION ; HAS EXTENDED ERROR BEEN REQUESTED?

BEQ 601 ; EXIT THE TEST IF IT HASN'T.

DEC R2 ; DECREMENT LOOP COUNT/IND.ADD.REG ADDRESS.

BGE 2 ; LOOP BACK TO TEST NEXT ADDRESS IF NOT DONE.
```
WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL REGISTERS ON ALL ACTIVE LINES. BEFORE WRITING EACH PATTERN, CLEAR ALL THE BITS.

REGIST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>>> ERROR 1603 1605 <<<...

INC ERRNBR ;SET THE ERROR NUMBER TO 1603.
CLR R5 ;INDICATE THAT WORD AIDS ARE TO BE USED.
MOV 02, R4 ;INDICATE R/W ACCESS, CLEAR FIRST.
JSR PC, REGIST ;WRITE AND VERIFY DATA PATTERNS.

EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

TST EXOERR ;IS THE "EXIT ON ERROR" FLAG SET?
BNE 601 ;EXIT IF IT IS.

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.

REGIST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>>> ERROR 1606 - 1608 <<<...

MOV #1606, ERRNBR ;SET UP ERROR NUMBER FOR REGIST ROUTINE.
CLR R3 ;INDICATE THAT WORD AIDS ARE TO BE USED.
NEG R4 ;INDICATE R/W ACCESS, SET FIRST.
JSR PC, REGIST ;WRITE AND VERIFY DATA PATTERNS.

EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

TST EXOERR ;IS THE "EXIT ON ERROR" FLAG SET?
BNE 601 ;EXIT IF IT IS.

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.
CLR R5 ;INDICATE THAT WE COMPLETED THE TEST.
ENDTST

L10045: TRAP CIETST
** **

** DEVICE REGISTER WORD ACCESS READ/MODIFY/WRITE TEST **

** THIS TEST VERIFIES THAT THE DEVICE REGISTERS CAN BE WRITTEN CORRECTLY **

** USING WORD READ/MODIFY/WRITE ACCESSES. **

---

```assembly
BGNST
SETPRI @PRI05
; ALLOW THE LTC TO INTERRUPT.

TNUM ++ TNUM + 1
; INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV @TNUM,TSTNUM
; SET UP THE TEST NUMBER.

MOV @-1,CTRCLF
; INDICATE THAT WE ARE WITHIN A TEST.

MOV @1,ERRTP
; SET UP DEVICE ERROR INDICATOR.

MOV @1701,ERRNBR
; SET UP ERROR NUMBER IN THE ERROR TABLE.

MOV @1701,ERRMSG
; SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.

CLR ERMRF
; CLEAR THE ERROR SUMMARY FLAGS.

MOV @ERCB9.RO
; CLEAR THE ERROR COUNTER TABLE.

CLR EXOERR
; CLEAR THE "EXIT ON ERROR" FLAG.

; RESET THE DUT TO A KNOWN STATE. DO NOT REMOVE THE STATUS CODES FROM THE FIFO.

; CLEAR TX AND RX INTERRUPT ENABLE BIT IN THE CSR.

; THIS SUBROUTINE REPORTS ERRORS >>>> 1701 <<<<.

; JSR PC,RESET
; SET THE DMU-11, REPORT ANY ERRORS FOUND.

; BCS .6
; FATAL. RESET ERROR? NO, CONTINUE WITH TEST.

; JMP 601
; YES, EXIT THE TEST.

; THE READ/MODIFY/WRITE CAPABILITY TO INDIRECT ADDRESS FIELD OF CSR IS
; NOT TESTED THIS THIS FORM OF ACCESS IS ILLEGAL.

; WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL REGISTERS ON ALL
; ACTIVE LINES USING R/M/W. BEFORE WRITING EACH PATTERN, CLEAR ALL THE BITS.
; REGST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1703 - 1705 <<<<.

; MOV @1703,ERRNBR
; SET THE ERROR NUMBER TO 1703.

; CLR R3
; INDICATE THAT WORD ACCESSES ARE TO BE USED.

; MOV @1,R4
; INDICATE R/M/W ACCESS. CLEAR FIRST.

; JSR PC,REGST
; WRITE AND VERIFY DATA PATTERNS.

; EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
; NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

; 1ST EXOERR
; IS THE "EXIT ON ERROR" FLAG SET ?

; BNE 601
; EXIT IF IT IS.

