

IBV11, IB11

IBV11-A DIAGNOSTIC  
CVIBABO

AH-T619B-MC  
FICHE 1 OF 1

OCT 1983  
COPYRIGHT © 78-83  
MADE IN USA



A grid of diagnostic data tables. Each cell contains a small table with columns and rows of text, likely representing test results or system parameters. The text is too small to read clearly but appears to be organized in a structured format.



IDENTIFICATION

Product Code: AC-A880B-MC  
Product Name: CVIBAB0 IBV11-A DIAG  
Date : FEB 1983  
Maintainer: Ray Shoop

Copyright (C) 1978,1983  
Digital Equipment Corporation, Maynard, Mass.

This software is furnished under a license for use only on a single computer system and may be copied only with the inclusion of the above copyright notice. This software, or any other copies thereof, may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title to and ownership of the software shall at all times remain in DEC.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment which is not supplied by DEC.

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	Equipment
2.2	Storage
3.0	LOADING PROCEDURE
3.1	Method
3.2	Non-Standard Address, Vector, or Use of Software Switch Register
4.0	STARTING PROCEDURE
4.1	Control Switch Settings
4.2	Starting Address
4.3	Program and/or Operator Action
5.0	OPERATING PROCEDURE
5.1	Switch Register Function
5.2	CPU Selection Function
5.3	Scope Loops
5.4	Program and/or Operator Action
5.4.1	Logic Test
6.0	ERRORS
6.1	Error Printout
6.2	Non-Standard Error Halts
7.0	RESTRICTIONS
7.1	Starting Restriction
7.2	Possible Program "Bombs"
7.3	Customer Devices
8.0	MISCELLANEOUS
8.1	Power Fail
8.2	XXDP, ACT, APT
8.3	Execution Time
8.4	LSI-11 "ODT" Commands
8.5	Entering LSI-11 "ODT"
8.6	Use of Program Software SWR
8.7	Trap Catcher

The 'B' revision allow the operator to indicate CPU type.  
The type of CPU is needed because signal timing is performed.  
The 'B' revision includes provisions for the 'ER1-INHIBIT'  
(S1-8) switch to be tested.

## 1.0 ABSTRACT

This program allows the user to check-out a greater portion of logic on this option if a second IBV11-A is available. See section 2.1. When a second IBV11-A can be obtained, the user must inform this diagnostic to exercise the logic on one IBV11-A that requires a KGM (Known Good Module). Please note that the second IBV11-A should be known good. When a KGM is available, the 'ER1-INHIBIT SWITCH' on the module under test should be CLEARED (OPEN-OFF). No attempt is made to checkout the KGM and no conclusion that if good passes are made through this diagnostic that the KGM is also good. Signals 'SRQ', 'ER1', 'BIAKI' and 'ER1IHB' are not tested on the IBV11-A if a KGM is not used.

If the user is unfamiliar with an LSI-11 he should review sections 8.4 and 8.5. A software switch register is included with this program.

Every effort was made to make this program conform to LSI-11 programming restrictions. However, the user should read sections 7.1 and 7.2.

## 2.0 REQUIREMENTS

### 2.1 Equipment

1. PDP-11 Family Computer with 4K of memory (or more) and console I/O facilities (i.e., TTY).
2. IBV11-A under test.
3. (Optional) Second IBV11-A 'KGM' (known good module). The 'KGM' must have an instrumentation Bus Cable between it and the first IBV-11. Its base address should be 760160 and vector address of 660 (see section 3.2 if different).

#### NOTE

While it is generally recommended that a 'KGM' is used, if one is available, deposit a '000001' into location '\$CDW1'. No test will be performed that requires the 'KGM' if \$CDW1 is zero.

### 2.2 Storage

This program occupies and uses the lower 4K of memory.

### 3.0 LOADING PROCEDURE

#### 3.1 Method

Standard procedure for normal binary tapes should be followed.