; 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.
; REGST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1706 - 1708 <<<<.
```
6297 032370 012767 003252 151372 ; MOV #1706..,EARNBR ; SET UP ERROR NUMBER FOR REGST ROUTINE.
6298 032376 005003 ; CLR R3 ; INDICATE THAT WORD ACCESSES ARE TO BE USED.
6299 032400 005404 ; NEG R4 ; INDICATE R/M/W ACCESS, SET FIRST.
6300 012402 004767 166470 ; JSR PC,REGST ; WRITE AND VERIFY DATA PATTERNS.
6301 /* EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS */
6302 ; NOT BEEN REQUESTED, I.E. EXOERR IS NON ZERO.
6303 /* IS THE "EXIT ON ERROR" FLAG SET? */
6304 /* EXIT IF IT IS. */
6305 032406 005767 147672 ; TST EXOERR
6306 032412 001005 ; BNE 601
6307 /* PRINT ERROR SUMMARY REPORTS IF NECESSARY. */
6308 /* THE FOLLOWING ROUTINE REPORTS ERRORS WITH NUMBER >>>>> ERROR # 1709 <<< */
6309 /* */
6310 032414 012767 003255 151346 ; MOV #1709..,EARNBR ; SET UP ERROR NUMBER FOR NEXT RTN.
6311 032422 004767 166702 ; JSR PC,REPSMR ; REPORT ERROR SUMMARY IF NECESSARY.
6312 032426 005067 147654 ; CLR CTLCF ; INDICATE THAT WE COMPLETED THE TEST.
6313 032432
6314 032432
6315 032432
6316 032432 104401
6317 L10046: TRAP C*ETST
BGNST

SETPRI @PRI05

TRAP CISPRI

MOV @PRI05,RO

T18:;

INCNUM = INCNUM + 1

T19:;

MOV @NUM,TSTNUM

CLF SET THE TEST NUMBER.

MOV @1,CTRL,CF

INDICATE THAT WE ARE WITHIN A TEST.

MOV @1,ERATYP

SET UP DEVICE FATAL INDICATOR IN ERROR TABLE.

MOV @1801,ERRNBR

SET UP ERROR NUMBER IN THE ERROR TABLE.

MOV @ERR01,ERRMSG

SET UP ERROR MESSAGE FOR TEST IN ERROR TABLE.

CLR ERSMRF

CLEAR THE ERROR SUMMARY FLAGS.

MOV @ERRNCB,RO

CLR EXDERR

CLEAR THE ERRNTER COUNTER TABLE.

CLR EXDERR

CLEAR THE EXIT ON ERROR" Flag.

RESET THE DUT TO A KNOWN STATE; DO NOT REMOVE THE STATUS CODES FROM THE FIFO.

CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.

THIS SUBROUTINE REPORTS ERRORS 1801, 1802.

JSR PC,RESETT

FATAL ERROR? NO, CONTINUE WITH TEST.

JMP 601

YES, EXIT THE TEST.

MOV @1802..ERRNBR

SET THE ERROR REPORT NUMBER TO 1802.

VERIFY READ/WRITE CAPABILITY TO INDIRECT ADDRESS FIELD OF CSR.

USE BYTE ACCESS.

MOV @17,R2

SET LOOP COUNT.

MOV CSRA,R4

GET CSR ADDRESS.

2:

MOVB R2,(R4)

WRITE CONTENTS OF THE CSR.

MOV @11401,R1

READ BACK THE CONTENTS OF THE CSR.

BIC @177760,R1

MAsk OUT ALL BUT THE IND.ADR. REG FIELD.

CMP R1,R2

CHECK FOR CORRECT DATA WRITTEN/READ.

BEQ 41

IS EXPECTED DATA BAD? NO, SKIP ERROR REPORT.

MOV @ERR1601,ERRBLK

REPORT "BAD BIT(S) IN DEVICE" CSR REGISTER FOR LINE 0 (Q)."

MOV @ERR10001,ERRBLK

SELECT THE PROPER ERROR REPORT ROUTINE.

CLR R5

SET OFFSET TO 0 TO CAUSE REPORT OF CSR REG.

CLR R5

CAUSE REPORT OF LINE 0.

ERROR

EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED

EXIT WITH TEST FAILURE MESSAGE IF

NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED

DURING THE SOFTWARE QUESTIONS.
WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL LOWER BYTES OF ALL
REGISTERS ON ALL ACTIVE LINES. USE READ/WRITE ACCESSES. BEFORE WRITING
EACH PATTERN, CLEAR ALL THE USED BITS OF ALL ACTIVE REGISTERS.
REGST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1803 1805 <<<<
INC ERRNR ;SET THE ERROR NUMBER TO 1803.
MOV #1,R3 ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
JSR PC,REGST ;WRITE AND VERIFY DATA PATTERNS.

* ;EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
 NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

1ST EXOERR ;IS THE "EXIT ON ERROR" FLAG SET ?
BNE 604 ;EXIT IF IT IS.

WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL HIGH BYTES OF ALL
REGISTERS ON ALL ACTIVE LINES. USE READ/WRITE ACCESSES. BEFORE WRITING
EACH PATTERN, CLEAR ALL THE USED BITS OF ALL ACTIVE REGISTERS.
REGST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1806 - 1808 <<<<
MOV #1806.,ERRNR ;SET UP ERROR NUMBER FOR REGST ROUTINE.
NEG R3 ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
JSR PC,REGST ;WRITE AND VERIFY DATA PATTERNS.

* ;EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
 NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

TST EXOERR ;IS THE "EXIT ON ERROR" FLAG SET ?
BNE 604 ;EXIT IF IT IS.

WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL LOWER BYTES OF ALL
REGISTERS ON ALL ACTIVE LINES. USE READ/WRITE ACCESSES. BEFORE WRITING
EACH PATTERN, SET ALL THE USED BITS OF ALL ACTIVE REGISTERS.
REGST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1809 - 1811 <<<<
MOV #1809.,ERRNR ;SET UP ERROR NUMBER FOR REGST ROUTINE.
NEG R3 ;INDICATE THAT LO BYTE ACCESSES ARE TO BE USED.
NEG R4 ;INDICATE R/W ACCESS, SET FIRST.
JSR PC,REGST ;WRITE AND VERIFY DATA PATTERNS.

* ;EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
 NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

TST EXOERR ;IS THE "EXIT ON ERROR" FLAG SET ?
BNE 604 ;EXIT IF IT IS.

WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL HIGH BYTES OF ALL
REGISTERS ON ALL ACTIVE LINES. USE READ/WRITE ACCESSES. BEFORE WRITING
EACH PATTERN, SET ALL THE USED BITS OF ALL ACTIVE REGISTERS.
REGST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1812 1814 <<<<
MOV #1812.,ERRNR ;SET UP ERROR NUMBER FOR REGST ROUTINE.
6427 032716 005403  NEG R3 ;INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
6428 032720 004767 166152  JSR PC,REGTST ;WRITE AND VERIFY DATA PATTERNS.
6429
6430 16
6431 16
6432 16
6433 032724 005767 147354  TST EXOERR ;IS THE "EXIT ON ERROR" FLAG SET ?
6434 032730 001005  BNE 60$ ;EXIT IF IT IS.
6435
6436 16
6437 16
6438 032732 012767 003427 151030  MOV #1815,ERRORNBR ;SET ERROR NUMBER FOR NEXT RTN.
6439 032740 004767 166364  JSR PC,REPSMR ;REPORT ERROR SUMMARY IF NECESSARY.
6440 032744 005067 147336 60$: CLR CTRLCF ;INDICATE THAT WE COMPLETED THE TEST.
6441 032750 032750
6442 032750 104401  ENDTST

L10047: TRAP C*ETST
.SBTL Hardware Test

;** **************************************************************************
;* - Device register byte access read/modify/write test
;* This test verifies that the device registers can be read and written
;* correctly using byte accesses in read/modify/write mode.
;******************************************************************************

BGNST

SETPRI @PRI05 ; Allow the LTC to interrupt.

MOV @PRI05,RO

TRAP CI$PRI

032752 032752 012700 000240 032756 104441
000023

032760 01767 000023 147336
032766 012767 177777 147312
032774 012767 000001 150764
033002 012767 003555 150760
033010 012767 012517 150754
033016 005067 147440
033022 012700 002464
033026 004767 164772
033032 005067 147246

;* Reset the interrupt enable bit to a known state, do not remove the
;* status codes from the FIFO.

;* CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.

;* THIS SUBROUTINE REPORTS ERRORS 0001 <<<<.

;* JSR PC,RESETT ; RESET THE DMU-11, REPORT ANY ERRORS FOUND.

;* BCS +6 ; FATAL RESET ERROR? NO, CONTINUE WITH TEST.

;* JMP 60$ ; YES, EXIT THE TEST.

;* MOV @01903,ERRNBR ; SET THE ERROR REPORT NUMBER TO 1903.

;* THE READ/MODIFY/WRITE CAPABILITY TO INDIRECT ADDRESS FIELD OF CSR IS NOT
;* TESTED SINCE THIS IS AN ILLEGAL FORM OF ACCESS TO THIS REGISTER.

;*

;* WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL LOWER BYTES OF ALL

;* REGISTERS ON ALL ACTIVE LINES. USE READ/MODIFY/WRITE ACCESSES. BEFORE

;* WRITING EACH PATTERN, CLEAR ALL THE USED BITS OF ALL ACTIVE REGISTERS.

;* REGST ROUTINE REPORTS ERRORS WITH NUMBERS 0001 <<<< ERROR 1903 - 1905 <<<<.

;* INC ERRNBR ; SET THE ERROR NUMBER TO 1903.