The program can also be loaded by XXDP, ACT or APT.

#### 3.2 Non-Standard Address, Vector, or Use of Software Switch Register

This program is set to test a IBV11-A with a standard address and vector. If any of these are different on the IBV11-A you are testing, change the corresponding location in memory before starting this test.

<u>TAG</u>	<u>ADDRESS</u>	<u>CURRENT CONTENTS</u>	<u>COMMENTS</u>
\$BASE:	1244	760150	::BASE ADDRESS OF EQUIPMENT :: UNDER TEST
\$VECT1:	1240	000420	::INTERRUPT VECTOR #1
\$WREG:	176	000000	::MANUAL SWR.
\$IBS2:	1402	760160	:: ADDRESS OF SECOND IBV11-A.
\$VECTA2:	1406	000660	:: VECTOR ADDRESS OF SECOND IBV11-A.
\$CDW1:	1250	000000	::DEVICE DESCRIPTOR WORD #1 (if = 000001 to use 'KGM' in testing 1st IBV-11) (Default = 00000 to disable use of KGM in tests.)
\$CPUOP:	1216	000010	::TYPE OF CPU CODE (DEFAULT = 00010 INDICATE PDP-11/03) (REF. SECTION 5.2 FOR LIST OF CODES)

#### 4.0 STARTING PROCEDURE

##### 4.1 Control Switch Setting

Before starting the diagnostic, set all switch register bits as desired. See section 5.1.

##### 4.2 Starting Addresses

200 Start of Logic Tests

##### 4.3 Program and/or Operator Action

1. Load program into memory.
2. Enter keyboard 'DDT'.
3. Alter location 'SWREG' to reflect desired options of a switch register - See section 5.1.
4. Alter location '\$CPUOP' to reflect current cpu type code - See section 5.2.
5. Type starting address, followed by 'G' to start program.

## 5.0 OPERATING PROCEDURE

### 5.1 Switch Register Function

<u>SWR BIT</u>	<u>OCTAL</u>	<u>FUNCTION WHEN SET</u>
15	100000	HALT ON ERROR
14	040000	LOOP ON TEST
13	020000	INHIBIT ERROR TYPEOUT
11	004000	INHIBIT ITERATIONS (SHORT PASS)
10	002000	INDICATE 'ER1-INHIBIT' SWITCH (S1-8) IS SET (CLOSED-ON)
09	001000	LOOP ON ERROR
08	000400	LOOP ON TEST IN SWR <7:0>

#### NOTE

The Switch Register may be changed at any time while the diagnostic is running by typing 'G'.

### 5.2 Cpu Type Code Function

01	PDP-11/04
02	PDP-11/05
03	PDP-11/20
04	PDP-11/40
05	PDP-11/45
06	PDP-11/70
07	PDP-11/34
10	PDP-11/03 <DEFAULT>
11	PDP-11/
12	PDP-11/
13	PDP-11/
14	PDP-11/
15	PDP-11/
16	PDP-11/
17	PDP-11/

### 5.3 Scope Loops

If an error occurs and the user wishes to scope the error, "SWREG" should be altered to "100000" at the start of the test to halt on error, then when the program halts on error and the CPU enters "ODT", "\$SWREG" should be altered to "060000" to loop on current test and inhibit error typeout, then type 'P' to continue program execution.

#### 5.4 Program and/or Operator Action

1. When the program is initially started it will type:

CVIBA-B

SWR=000000 NEW=

2. Program now waits for the operator to enter a switch register setting (see section 8.6). To change the switch register setting, see section 8.6.
3. Program executes first pass of logic tests, subtest iterations inhibited.
4. Program reports any errors it detects.
5. Program reports 'END PASS 1'.
6. Program executes second pass of logic tests, only this time it will loop on each test 2000 times.
7. Program the reports 'END PASS 2'.
8. Program will continue executing steps 6 and 7 until stoped.