;* MOV 0-1,R3 ; INDICATE THAT LO BYTE ACCESSSES ARE TO BE USED.

;* MOV 0-1,R4 ; INDICATE R/M/W ACCESS, CLEAR FIRST.

;* JSR PC,REGST ; WRITE AND VERIFY DATA PATTERNS.

;* EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
;* NOT BEEN REQUESTED, I.E. EXODIR IS NON-ZERO.

;* TST EX0ERR ; IS THE "EXIT ON ERROR" FLAG SET?

;* BNE 60$ ; EXIT IF IT IS.

;* WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL HIGH BYTES OF ALL
REGISTERS ON ALL ACTIVE LINES. USE READ/MODIFY/WRITE ACCESSES. BEFORE
WRITING EACH PATTERN, CLEAR ALL THE USED BITS OF ALL ACTIVE REGISTERS.

REGIST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1906 - 1908 <<<<<.

MOV @1906..,ERRNR : SET ERROR NUMBER FOR REGIST ROUTINE.
NEG R3 : INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
JSR PC,REGIST : WRITE AND VERIFY DATA PATTERNS.

EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

TST EXOERR : IS THE "EXIT ON ERROR" FLAG SET?
BNE 60$ : EXIT IF IT IS.

WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL LOWER BYTES OF ALL
REGISTERS ON ALL ACTIVE LINES. USE READ/MODIFY/WRITE ACCESSES. BEFORE
WRITING EACH PATTERN, SET ALL THE USED BITS OF ALL ACTIVE REGISTERS.

REGIST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1909 - 1911 <<<<<.

MOV @1909..,ERRNR : SET ERROR NUMBER FOR REGIST ROUTINE.
NEG R4 : INDICATE R/M/W ACCESS, SET FIRST.
JSR PC,REGIST : WRITE AND VERIFY DATA PATTERNS.

EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

TST EXOERR : IS THE "EXIT ON ERROR" FLAG SET?
BNE 60$ : EXIT IF IT IS.

WRITE AND VERIFY 16 DATA PATTERNS IN ALL USED BITS OF ALL HIGH BYTES OF ALL
REGISTERS ON ALL ACTIVE LINES. USE READ/MODIFY/WRITE ACCESSES. BEFORE
WRITING EACH PATTERN, SET ALL THE USED BITS OF ALL ACTIVE REGISTERS.

REGIST ROUTINE REPORTS ERRORS WITH NUMBERS >>>>> ERROR 1912 - 1914 <<<<<.

MOV @1912..,ERRNR : SET ERROR NUMBER FOR REGIST ROUTINE.
NEG R3 : INDICATE THAT HI BYTE ACCESSES ARE TO BE USED.
JSR PC,REGIST : WRITE AND VERIFY DATA PATTERNS.

EXIT THE TEST IF AN ERROR HAS BEEN FOUND AND EXTENDED ERROR REPORTING HAS
NOT BEEN REQUESTED, I.E. EXOERR IS NON-ZERO.

TST EXOERR : IS THE "EXIT ON ERROR" FLAG SET?
BNE 60$ : EXIT IF IT IS.

PRINT ERROR SUMMARY REPORTS IF NECESSARY.
THE FOLLOWING ROUTINE REPORTS ERRORS WITH NUMBER >>>>> ERROR 0 1915 <<<<<.
MOV @1915..,ERRNR : SET ERROR NUMBER FOR NEXT RTN.
JSR PC,REPMSR : REPORT ERROR SUMMARY IF NECESSARY.
CLR CTRLCF : INDICATE THAT WE COMPLETED THE TEST.
**SBTL Hardware Test - IDBIT**

**- Device Register ID Bit Test**

*This test verifies that the DUT Stat Register ID Bit reads as set.*

- **BGNS1**
  - **SETPRI @PRIO5,RO**
  - **T20:**
    - **MOV @PRIO5,RO**
    - **TRAP CI$P$R**

- **INUM ** **TNUM 1**
  - **INUM TNUM 1**
  - **TRAP CI$P$R**

- **MOV @TNUM,TSTNUM**
  - **SETUPTESTNUMBER**
  - **T20:**
    - **MOV @TNUM,TSTNUM**
    - **SETUPTESTNUMBER**

- **MOV 01,CTRLCF**
  - **INDICATETHATWEAREINATEST.**
  - **T20:**
    - **MOV 01,CTRLCF**
    - **INDICATETHATWEAREINATEST.**

- **MOV 01,ERRTP**
  - **SETUPOFFICIALERRORINDICATORINERRORTYPE.**
  - **T20:**
    - **MOV 01,ERRTP**
    - **SETUPOFFICIALERRORINDICATORINERRORTYPE.**

- **MOV @2001,ERRNR**
  - **SETUPERRORNUMBERINERRORTABLE.**
  - **T20:**
    - **MOV @2001,ERRNR**
    - **SETUPERRORNUMBERINERRORTABLE.**

- **MOV @EM2001,ERMSG**
  - **SETUPERRORMESSAGEFORTESTINERRORTABLE.**
  - **T20:**
    - **MOV @EM2001,ERMSG**
    - **SETUPERRORMESSAGEFORTESTINERRORTABLE.**

- **RESET THE DUT TO A KNOWN STATE. DO NOT REMOVE THE STATUS CODES FROM THE FIFO.**
  - **CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.**
  - **T20:**
    - **RESET THE DUT TO A KNOWN STATE. DO NOT REMOVE THE STATUS CODES FROM THE FIFO.**
    - **CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.**

- **THIS SUBROUTINE REPORTS ERRORS >>>>> 2001 <!<!<!<!<!</**
  - **THIS SUBROUTINE REPORTS ERRORS >>>>> 2001 <!<!<!<!<!<!</**

- **JSR PC,RESET**
  - **RESET THE D*U-11, REPORT ANY ERRORS FOUND.**
  - **FATAL RESET ERROR? YES, EXIT THE TEST.**
  - **BCC 601**
    - **FATAL RESET ERROR? YES, EXIT THE TEST.**

- **READ THE STAT REGISTER ID BIT AND VERIFY THAT IT IS CLEAR.**
  - **READ THE STAT REGISTER ID BIT AND VERIFY THAT IT IS CLEAR.**

- **MOV @FLSLA,RI**
  - **READ THE STAT REGISTER CONTENTS.**
  - **BNE 601**
    - **READ THE STAT REGISTER CONTENTS.**
    - **BNE 601**

- **MOV @OB18,RI**
  - **CHECK THE ID BIT.**
  - **BNE 601**
    - **CHECK THE ID BIT.**
    - **BNE 601**

- **MOV @2002,ERRNR**
  - **SET THE ERROR REPORT NUMBER TO 2002.**
  - **MOV @2002,ERRNR**
    - **SET THE ERROR REPORT NUMBER TO 2002.**

- **MOV @EM2001,ERMSG**
  - **SELECT THE PROPER ERROR REPORT ROUTINE.**
  - **MOV @EM2001,ERMSG**
    - **SELECT THE PROPER ERROR REPORT ROUTINE.**

- **ERROR**
  - **ERROR**
  - **ERROR**

- **CLR CTRLCF**
  - **INDICATE THAT WE COMPLETED THE TEST.**

- **ENDST**
  - **ENDST**

- **L10051:**
  - **L10051:**

- **TRAP CI$R$R**
  - **TRAP CI$R$R**

- **TRAP CI$R$R**
  - **TRAP CI$R$R**

- **TRAP CI$R$R**
  - **TRAP CI$R$R**

- **TRAP CI$R$R**
  - **TRAP CI$R$R**
6587
6588
6589
6590
6591
6592
6593
6594
6595
6596 033330
6597            033330
6598            033330
6599            033330 0127 0 000240
6600            033334 104441
6601 033336 000025
6602 033344 012767 177777 146734
6603 033360 012767 000001 150406
6604 033366 012767 012725 150376
6605
6606
6607
6608
6609
6610 033402 004677 16437A
6611 033406 103114
6612
6613
6614
6615
6616
6617
6618 033410 016705 146614
6619 033414 012700 000200
6620 033420 004677 170126
6621 033424 012700 177670
6622 033430 004677 170146
6623 033434 012700 000012
6624 033440 004677 164460
6625 033444 012705 177777
6626 033450 004677 167224
6627
6628
6629
6630
6631 033454 012703 000001
6632 033460 005004
6633 033462 012767 004376 150300 2;
6634 033470 012700 146534
6635 033474 001453
6636
6637
6638
6639
6640

SBTIL HARDWARE TEST TXENBI

** ********************************************************************

** TX_ENABLE (INACTIVE) TEST **
** THIS TEST VERIFYES THAT WHEN THE LINE UNDER TEST'S TX_ENABLE BIT IS **
** CLEAR, TRANSMISSION WILL NOT TAKE PLACE ON THAT LINE. **
** THIS TEST IS PERFORMED IN INTERNAL LOOPBACK, AND ON ALL ACTIVE LINES. **

******************************************************************** 

BGN1ST

;ETPRI @PRI05 ;ALLOW LTC INTERRUPTS. 

MOV @PRI05,RO
TRAP CISPRI

T21:

;TNUM == TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.

MOV TNUM,TSTNUM ;SET UP THE TEST NUMBER. (23)

MOV 0,1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.

MOV #1,ERRTYP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.

MOV #2301,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.

MOV #EM2301,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRBL.

MOV #ER9101,ERRBLK ;SELECT THE CORRECT ERROR REPORTING ROUTINE.

;RESET THE DUT TO A KNOWN STATE, REMOVE THE STATUS CODES FROM THE FIFO.

;CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.

;THIS SUBROUTINE REPORTS ERROR >>>> 2301 <<<.

;DISABLE TRANSMISSION ON EACH ACTIVE LINE.

;DISABLE TRANSMISSION ON EACH ACTIVE LINE.

;FAIL TO SET UP THE LINE BIT MAP FOR CHANNEL 0.

;CHECK IF THE LINE IS ACTIVE.

;SKIP TESTING THIS LINE IF IT IS INACTIVE.

;CLEAR THE TX_ENABLE BIT IN TBUFFAD REGISTER.

;SELECT THE LINE UNDER TEST.

;VERIFY IT IS CLEAR, REPORT ERROR IF SET.
1: ; VERIFY ALL ACTIVE LINES HAVE BEEN TESTED.

2: ; VERIFY DATA BYTE (ASCII 'LF') TO THE OUTPUT FIFO.