#### 6.0 ERRORS

##### 6.1 Error Printout

Printout varies with the error detected. The error PC typed out is the actual location of the error call.

##### 6.2 Non-Standard Error Halt

Bus errors will cause a Halt in the routine "IOTRD". The address that caused this trap will be in "TRTO".



## 7.0 RESTRICTIONS

### 7.1 Starting Restriction

If a free-running clock, such as 60Hz from the power supply, is attached to the 'BEVNT' bus line on both REV level C/D and E systems, an interrupt to location 100 will occur when using the 'G' and 'L' commands prior to executing the first instruction. Therefore this program can not disable the BEVNT bus line by inhibiting interrupts.

User systems requiring a free-running clock attached to the BEVNT bus line can temporarily avoid this situation by setting the PSW(RS) to 200, loading the PC with the starting address instead of using the 'G' command, and then using the 'P' command. Before using the 'L' command, the PSW(RS) can be set to 200, thereby inhibiting interrupts, to avoid receiving the event interrupt after loading the ABS loader.

### 7.2 Possible Program 'BOMBS'

The first two tests of this program check to see if the IBV11-A responds to the address the program thinks its at. If the IBV11-A does not respond, a bus error occurs.

For more information on the next subject, see JAN. 1976 LSI-11 engineering bulletin issued by the Digital Components Group.

Bus errors may alter the preset contents of location 4 before the trap is executed, thereby transferring program control to area in the program that was not set up to handle the trap. If this happens, the program will 'BOMB' and possibly rewrite parts of itself.

### 7.3 Customer Devices

No customer devices should be connected to the unit under test while executing the program.

8.0 MISCELLANEOUS

8.1 Power Fail

After a power failure occurs, the program execution will continue at the point where the power occurred. The program will type "POWER".

8.2 XXDP, ACT, APT

The program is chainable under XXDP, ACT, or APT. Although "APT HOOKS" have been installed, they have not been tested.

8.3 Execution Time

PDP-11/03 CPU WITH ONLY 1 BOARD

0.1 Minutes (6 sec) Iteration Inhibited - No Errors  
0.5 Minutes (30 sec) With Iterations - No Errors

PDP-11/03 CPU WITH KGM AND 1 BOARD

1.4 Minutes (80 sec) With Iterations - No Errors

#### 8.4 LSI-11 'ODT' Commands

<u>FORMAT</u>	<u>DESCRIPTION</u>
<CR> RETURN	Close opened location and accept next command.
<LF> LINE FEED	Close current location; open next sequential location.
^(UPARROW)	Open previous location.
< (LEFT ARROW)	Take contents of opened location, indexed by contents of PC, and open that location.
@	Take contents of opened location as absolute address and open that location.
R/	Open the word at location R.
/	Reopen the last location.
\$N/ or RN/	Open general register N(0-7) or S(PS register).
R:G or RG	GOTO location R and start program.
NL	Execute Bootstrap loader using N as device CSR. Console device is 177560.
:P or P	Proceed with program execution.
RUBOUT	Erases previous numeric character. Response is a backslash ().

#### 8.5 Entering LSI-11 'ODT'

The halt or ODT microcode state of the KD11F (LSI-11 module) can be entered in five different ways (others are a subset of these) from the run state:

1. Execution of a LSI-11 halt instruction.
2. A double bus error.
3. As a power up option.
4. ASCII break with DLV11 framing error asserting the B halt line (enabled by jumper of DLV11).

Upon entering the halt state, the KD11F responds through the set of command listed in section 8.4.

## 8.6 Use of Program Software SWR

The software switch register may be changed by typing ^G (control and letter G keys typed simultaneously). When ^G is typed, the program responds by typing "SWR=XXXXXX" where XXXXXX equals the former contents of the switch register.

If you wish to keep the current value, type <CR>. If you wish to change the value, type the new value followed by a <CR>.