; WAIT FOR A TX_ACTION TO BE RETURNED. REPORT ERROR IF A TX_ACTION
; IS FOUND BEFORE TIME-OUT OCCURS.

3: MOV @Z303.,ERABR
 ; SET ERROR NUMBER TO 2303.

4: MOV @Z12.,DFDATA
 ; WRITE THE DATA BYTE TO THE DUT's OUTPUT FIFO.

5: MOV @Z7000,R1
 ; TEST BIT 7, TIMEOUT OF 10 MILLI SECS.

6: MOV CSRA,R2
 ; PASS THE ADDRESS OF THE REGISTER TO TEST.

7: JSR PC,WAIBUS
; WAIT FOR TX_ACTION TO COME BACK.

8: BCS 41
 ; GO REPORT ERROR IF A TX_ACTION FOUND.

9: MOV @Z502.,ERABR
 ; SET ERROR NUMBER TO 2305.

10: MOV BRABUS,R2
 ; READ THE DATA FROM THE FIFO.

11: BPL 61
 ; SKIP ERROR REPORT IF DATA ISN'T THERE.

12: MOV @Z41.,ERABR
 ; SET ERROR NUMBER TO 2304.

13: MOV @Z203.,ERABR
 ; SET ERROR NUMBER TO 2305.

14: ; "TX_ENABLE BIT BAD ON LINE: NN".

15: ; EXIT THE TEST, WITH THE TEST FAILED MESSAGE.

16: BEQ 60
; IF EXTENDED ERROR REPORTING HAS NOT BEEN REQUESTED.

17: ; CLEAR THE CARRY BIT PRIOR TO ROTATION.

18: ; SHIFT THE BIT MAP FOR THE NEXT LINE.

19: ; INCREMENT THE LINE NUMBER COUNTER.

20: ; HAVE ALL THE LINES BEEN TESTED?

21: ; NO: BRANCH TO TEST THE NEXT LINE.

22: ; CLA CIRLCF ; INDICATE THAT WE ARE NOT WITHIN A TEST.
**HARDWARE TEST**

**SBITL** HARDWARE TEST TXENBA

**TX ENABLE (ACTIVE) TEST**

**TX_ENABLE**

**TSTW** TX_ENABLE Bit is set in the appropriate LINE REGISTER, Transmission will take place on that LINE.

**TSTW** TX_ENABLE bit is performed in internal loopback, and on all active lines.

**BGMTST**

SETPRI @PRI05 ; ALLOW LTC INTERRUPTS.

**MOV** @PRI05,RO

**T22:**

**TRAP** C1SPRI

**THUM** **THUM : = 1 ; INCREMENT THE ASSEMBLY TIME TEST COUNTER.

**MOV** @THUM, TSTNUM ; SET UP THE TEST NUMBER.

**MOV** @1, TSTLCF ; INDICATE THAT WE ARE IN A TEST.

**MOV** @1, ERRTYE ; SET ERROR TYPE AS FATAL IN ERROR TABLE.

**MOV** @2401, ERRNBR ; SET THE FIRST ERROR NUMBER IN ERROR TABLE.

**MOV** @ER2401, ERRMSG ; SET ERROR MESSAGE ADDRESS IN ERRBL.

**MOV** @ER9101, ERRBLK ; SELECT THE CORRECT ERROR REPORTING ROUTINE.

**7020**

**SET THE DUT TO A KNOWN STATE, REMOVE THE STATUS CODES FROM THE FIFO.

**CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.

**THIS SUBROUTINE REPORTS ERROR >>>> 2401 <<<<.

**BCC** 601 ; SET FAILURE?, ABORT THIS TEST.

**7050**

**SET INTERNAL LOOPBACK ON ALL ACTIVE LINES.

**SET LPR ON ALL LINES TO 58.4K BAUD, 8 BITS PER CHARACTER, ODD PARITY.

**2 STOP BITS.

**DISABLE TRANSMITTERS ON ALL LINES.

**7200**

**MOV** ACTLNS,R5 ; PASS THE ACTIVE LINE BIT MAP.

**MOV** @200,RO ; PASS THE LCTRL CONTENTS.

**JSR** PC, LCTRL ; INITIALISE THE LCTRL REGISTERS.

**MOV** @177670,RO ; PASS THE LPR CONTENTS.

**JSR** PC, LPR ; INITIALISE THE LPR REGISTERS ON ALL LINES.

**MOV** @10, R4 ; PASS DELAY TIME OF 10 MILLI-SECONDS.

**JSR** PC, DELAY ; WAIT FOR LNCIR AND LPR REG TO BE UPDATED.

**MOV** @MAPLNS,H5 ; PASS THE BIT MAP CORRESPONDING TO ALL LINES.

**JSR** PC, INDEX ; DISABLE TRANSMITTERS ON ALL LINES.

**7300**

**TEST ALL ACTIVE LINES INDIVIDUALLY.

**ENABLE TRANSMISSION ON EACH ACTIVE LINE.

**7330**

**MOV** @1,R3 ; SET UP THE LINE BIT MAP FOR CHANNEL 0.

**CLR** R4 ; CLEAR THE LINE NUMBER COUNTER.

**MOV** @2402, ERRNBR ; SET THE ERROR NUMBER TO 2402.

**BIT** R3, ACTLNS ; CHECK IF THE LINE IS ACTIVE.

**BEQ** B1 ; SKIP TESTING THIS LINE IF IT IS INACTIVE.

**7359**

**SELECT THE LINE UNDER TEST.

**SET THE TX ENABLE BIT IN IBUFFAD2 REGISTER.

**VERIFY IT IS SET, REPORT ERROR IF CLEAR.

**7433**
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0800</td>
<td>034214</td>
</tr>
<tr>
<td>0801</td>
<td>054220</td>
</tr>
<tr>
<td></td>
<td>034220</td>
</tr>
<tr>
<td></td>
<td>034220</td>
</tr>
</tbody>
</table>

604: CLR CTRL CF
ENDTST

;INDICATE THAT WE ARE NOT WITHIN A TEST.

L10053: TRAP C#ETST
.SBTL HARDWARE TEST - INTA -

* INTERRUPT TEST *

* THIS TEST VERIFIES THAT THE DEVICE UNDER TEST (DUT) WILL GENERATE
* RECEIPTION AND TRANSMISSION INTERRUPTS CORRECTLY. THIS TEST DOES
* NOT DEPEND ON THE USE OF THE SERIAL LINE TRANSMISSION OR RECEIPTION
* CAPABILITIES OF THE DUT. THE LINES ARE PUT IN INTERNAL LOOPBACK
* TO MINIMIZE ANY EXTERNAL EFFECTS THAT COULD BE CAUSED ON DEVICES.
* ATTACHED TO THE SERIAL LINES.

BGN1ST

SETPRI @PRIO5, 0

; ALLOW THE LTC TO INTERRUPT.

I23: NOP

MOV @PRIO5, RO

; TRAP C15PR1

TNUM ++ TNUM . 1

; INCREMENT THE ASSEMBLY TIME TEST COUNTER.

MOV @TNUM.TSTNUM, @TNUM.TSTNUM

; SET UP THE TEST NUMBER.

MOV 0, CTRLC;

; INDICATE THAT WE ARE IN A TEST.

MOV 0, ERRYP;

; SET ERROR FAULT ERROR TYPE IN ERROR TABLE.

MOV 0, ERRWARN;

; SET FIRST ERROR REPORT NUMBER IN ERROR TABLE.

MOV @EM2601..ERRMGR, @EM2601..ERRMGR

; SET TEST ERROR MESSAGE IN ERROR TABLE.

; RESET THE DUT TO A KNOWN STATE, DO NOT REMOVE THE STATUS CODES FROM THE FIFO.

; CLEAR TX AND RX INTERRUPT ENABLE BIT IN THE CSR.

; THIS SUBROUTINE REPORTS ERRORS FROM >2601 THRU 2602 <<.<

; JSR PC,RESEET;

; RESET THE DMU-11, REPORT ANY ERRORS FOUND.

BCT @BCTR;

; SKIP AROUND ABORTING TEST IF NO ERROR FOUND.

JMP 0, 0;

; ABORT TEST IF ERROR FOUND DURING RESEET.

MOV 0, 2603..ERRMGR;

; SET ERROR REPORT NUMBER TO 2603.

; ENABLE TRANSMITTERS ON ALL LINES.

MOV @MAPLIN, RS;

; PASS ACTIV LINE BIT MAP.

JSR PC, TXENBL;

; ENABLE TRANSMISSION ON ALL LINES.

; TEST RECEIPTION INTERRUPTS.

; SET UP FOR RX AND TX INTERRUPTS.

; RX INTERRUPT SERVICE ROUTINE INPUTS A CHAR AND COUNTS THE INTERRUPT.

; TX INTERRUPT SERVICE ROUTINE COUNTS TX INTERRUPTS.

CLR RXINTC;

; CLEAR THE RX INTERRUPT COUNTER.

CLR RXINFT;

; CLEAR THE RX INTERRUPT FLAGS.

CLR TXINTC;

; CLEAR THE TX INTERRUPT COUNTER.

MOV @EMBUFF.BUFFPTR, @EMBUFF.BUFFPTR

; LOAD THE BUFFER PTR WITH THE BUFFER BASE ADDR.

SETVEC AXVECA, @AXINPT, @PRIO6;

; SET UP INTERRUPT VECTOR TO CATCH RX INT.

MOV @PRIO6, SP;

MOV @AXINPT, SP;

MOV @AXVECA, SP;

MOV 0, SP;

TRAP C15VEC;

ADD @T0, SP;

SETVEC TXVECA, @TACHTXT, @PRIO6;

; SET UP INTERRUPT VECTOR TO CATCH TX INT.
I14

HARDWARE TEST - INTA -

034376  016746  145632
034402  012746  000003
034406  106877  000010
034410  062706  000010
034414  002700  000200
034420  104441

6849  034414
6850

6851
6852

6853
6854

6855  034422  004767  165624
6856  034426  012704  000004
6857  034432  004767  163666
6858  034436  004767  165550
6859

6860
6861
6862
6863  034442  005767  145646
6864  034446  001017

6865

6866
6867

6868  034450  012701  013310
6869  034454  032777  000200  145560
6870  034458  001416  000000  145560
6871  034464  012701  013222
6872  034470  032777  100000  145546
6873  034476  001410
6874  034500  012701  013131
6875  034504  000045
6876

6877
6878

6879  034506  005767  145604
6880  034512  100014
6881  034514  012701  013404

6882
6883
6884

6885  034520
6886
6887
6888
6889
6890

6891  034530  032767  000100  145466
6892  034536  001002
6893  034540  000167  000556
6894
6895
; VERIFY THAT NO TX INTERRUPTS HAVE BEEN GENERATED SO FAR IN THIS TEST.

104: MOV  TXINTC,R2  ;LOAD 0 OF TX INTERRUPTS FOR ER0504 RIN.
BEQ  12$  ;SKIP ERROR IF NO TX INTERRUPTS.

; REPORT "TX INTERRUPTS(S) RECEIVED WITH TX INTERRUPTS DISABLED."
MOV  ER0504,R1  ;SET UP MESSAGE ADDR FOR INDIRECT PRINT.
ERADC  2604,EM2601,ER0504;  >>>>> ERROR 02604 <<<..
TRAP  C1ERDF
    WORD  2604
    WORD  EM2601
    WORD  ER0504

; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED

908: BIT  @BIT06,OPTION  ;EXIT WITH TEST FAILURE MESSAGE IF
BNE  16$  ;NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
JMP  14$  ;NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED

; CLEAN OUT THE INTERRUPT VECTORS USED IN THIS TEST.

12$: SETPRI  @PRI06  ;DISABLE DEVICE INTERRUPTS.
MOV  @PRI06,RO  
TRAP  C1SPRI

; TEST TRANSMISSION INTERRUPTS.
; SET UP FOR RX AND TX INTERRUPTS:
; RX INTERRUPT SERVICE ROUTINE COUNTS RX INTERRUPTS.
; TX INTERRUPT SERVICE ROUTINE COUNTS THE INTERRUPT AND SETS FLAGS.

18$: CLR  RXINTC  ;CLEAR THE RX INTERRUPT COUNTER.
CLR  TXINTC  ;CLEAR THE TX INTERRUPT COUNTER.

; TX INTERRUPT SERVICE ROUTINE COUNTS THE INTERRUPT AND SETS FLAGS.

1E$: CLR  RXINTC  ;CLEAR THE RX INTERRUPT COUNTER.
CLR  TXINTC  ;CLEAR THE TX INTERRUPT COUNTER.
SETVEC  RXVECA,0,CACHRX,0,PRI06  ;SET UP INTERRUPT VECTOR TO CATCH RX INT.
MOV  @PRI06,-(SP)
MOV  @CACHRX,-(SP)
MOV  RXVECA,-(SP)
MOV  @S,-(SP)
TRAP  C1SVEC
ADD  @10,SP

SETVEC  TXVECA,0,TXINTR,0,PRI06  ;SET UP INT VECTOR TO TX INT ROUTINE.
MOV  @PRI06,-(SP)
MOV  @TXINTR,-(SP)
MOV  TXVECA,-(SP)
MOV  @S,-(SP)
TRAP  C1SVEC
ADD  @10,SP

SETPRI  @PRI04  ;ALLOW DEVICE INTERRUPTS.
MOV  @PRI04,RO
6931 034720 104441 ; VERIFY THAT THE TX_ACTION BIT IS CLEAR.
6932           ; INITIALIZE THE LOOP COUNTER.
6933 MOV @18,R5 ; SET 100 MS TIME OUT.
6934 MOV @100,R1 ; SELECT TX_ACTION BIT TO TEST.
6935 MOV @BIT15,R2 ; PASS OUT CSR AS THE WORD TO TEST.
6936 MOV CSRA,R4 ; WAIT FOR TX_ACTION TO BE SET.
6937 MOV @100000,000000 ; WAIT UPTO 100 MS FOR TX_ACTION SET.
6938 JSR PC,MLOOP ; IF TIME-OUT, CONSIDER TX_ACTION CLEAR.
6939 CLR R3 ; NOW, WAIT FOR TX_ACTION CLEAR.
6940 JSR PC,MLOOP ; WAIT UPTO 100 MS FOR TX_ACTION CLEAR.
6941 BCC 20 ; IF TIME-OUT, REPORT TX_ACTION WON'T CLEAR.
6942 DEC R5 ; DECREMENT THE TX_ACTION SET COUNTER.
6943 BNE 14 ; LOOP IF NOT TOO MANY TX ACTIONS FOUND.
6944 ; REPORT "TX_ACTION SET REPEATEDLY AFTER RESET, NO DATA SENT."
6945 MOV @EM2607,R1 ; SELECT ERROR MESSAGE.
6946 MOVS @EM2608,R1 ; SELECT TX_ACTION STUCK SET MSG.
6947 BR 18 ; GO TO REPORT THE ERROR.
6948 ; ERROR #2605 TRAP C1ERDF .....
6949 035002 104455 ; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED.
6950           ; EXIT WITH TEST FAILURE MESSAGE IF
6951           ; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
6952           ; ENABLE TX_INTERRUPTS FOR THE TX_INT TESTING.
6953           ; VERIFY THAT NO INTERRUPTS OCCUR WITH TX_ACTION CLEAR.
6954 035012 032767 000100 145204 ; BIT @BIT06,OPTION
6955 BEQ 34 ; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
6956 JSR PC,TXIE1 ; ENABLE TX_INTERRUPTS FOR THE TX_INT TESTING.
6957 BR 24 ; GO TO TEST WITH TX_ACTION SET.
6958           ; REPORT "TRANSMIT INTERRUPT TEST ERROR:"
6959 035030 004767 166000 ; PASS 50 MS TIME TO THE DELAY ROUTINE.
6960 MOV @50,R4 ; DELAY 50 MILLISECONDS TO ALLOW INTS TO OCCUR.
6961 JSR PC,DELAY ; TEST THE TX INTERRUPT COUNT.
6962 MOV @EM2607,R1 ; SELECT MESSAGE IN CASE TX_INT_FLAG CLEAR.
6963 JSR PC,DELAY ; TEST THE TX INTERRUPT FLAGS.
6964 MOV @EM2609,R1 ; TX_FLAG IS SET, SELECT PROPER ERROR MESSAGE.
6965 JSR PC,DELAY ; REPORT "TRANSMIT INTERRUPT TEST ERROR:"
6966 035052 012701 035058 012701 013526 ; REPORT "TRANSMIT INTERRUPT TEST ERROR:"
6967 035056 005767 145246 ; PASS 50 MS TIME TO THE DELAY ROUTINE.
6968 TST IXINT ; SELECT MESSAGE IN CASE TX_INT_FLAG CLEAR.
6969 TST IXINT ; TEST THE TX_INTERRUPT FLAGS.
6970 JSR PC,DELAY ; REPORT "TRANSMIT INTERRUPT TEST ERROR:"
6971 MOV @EM2606,EM2606.EROS03; TRAP C1ERDF .....
6972 035070 104455 ; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED.
6973          ; EXIT WITH TEST FAILURE MESSAGE IF
6974          ; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
6975 035100 032767 000100 145116 ; BIT @BIT06,OPTION
6976 BEQ 32 ; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
; PREPARE TX INTERRUPT COUNTER AND FLAGS.
CLR TXINIC ; CLEAR THE TX INTERRUPT COUNTER.
CLR TXINF ; CLEAR THE TX INTERRUPT FLAGS.

; SET UP LINE PARAMETERS FOR TRANSMISSION.
MOV #MAPLSNS,R5 ; PASS ACTIVE LINES BIT MAP.
MOV #200,R0 ; PASS INERT STATE, INTERNAL LOOPBACK.
JSR PC,WTULNC ; DISABLE RECEPTION AND DMA, ETC. ON DUT.
MOV 0156430,R0 ; SPECIFY 9600BPS,1STOP,NO PARITY,8BITS/CHAR.
JSR PC,WTULPR ; WRITE TO ALL LPR REGISTERS.

; SEND A NULL CHAR TO EACH LINE.
MOV IESTAT,R1 ; SET UP THE STATE OF THE INTERRUPT ENABLE BITS.
CLR R2 ; CLEAR THE LINE COUNTER.
MOV R1,BCSRA ; SET UP THE LINE NUMBER AND INTERRUPT ENABLE
BITS IN THE CSR.
MOVB #0,FRDATA ; SEND A NULL CHARACTER TO THE OUTPUT FIFO.
INC R1 ; NEXT CSR CONTENTS.
INC R2 ; NEXT LINE.
CMP R2,#NUMULNS ; IF ALL LINES HAVE NOT BEEN SERVICED THEN
BLT 25$ ; BRANCH.

; DELAY 250 MILI-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 TXINIC ; CHECK THE TX INTERRUPT COUNTER.
BNE 26$ ; SKIP THE FOLLOWING ERROR IF WE GOT TX INT.

; DETERMINE THE REASON THAT WE RECEIVED NO INTERRUPTS.
MOV #EM2610,R1 ; SET UP MSG IN CASE "TX_ACTION IS SET".
TST BCSRA ; CHECK THE OUT CSR.
BMI 28$ ; GO TO REPORT ERROR IF TX_ACTION IS SET.
MOV #EM2611,R1 ; SET UP "TX_ACTION NOT SET" MESSAGE.
BR 28$ ; GO AND REPORT THE ERROR.

; CHECK TO VERIFY THAT TX_ACTION WAS SET FOR EACH INTERRUPT.
TST TXINF ; CHECK THE TX INTERRUPT FLAGS.
BPL 30$ ; SKIP IF ERRORS IF TX_ACTION CLR FLAG IS CLEAR.
MOV #EM2609,R1 ; SET UP TX INT WITH "TX_ACTION CLR" MSG.

; REPORT "TRANSMIT INTERRUPT TEST ERROR:".
ERRDF 2607,EM2606,ER0503; ; ERROR 2607 "<<<<<
TRAP CIERDF
.WORD 2607
.WORD EM2606
7052                035254 016136
7053                .WORD  ER0503
7054                ;EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED
7055                ;
7056                ;BIT  0BIT06,OPTION ;EXIT WITH TEST FAILURE MESSAGE IF
7057                BEQ     32$; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
7058                ;
7059                ;VERIFY THAT NO RX INTERRUPTS HAVE BEEN GENERATED SO FAR IN THIS TEST.
7060                ;
7061                30$:     MOV  RXINTC,R2 ;LOAD # OF RX INTERRUPTS FOR ER0504 RTN.
7062                BEQ     32$; SKIP ERROR IF NO RX INTERRUPTS.
7063                MOV  EM0525,R1 ;SET UP MESSAGEADR FOR INDIRECT PRINT.
7064                ;REPORT "RX INTERRUPTS(S) RECEIVED WITH RX INTERRUPTS DISABLED."
7065                ERDF  2608,EM2606,ER0504;
7066                ;>ERROR #2608 <<<< TRAP C\$ERDF
7067                .WORD  2608
7068                .WORD  EM2606
7069                .WORD  ER0504
7070                ;
7071                ;DISABLE INTERRUPTS AND CLEAN OUT THE INTERRUPT VECTORS USED IN THIS TEST.
7072                ;
7073                32$:     CLR  R1 ;CLEAR BOTH TRANSMITTER
7074                JSR     PC,TXIE0 ;INTERRUPT ENABLE AND RECEIVER
7075                JSR     PC,RXIE0 ;INTERRUPT ENABLE BITS IN THE DUT CSR.
7076                34$:     SETPRI #PRI06 ;DISABLE DEVICE INTERRUPTS.
7077                MOV  #PRI06,RO
7078                TRAP  CISPRI
7079                CLRVEC RXVECA ;RETURN RX INT VECTOR TO UNUSED POOL.
7080                MOV  RXVECA,RO
7081                TRAP  CICVEC
7082                CLRVEC TXVECA ;RETURN TX INT VECTOR TO UNUSED POOL.
7083                MOV  TXVECA,RO
7084                TRAP  CICVEC
7085                ;
7086                50$:     CLR  CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.
7087                SETPRI #PRI07 ;DISABLE ALL INTERRUPTS.
7088                MOV  #PRI07,RO
7089                TRAP  CISPRI
7090                ENDTST
7091                L10054: TRAP C\$ESTT
**HARDWARE TEST**

**BR LEVEL TEST B**

This test verifies that the device under test (DUT) will generate reception and transmission interrupts at the correct BR level.