It is important to note that the diagnostic is not running after the ^G until a <CR> is typed.

## 8.7 Trap Catcher

The Trap Catcher in this diagnostic employs a new concept. This concept will enable the user of this diagnostic to gain more knowledge of the events that lead the program to this area.

The Trap Catch consists of PC+2 and JSR PC,R0. (i.e., Location 300 would contain 302 and location 302 would contain 4700).

When a device interrupts to the Trap Catcher, it would pick up the PC+2 of the trap as an address of the interrupt service routine.

The program would then pick up "4700" as the new PSW. Bit 7 of the new PSW having been set, would cause further interrupts from happening. When the CPU attempts to execute "4700" (JSR PC,R0), a Bus-time-out trap will occur to location 4. Location 4 contains a pointer to "IOTRD", a routine that will report the trap as an error.

To guard against "Real" Bus errors routing us through location 4 to "IOTRD", we check to see if the trap that brought us to location 4 really came from the Trap Catcher area. If not we'll halt and leave the Trap Address in "TRTO".

More about the interrupt error can be found in the description of the error in the program listing in the routine "IOTRD".

14	OPERATIONAL SWITCH SETTINGS
16	TRAP CATCHER
46	BASIC DEFINITIONS
54	ACT11 HOOKS
58	APT PARAMETER BLOCK
60	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
162	REG ADDRESS AND COMMON TAGS
208	PROGRAM START
212	INITIALIZE THE COMMON TAGS
271	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
277	T1 *TEST THE ADDRESSABILITY OF THE IBS, IBD REGISTERS
315	T2 *TEST THAT BASE ADDRESSES +4,+6 RESPOND WHEN ADDRESSED
345	T3 *TEST THAT IBS IS CLEAR AT INIT OF TESTING
356	T4 *TEST THAT IBD IS CLEAR AT INIT OF TESTING
375	T5 *TEST THAT BASE ADDRESSES +4,+6 RETURN ZERO WHEN READ
394	T6 *TEST THAT WE CAN SET TCS (BIT00), TCS SETS CMD (BIT10)
415	T7 *TEST THAT EOP (BIT01) WILL SET
436	T10 *TEST THAT REM (BIT02) WILL SET + CLEAR
458	T11 *TEST THAT IBC (BIT03) WILL SET AND CLEAR
481	T12 *TEST THAT TON (BIT05) AND TKR (BIT09) SET AND CLEAR
504	T13 *TEST THAT LON (BIT04) WILL SET AND CLEAR
523	T14 *TEST THAT IE (BIT06) SET AND CLEAR
548	T15 *TEST THAT ACC (BIT07) CAN BE SET AND CLEARED
570	T16 *TEST THAT SRQ (BIT15) CAN BE SET AND CLEARED (IF SWR10=1)
620	T17 *TEST THAT IBD BIT 0 CAN BE SET + CLEARED
622	T20 *TEST THAT IBD BIT 1 CAN BE SET + CLEARED
624	T21 *TEST THAT IBD BIT 2 CAN BE SET + CLEARED
626	T22 *TEST THAT IBD BIT 3 CAN BE SET + CLEARED
628	T23 *TEST THAT IBD BIT 4 CAN BE SET + CLEARED
630	T24 *TEST THAT IBD BIT 5 CAN BE SET + CLEARED
632	T25 *TEST THAT IBD BIT 6 CAN BE SET + CLEARED
634	T26 *TEST THAT IBD BIT 7 CAN BE SET + CLEARED
637	T27 *TEST THAT NO DATA GETS XFERRERD, IF NOT ENABLED
652	T30 *TEST IBD-DAC (BIT08) AND IBD-DAV (BIT09) CAN GE SET + CLEARED
745	T31 *TEST THAT IBD - REN SETS WHEN IBS - REM SETS, ALSO TEST CLEAR
748	T32 *TEST THAT IBD - IFC SETS WHEN IBS - IBC SETS, ALSO TEST CLEAR
751	T33 *TEST THAT IBD - ATN SETS WHEN IBS - TCS SETS, ALSO TEST CLEAR
754	T34 *TEST THAT IBD - EOI SETS WHEN IBS - EOP SETS, ALSO TEST CLEAR
758	T35 *TEST THAT IBD-SRQ SETS WHEN IBS-SRQ SETS, ALSO TEST CLEAR (IF SWR10=1)
779	T36 *TEST THAT RFD SET WHEN CSR CLEAR, CLEAR WHEN ACC SET
797	T37 *TEST THAT WE CAN GENERATE AN ER2
811	T40 *TEST THAT BUS INIT CLEARS ACC,TON,LON,REM,EIP,TCS
822	T41 *TEST IBC CLEARS ACC,TON,LON,REM AND EOP
834	T42 *TEST THAT BUS INIT INDIRECTLY CLEARS IBD
846	
847	INTERRUPT TESTS
848	
850	T43 *TEST THAT CMD CAN GENERATE AN INTERRUPT
875	T44 *TEST THAT TKR AND LNR CAN GENERATE INTERRUPTS
925	T45 *TEST THAT ER2 CAN GENERATE AN INTERRUPT
953	T46 *TEST THAT SRQ CAN GENERATE AN INTERRPUT (SWR10=1)
980	
981	SECOND MODULE TESTS