This test does not depend on the use of the serial line transmission or reception capabilities of the DUT. The lines are put in internal loopback to minimize any external effects that could be caused on devices attached to the serial lines.

---

**BGNST**

**SETPRI @PRI05 : ALLOW LTC INTERRUPTS.**

**MOV @PRI05,RO**

**TRAP C!SPRI**

**TNUM == TNUM+1**

**MOV @TNUM,TSTNUM**

**SET UP THE TEST NUMBER.**

**MOV $01.ERRTP**

**SET ERROR TYPE AS FATAL IN ERROR TABLE.**

**MOV $0301.ERRNBR**

**SET THE FIRST ERROR NUMBER IN ERROR TABLE.**

**MOV @EM3001.ERRMSG**

**SET ERROR MESSAGE ADDRESS IN ERRBL.**

**CLR ERRMRF**

**INITIALIZE THE "REPORT ERROR SUMMARY" FLAGS.**

**RESET THE DUT TO A KNOWN STATE. DO NOT REMOVE THE STATUS CODES FROM THE FIFO.**

**CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.**

**THIS SUBROUTINE REPORTS ERRORS FROM >>>>>> 3001 THRU 3002 <<<<<<.**

**JSR PC,RESET**

**RESET THE DMU-11, REPORT ANY ERRORS FOUND.**

**BSC 2**

**SKIP AROUND ABORTING TEST IF NO ERROR FOUND.**

**JMP 601**

**ABORT TEST IF FATAL ERROR FOUND DURING RESET.**

**MOV $0303.ERRNBR**

**SET THE ERROR REPORT NUMBER TO 3003.**

**ENABLE TRANSMITTERS ON ALL LINES.**

**MOV @MAPLNS,R5**

**PASS ACTIVE LINE BIT MAP.**

**JSR PC,TXENBL**

**ENABLE TRANSMISSION ON ALL LINES.**

**GENERATE A TRANSMISSION INTERRUPT REQUEST.**

**PROCESSOR PRIORITY SHOULD BE AT 7 DISABLING INTS.**

**SETPRI @PRI07 : DISABLE ALL INTERRUPTS.**

**MOV @PRI07,RO**

**TRAP C!SPRI**

**SETVEC TXVECA, &TXINTR, @PRI07 : SET UP INTERRUPT VECTOR TO CATCH TX INT.**

**MOV @PRI07,-SP**

**MOV @TXINTR,-SP**

**MOV TXVECA,-SP**

**MOV @3,-SP**

**TRAP C!VEC**

**ADD @10,SP**

**SET UP DUT FOR TRANSMISSION INTERRUPTS:**

**SET UP INTERNAL LOOPBACK:**

**SET UP LINE PARAMETERS FOR TRANSMISSION:**
MOV @MAPLNS,R5 ; PASS ACTIVE LINES BIT MASK.
JSR PC,MTM,NC ; DISABLE RECEPTION AND DMA, ETC. ON OUT.
MOV @156430,RO ; SPECIFY 9600BPS, ISOTOP, NO PARITY, 8BIT/CHAR.
JSR PC,WTMLPR ; WRITE INTO ALL LPR REGISTERS.

; SEND A NULL CHAR TO EACH LINE.

MOV IESTAT,R1 ; SET UP THE STATE OF THE INTERRUPT ENABLE BITS.
MOV R1,R,BCSR ; SET UP THE LINE NUMBER AND INTERRUPT ENABLE
MOV @,bpsdata ,BCSR ; BITS IN THE CSR.

INC R1 ; SEND A NULL CHARACTER TO THE OUTPUT FIFO.
CMP R1,#0x0 ; IF ALL LINES HAVE NOT BEEN SERVICED THEN
BLT 51 ; BRANCH.

; DELAY 50 MS TO ALLOW TIME FOR THE INTERRUPT TO BE Generated.
MOV #50,R4 ; PASS 50 MS TIME TO THE DELAY ROUTINE.
JSR PC,DELAY ; DELAY 50 MILLISECONDS.

; GENERATE A RECEPTION INTERRUPT REQUEST.

SETVEC RXVECA,RXRBRAT,R,PRI07 ; SET UP INTERRUPT VECTOR TO CATCH RX INT.
MOV #PRI07,SP ;
MOV #0x0,SP ;
MOV RXVECA,SP ;
MOV #3,SP ;
TRAP CISPRI ;
ADD #10,SP ;

; SET UP FOR THE LOOP WHICH TESTS THE INTERRUPT BR LEVELS.

MOV #340,R5 ; SET UP THE PRIORITY LEVEL TO 7.
CLR R3 ; CLEAR THE RX PRIORITY STORE AND FLAGS.
CLR R2 ; CLEAR THE TX PRIORITY STORE AND FLAGS.

; ENABLE RX AND TX INTERRUPTS.
; PROCESSOR PRIORIT SHOULD BE AT 7 DISABLING THE INTERRUPTS.

JSR PC,AXIE1 ; ENABLE RECEIVER INTERRUPTS.
JSR PC,AXIE1 ; ENABLE TRANSMITTER INTERRUPTS.

; LOOP, LOWERING THE PROCESSOR PRIORITY UNTIL THE DUT INTERRUPTS ON RX AND TX.

; DETERMINE IF ANY RX DUT INTERRUPTS OCCURRED.
; LOG THE PROCESSOR PRIORITY FOR THE RX INTERRUPT IF FIRST RX INT.
IST RXINTC  "CHECK THE RECEIVE INTERRUPT COUNTER.
BEQ 8;  "SKIP THE PRIORITY LOG IF NO RX INT OCCURRED.

IF THIS IS THE FIRST RX INTERRUPT, LOG THE PRIORITY.

IST R3  "CHECK THE RX PRIORITY STORE AND FLAGS.
BNE 8;  "GOTO TEST FOR TX INTS IF NOT THE FIRST RX INT.
MOV R5,R3  "LOG THE PRESENT PRIORITY IN THE TX PRIOR STORE.
BIS #115,R3  "SET THE TX INTERRUPT ROUTINE FLAGS.
MOV RXINTF,R0  "GET THE RX INTERRUPT ROUTINE FLAGS.
BIC #157777,R0  "CLEAR ALL BUT THE TX INT ERRH FLAG.
BIS R0,R3  "IF TX INT ERROR, SET BIT 14 OF THE PRIOR FLAGS.

DETERMINE IF ANY TX OUT INTERRUPTS HAVE OCCURRED.
LOG THE PRESENT PROCESSOR PRIORITY IF THIS IS THE FIRST TX INTERRUPT.

IST TXINTC  "CHECK THE TRANSMIT INTERRUPT COUNTER.
BEQ 10;  "SKIP THE PRIORITY LOG IF NO TX INT OCCURRED.
MOV  "IF THIS IS THE FIRST TX INTERRUPT, LOG THE PRIORITY.

IST R2  "CHECK THE TX PRIORITY STORE AND FLAGS.
BMI 10;  "SKIP THE LOGGING IF NOT FIRST TX INTERRUPT.
MOV R5,R2  "LOG THE PRESENT PRIORITY IN THE TX PRIOR STORE.
BIS #115,R2  "SET THE TX INT HAS OCCURRED FLAG.

SELECT NEXT PROCESSOR PRIORITY.
TEST FOR BOTH RX AND TX INTERRUPTS HAVING OCCURRED, LOOP IF NOT.

SUB #40,R5  "DECREMENT PRIORITY LEVEL BY ONE.
BLT 12;  "GOTO CHECK FOR ERRORS IF BELOW PRIOR LEVEL ZERO.
BIT R2,R3  "AND PRIOR FLAGS TOGETHER, ALTER NONE OF THEM.
BPL 6;  "LOOP IF RX AND TX INTS HAVEN'T BOTH OCCURRED.

DISABLE INTERRUPTS AND CLEAR INTERRUPT VECTORS.

SETPRI #PRI07  "DISABLE ALL INTERRUPTS.

VERIFY THAT RX AND TX INTERRUPTS OCCURRED.
AT THE PROPER BR LEVEL, AND
IN THE PROPER ORDER, DETERMINE IF TX INTERRUPT OCCURRED.

IST R2  "DETERMINE WHETHER TX INT OCCURRED OR NOT.
BMI 16;  "SKIP THESE ERRORS IF TX INT OCCURRED.

DETERMINE REASON THAT NO TX INT OCCURRED.
MOV #EM2610,R1  ; SELECT "NO RX INT FROM RX,DATA,AVAL" MSG.

ASR BCSRA     ; CHECK RX,DATA.AVAL BIT OF THE DUT CSR.

BMI 147       ; SKIP RX,DATA.AVAL CLR MSG IF BIT IS SET.

MOV #EM2611,R1  ; SELECT "TX,DATA CLEAR AFTER CHAR SENT" MSG.

REPORT "INTERRUPT BR LEVEL TEST ERROR:" ERRDF 3003,EM3001,ER0503; >>>>> ERROR #3003 <<<<<.

TRAP CIERDF .WORD 3003 .WORD EM3001 .WORD ER0503

; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED

; BIT #BIT06,OPTION ; EXIT WITH TEST FAILURE MESSAGE IF

BED 261  ; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED.

BR 184  ; SKIP THE BR LEVEL CHECK, NO TX INT OCCURRED.

; VERIFY THAT THE TX INTERRUPT WAS AT THE PROPER BR LEVEL.

; CALCULATE THE BR LEVEL

BIC #177400,R4  ; THAT THE TRANSMIT

ASR R4  ; INTERRUPT

ASR R4  ; REQUESTED AT WHICH

ASR R4  ; IS ONE GREATER THAN

AT R4  ; THE PROCESSOR PRIORITY

ASR R4  ; LEVEL AT WHICH THE

INC R4  ; TRANSMIT INTERRUPT OCCURRED.

MOV #RRLEV,RS  ; GET THE EXPECTED INTERRUPT BR LEVEL.

CMPB R4,RS  ; COMPARE THE INTERRUPT BR LEVEL WITH EXPECTED.

BEQ 184  ; SKIP THE ERROR IF BR LEVEL IS CORRECT.

; REPORT "TX INTERRUPT GENERATED AT WRONG BR LEVEL: ..."

MOV #EM3003,R1  ; SELECT THE ERROR MESSAGE FOR THE ERROR CALL.

ERRDF 3004,EM3001,ER3001; >>>>> ERROR #3004 <<<<<.

TRAP CIERDF .WORD 3004 .WORD EM3001 .WORD ER3001

; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED

; BIT #BIT06,OPTION ; EXIT WITH TEST FAILURE MESSAGE IF

BEQ 261  ; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED.

; DETERMINE IF RX INTERRUPT OCCURRED.

; CHECK THE RX INT OCCURRED FLAG.

BMI 224  ; SKIP THESE ERRORS IF RX INT OCCURRED.

; DETERMINE REASON THAT NO RX INT OCCURRED.

MOV #EM2612,R1  ; SELECT "NO RX INT FROM RX,DATA,AVAL" MSG.

BIT #BIT7,BCSRA ; CHECK RX,DATA.AVAL BIT OF THE DUT CSR.

BNE 201  ; SKIP RX,DATA.AVAL CLR MSG IF BIT IS SET.

MOV #EM3002,R1  ; SELECT "NO RX,DATA.AVAL AFTER RESET MSG.

MOV #EM2612,R1  ; SELECT "NO RX INT FROM RX,DATA,AVAL" MSG.

BIT #BIT7,BCSRA ; CHECK RX,DATA.AVAL BIT OF THE DUT CSR.

BNE 201  ; SKIP RX,DATA.AVAL CLR MSG IF BIT IS SET.

MOV #EM3002,R1  ; SELECT "NO RX,DATA.AVAL AFTER RESET MSG.

MOV #EM2612,R1  ; SELECT "NO RX INT FROM RX,DATA,AVAL" MSG.

BIT #BIT7,BCSRA ; CHECK RX,DATA.AVAL BIT OF THE DUT CSR.

BNE 201  ; SKIP RX,DATA.AVAL CLR MSG IF BIT IS SET.

MOV #EM3002,R1  ; SELECT "NO RX,DATA.AVAL AFTER RESET MSG.
```
201: ERRDF 3005,EM3001,EROF<3; TRAP CIERDF

[*] EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED

BIT #0BIT06,OPTION ;EXIT WITH TEST FAILURE MESSAGE IF
BEQ 261 ;NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED
BR 241 ;SKIP THE BR CHECK IF NO RX INT OCCURRED.

VERIFY THAT THE RX INTERRUPT WAS AT THE PROPER BR LEVEL.

;BR 241
;CALCULATE THE BR LEVEL
MOV R3,R4
BIC #177400,R4
THAT THE RECEIVE
ARU R4
INTERUPT WAS
ARU R4
REQUESTED AT, WHICH
ARU R4
IS ONE GREATER THAN
ARU R4
THE PROCESSOR PRIORITY
BR 241
LEVEL AT WHICH THE
INC R4
RECEIVE INTERRUPT OCCURRED.

;BR 241
;GET THE EXPECTED INTERRUPT BR LEVEL.
MOV BALEVL,RS
CMPB R4,RS
;COMPARE THE INTERRUPT BR LEVEL WITH EXPECTED.
BEQ 241 ;SKIP THE ERROR IF BR LEVEL IS CORRECT.

;REPORT "RX INTERRUPT GENERATED AT WRONG BR LEVEL: . . . ;
ERRDF 3005,EM3001,ER3001; TRAP CIERDF

[*] EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED

BIT #0BIT06,OPTION ;EXIT WITH TEST FAILURE MESSAGE IF
BEQ 261 ;NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED

;TEST FOR INTERRUPTS OCCURRING IN THE PROPER ORDER.

BIT #0BIT14,R3 ;CHECK THE IMPROPER INT ORDER ERROR FLAG.
BEQ 261 ;SKIP ERROR REPORT IF ERROR DID NOT OCCUR.

;REPORT "TX INTERRUPT GIVEN PRECEDENCE OVER SIMULTANEOUS RX INT.
MOV #EM3005,RI ;SELECT THE ERROR MESSAGE FOR INDIRECT PRINT.
ERRDF 3005,EM3001,ER3001; TRAP EIERDF

;CLEAN UP, EXIT THE TEST.

JSR PC,IXIEO ;CLEAR TRANSMITTER INTERRUPTS.
JSR PC,IXIEO ;CLEAR RECEIVER INTERRUPTS.
```
7302 036320 005067 143774 604:    CPI CPILOF
7303 036312 036312 012700 000340 ;INDICATE THAT WE ARE NOT WITHIN A TEST.
7304 036316 104441 ;DISABLE ALL INTERRUPTS.
7305 036320 ENOTST
7306 036320 104401

L10055: TRAP C1ETST
MOV @PRI07,RO
TRAP C1SPRI
THIS TEST VERIFIES THAT A REQUEST TO THE OUT TO REPORT BMP STATUS CODES IS COMPLIED WITH, WITHIN THE SPECIFIED TIME. ALL ACTIVE LINES ARE TESTED.

BGNST
SETPRI @PRI05 ;ALLOW LTC INTERRUPTS.
MOV @PRI05.0
TRAP C@SPRI

INUM == INUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV @INUM,TSTNUM ;SET THE TEST NUMBER.
MOV 0-1,CTRCLC ;INDICATE THAT WE ARE IN A TEST.
MOV #1,ERRTP ;SET ERROR TYPE AS FATAL IN ERROR TABLE.
MOV #3101,ERRNBR ;SET THE FIRST ERROR NUMBER IN ERROR TABLE.
MOV #EM3101,ERRMSG ;SET ERROR MESSAGE ADDRESS IN ERRBL.
MOV #ER9101,ERRBUC ;SELECT THE CORRECT ERROR REPORTING ROUTINE.

RESET THE OUT TO A KNOWN STATE, REMOVE THE STATUS CODES FROM THE FIFO.
CLEAR TX AND RX INTERRUPT ENABLE BITS IN THE CSR.
THIS SUBROUTINE REPORTS ERROR >> 3101 <<<.
JSR PC,CLRINT ;RESET THE DHU-11, REPORT ANY ERRORS FOUND.
BCC 60$ ;RESET FAILURE?, ABORT THIS TEST.

TEST ALL ACTIVE LINES INDIVIDUALLY.
WRITE THE REQUEST CODE TO THE DIAGNOSTIC FIELD IN THE LPR REGISTER.
VERIFY THAT A BMP CODE IS RETURNED WITHIN THE CORRECT TIME.

MOV ACTLNS,R5 ;GET THE ACTIVE LINE BIT MAP.
CLR R4 ;CLEAR THE LINE NUMBER COUNTER.
MOV CSRA,R3 ;GET THE ADDRESS OF THE OUT'S CSR.
2$:
CLC ;CLEAR THE CARRY BIT PRIOR TO SHIFTING BIT MAP.
ROR R5 ;SHIFT THE BIT MAP INTO THE CARRY BIT.
BCC 8$ ;DO NOT TEST THE LINE IF IT IS INACTIVE.

SELECT THE LINE UNDER TEST.
WRITE THE BMP REQUEST CODE TO THE DIAG FIELD IN THE LPR REGISTER.

MOV #3102,ERRNBR ;SET THE ERROR NUMBER TO 3102.
MOV R4,(R3) ;SELECT THE LINE CURRENTLY UNDER TEST.
BIS @2,ALPRA ;WRITE THE BMP REQUEST CODE TO THE LPR.

WAIT FOR BMP REQUEST CODE TO BE CLEARED, REPORT ERROR IF TIME-OUT OCCURS.

MOV #11750,R1 ;TEST BIT 1, TIMEOUT OF 1 SEC.
MOV LPRA,R2 ;PASS THE ADDRESS OF THE REGISTER TO TEST.
JSR PC,WAIBIC ;WAIT FOR REQUEST CODE TO CLEAR.
BCC 6$ ;GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.

WAIT FOR BMP CODE TO APPEAR IN THE FIFO, REPORT ERROR IF TIME OLY OCCURS.
7361 036456 005267 145306
7362 036462 012701 070012
7363 036466 016702 143550
7364 036472 004767 164572
7365 036476 103031
7366
7367
7368
7369
7370
7371
7372
7373 036500 005267 145264
7374 036504 017702 143534
7375 036510 100024
7376 036512 005267 145252
7377 036516 012700 170301
7378 036522 040200
7379 036524 001016
7380 036526 005267 145236
7381 036532 010200
7382 036534 000300
7383 036536 042700 177760
7384 036542 120400
7385 036544 001006
7386 036546 120227 000305
7387 036554 001364
7388 036554 001364
7389 036554 001040
7390
7391 036562 010401
7392 036564 012702 014643
7393
7394 036570 001403
7395
7396
7397
7398
7399 036572 032767 000100 143424
7400 036600 001403
7401
7402
7403
7404
7405 036602 005204
7406 036604 005705
7407 036606 001302
7408 036610 005067 143472
7409 036614
7410 036614 104401

; SET ERROR NUMBER TO 3103.
; TEST 017010 070012.
; PASS THE ADDRESS OF THE REGISTER TO TEST.
; WAIT FOR RX DATA AVAILABLE TO SET.
; GO REPORT ERROR IF CODE DID NOT CLEAR IN TIME.
; READ THE BMP CODE (IF IT IS THERE) FROM THE RBUF REGISTER.
; DETERMINE IF IT IS A VALID BMP CODE.
; VERIFY THE BMP CODE WAS RECEIVED FROM THE CORRECT CHANNEL.
; IF THE BMP CODE DOES NOT INDICATE OUT RUNNING OK, THEN SAVE IT ON
; THE QUEUE TO BE REPORTED IN A LATER TEST.

; SET ERROR NUMBER TO 3104.
; SET THE BMP CODE FROM THE FIFO.
; GO REPORT ERROR IF NO BMP CODE FOUND.
; SET-UP A BMP CODE MASK.
; TRY TO CLEAR THE BMP MASK.
; GO REPORT ERROR IF IT IS NOT A VALID BMP CODE.
; SET THE ERROR NUMBER TO 3106.
; COPY BMP CODE.
; PUT THE LINE NUMBER IN THE LOW BYTE.
; CLEAR THE UNWANTED BITS.
; DID THE BMP CODE COME FROM THE CORRECT LINE?.
; NO; GO REPORT ERROR.
; IS THE BMP CODE A "GOOD ONE"?.
; YES; SKIP SAVING THE BMP CODE ON THE QUEUE.
; SAVE THE BMP CODE ON THE QUEUE.
; GO SEE IF THERE ARE ANY MORE LINE TO TEST.

; MOV R4,R1
; MOV #EM3102,R2
; "BMP REQUEST BIT BAD ON LINE:"
; ERROR

; EXIT THE TEST IF EXTENDED ERROR REPORTING HAS NOT BEEN ENABLED

; BIT #IT06,OPTION
; EXIT WITH TEST FAILURE MESSAGE IF
; NO EXTENDED ERROR REPORTING HAS BEEN REQUESTED

; VERIFY ALL ACTIVE LINES HAVE BEEN TESTED.

; INCREMENT THE LINE NUMBER COUNTER.
; ARE THERE ANY MORE ACTIVE LINES TO TEST?.
; YES; BRANCH TO TEST THE NEXT LINE.
; INDICATE THAT WE ARE NOT WITHIN A TEST.
.SBTL HARDWARE TEST
.REPBMP

;** REPBM CODES IN THE QUEUE
;** THIS IS A PSEUDO-TEST USED TO REPORT ANY BMP CODES THAT WERE FOUND
;** IN THE DUT'S FIFO DURING PREVIOUS TEST, AND LOGGED IN THE BMP CODE
;** QUEUE.
;** IT IS UNLIKELY THAT RUNNING THIS PSEUDO TEST ALONE WILL PRODUCE ANY
;** ERROR REPORTS.
;**
;**
** BGNST

TNUM = TNUM + 1 ;INCREMENT THE ASSEMBLY TIME TEST COUNTER.
MOV @TNUM,TSTNUM ;SET UP THE TEST NUMBER. (93)
MOV 0-1,CTRLCF ;INDICATE THAT WE ARE IN A TEST.
MOV BMPQ,R2 ;GET THE CONTENTS OF THE POINTER.
MOV #BMPQ,R3 ;GET THE START ADDRESS OF THE QUEUE.
CMP R2,R3 ;SEE IF THE POINTER HAS MOVED FROM THE BASE.
BEQ 60$ ;EXIT NO CODES IN THE QUEUE.

; THERE IS AT LEAST ONE BMP CODE IN THE QUEUE. REPORT THE ERROR.

; REPORT ERROR BMP CODE FOUND IN TEST N8, BMP CODE:NNNNNN

MOV #EM9304,R1 ;PASS THE FIRST MESSAGE TO BE REPORTED.
ERROR 9301,EM9301,ER9301 ; >>>>>> ERROR 09301 <<<.

MOV #BMPQ,R3 ;SET POINTER BACK TO THE BEGINNING OF THE QUEUE.

CLR CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.

ENDST

L1005$; TRAP CIEST

434
7435 036646 012701 015433
7436 036652 104455
036654 022125
036656 015257
036660 017262
7437
7438 036662 012767 002526 143684
7439
7440 036670 005067 143412
60$; CLR CTRLCF ;INDICATE THAT WE ARE NOT WITHIN A TEST.
7441 036674
036674
036674 104401
.SBTIL HARDWARE PARAMETER CODING SECTION

;**
; THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
; WITH THE OPERATOR.
;
.036676 000022
.036700

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

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

.ACTIVE LINES BIT MAP QUESTION:
GPRMA HWPTQ3.4,0,MAPLNS,0,MAPLNS,YES

.INTERRUPT BR LEVEL QUESTION:
GPRMA HWPTQ4.6,0,377,0,6,YES

ENDHRD

036744

036744 103 123 122
036747 040 101 104
036752 104 122 105
036755 123 123 072
036760 040 000
**HWPTQ2: .ASCIZ /INTERRUPT VECTOR ADDRESS: /**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>036762</td>
<td>111</td>
</tr>
<tr>
<td>036765</td>
<td>105</td>
</tr>
<tr>
<td>036770</td>
<td>125</td>
</tr>
<tr>
<td>036773</td>
<td>040</td>
</tr>
<tr>
<td>036776</td>
<td>103</td>
</tr>
<tr>
<td>037001</td>
<td>122</td>
</tr>
<tr>
<td>037004</td>
<td>104</td>
</tr>
<tr>
<td>037007</td>
<td>105</td>
</tr>
<tr>
<td>037012</td>
<td>072</td>
</tr>
</tbody>
</table>

**HWPTQ3: .ASCIZ /ACTIVE LINE BIT MAP: /**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>037015</td>
<td>101</td>
</tr>
<tr>
<td>037020</td>
<td>111</td>
</tr>
<tr>
<td>037023</td>
<td>040</td>
</tr>
<tr>
<td>037026</td>
<td>116</td>
</tr>
<tr>
<td>037031</td>
<td>102</td>
</tr>
<tr>
<td>037034</td>
<td>040</td>
</tr>
<tr>
<td>037037</td>
<td>120</td>
</tr>
<tr>
<td>037042</td>
<td>000</td>
</tr>
</tbody>
</table>

**HWPTQ4: .ASCIZ /INTERRUPT BR LEVEL: /**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>037043</td>
<td>111</td>
</tr>
<tr>
<td>037046</td>
<td>105</td>
</tr>
<tr>
<td>037051</td>
<td>125</td>
</tr>
<tr>
<td>037054</td>
<td>040</td>
</tr>
<tr>
<td>037057</td>
<td>040</td>
</tr>
<tr>
<td>037062</td>
<td>126</td>
</tr>
<tr>
<td>037065</td>
<td>072</td>
</tr>
</tbody>
</table>

**.EVEN**
SOFT: SOFTWARE PARAMETER CODING SECTION

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

BEGIN

BGNSFT

L1 SOFT:

UNIT NUMBER PRINTOUT QUESTION:
GPRML SWPTQ1.0,20,YES .WORD T$CODE
.GRPML SWPTQ1 .WORD SWPTQ1
.GRPML SWPTQ2 .WORD 20

ROM VERSION NUMBER PRINTOUT ON FIRST PASS QUESTION:
GPRML SWPTQ2.0,1,YES

EXTENDED ERROR REPORTING QUESTION:
GPRML SWPTQ3.0,100,YES

IF EXTENDED ERROR REPORTING IS NOT REQUIRED THEN SKIP THE NEXT QUESTION.
XFERF ENDD

NUMBER OF INDIVIDUAL DATA ERRORS TO REPORT ON A LINE QUESTION:
GPRML SWPTQ4.2,0,177777,0,177777,YES

ENDD: ENDSFT

NUMBER: 10061

L10061:

UNIT NUMBER AS EACH UNIT IS TESTED: /
SOFTWARE PARAMETER CODING SECTION

037147  102  105  122
037152  040  101  123
037155  040  105  101
037160  103  110  040
037163  125  116  111
037166  124  040  111
037171  128  040  124
037174  105  123  124
037177  105  104  072
037202  040  000

7557 037204  122  117  115
037207  040  126  105
037212  122  123  111
037215  117  116  040
037220  120  122  111
037223  116  124  117
037226  125  124  040
037231  117  116  040
037234  124  110  105
037237  040  106  111
037242  122  123  124
037245  040  120  101
037250  125  123  072

7558 037253  105  130  124
037260  105  116  104
037263  105  104  040
037266  105  122  122
037271  117  122  040
037274  122  105  120
037277  117  122  124
037302  111  116  107
037305  072  040  000

7559 037310  116  125  115
037313  102  105  122
037316  040  117  106
037321  040  111  116
037324  104  111  126
037327  111  104  125
037332  101  114  040
037335  104  101  124
037340  104  040  105
037343  122  122  117
037346  122  123  040
037351  124  117  040
037354  122  105  120
037357  117  122  124
037362  040  117  116
037365  040  101  040
037370  114  111  116
037373  105  072  040

7560 .EVEN
.PATCH::
.BLKW 24
LASTAD
.037450 000000
.037452 000000
.037454
LLAST::
.ENDMOD
.END
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>VALUE</th>
<th>ADDRESS</th>
<th>VALUE</th>
<th>ADDRESS</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1STS</td>
<td>000001</td>
<td>T1SOF</td>
<td>010061</td>
<td>T1</td>
<td>032244G</td>
</tr>
<tr>
<td>T1IAU</td>
<td>010025</td>
<td>T1SIN</td>
<td>010001</td>
<td>T1B</td>
<td>032834G</td>
</tr>
<tr>
<td>T1IAUI</td>
<td>010022</td>
<td>T1ITE</td>
<td>010057</td>
<td>T1C</td>
<td>032752G</td>
</tr>
<tr>
<td>T1IDR</td>
<td>010023</td>
<td>T1D</td>
<td>025214G</td>
<td>T2</td>
<td>025476G</td>
</tr>
<tr>
<td>T1H0U</td>
<td>010024</td>
<td>T1H0</td>
<td>027610G</td>
<td>T20</td>
<td>033214G</td>
</tr>
<tr>
<td>T1H1R</td>
<td>010060</td>
<td>T1N0024</td>
<td>030024G</td>
<td>T21</td>
<td>033330G</td>
</tr>
<tr>
<td>T1H2W</td>
<td>010000</td>
<td>T1C2</td>
<td>030252G</td>
<td>T22</td>
<td>033646G</td>
</tr>
<tr>
<td>T1IN1</td>
<td>010021</td>
<td>T1I3</td>
<td>030546G</td>
<td>T23</td>
<td>034222G</td>
</tr>
<tr>
<td>T1MSG</td>
<td>010016</td>
<td>T14</td>
<td>031034G</td>
<td>T24</td>
<td>035360G</td>
</tr>
<tr>
<td>T1PR0</td>
<td>010020</td>
<td>T15</td>
<td>031454G</td>
<td>T25</td>
<td>036322G</td>
</tr>
<tr>
<td>T1PR1T</td>
<td>010017</td>
<td>T16</td>
<td>031776G</td>
<td>T26</td>
<td>036616G</td>
</tr>
</tbody>
</table>

ABS. 037454  000
000000  091

VIRTUAL MEMORY USED: 28661 WORDS  (112 PAGES)
DYNAMIC MEMORY: 20060 WORDS  (77 PAGES)
ELAPSED TIME: 00:05:14

CZDHUB0.BIN, CZDHUB0.LST, -SP=SVC34R/ML,CZDHUB0.P11