982	
984	
1021	T47 *TEST THAT MODULE PASSES 'BIAKI''
1050	T50 *TEST THAT SRQ CAN GENERATE AN INTERRUPT
1068	T51 *TEST THAT ERROR 1 IS GENERATED IF ATN IS ON THE IB BUS
1083	T52 *TEST THAT ERROR 1 IS GENERATED IF IFC IS ON IB BUS BY SECONUD MODULE
1098	T53 *TEST THAT ERROR 1 IS GENERATED IF REN IS ON IB BUS
1135	T54 *TEST THAT AN ERROR 1 CAN GENERATE AN INTERRUPT
1162	T55 *TEST THAT DATA CAN BE XFERRED BETWEEN THE MODULE UNDER TEST AND THE KGM
1164	T56 *TEMP END OF TESTS
1165	SYSMAC ROUTINES:
1168	END OF PASS ROUTINE
1169	ERROR HANDLER ROUTINE
1170	ERROR MESSAGE TYPEOUT ROUTINE
1171	SCOPE HANDLER ROUTINE
1173	TTY INPUT ROUTINE
1175	BINARY TO OCTAL (ASCII) AND TYPE
1176	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1177	TYPE ROUTINE
1179	APT COMMUNICATIONS ROUTINE
1255	POWER DOWN AND UP ROUTINES
(3)	TRAP DECODER
1257	TRAP TABLE
	MESSAGES AND TABLES

1  
2  
3  
4  
10  
11  
12  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
13  
14  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
15  
16  
17  
18  
19  
20  
21  
22  
23  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
(1)  
(1)  
(1)  
(1)

165400

000001

000000

000004 013126 000200

000140 170000 000300

000174 000000

000176 000000

000100 000104 000200 000002

000200 000137 001460

001100

```
.NLIST MC,MD,CND
.LIST ME
.ENABL ABS
.ENABL AMA
$SWR=165400
```

```
.TITLE CVIBA-B
.*COPYRIGHT (C) 1983
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY EDWARD C. BADGER MOD BY R. SHOOP
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
.*
$TN=1
```

```
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.* SWITCH USE
.* -----
.* 15 HALT ON ERROR
.* 14 LOOP ON TEST
.* 13 INHIBIT ERROR TYPEOUTS
.* 11 INHIBIT ITERATIONS
.* 10 INDICATE 'EH1-INHIBIT' SWITCH IS SET (CLOSED-ON)
.* 9 LOOP ON ERROR
.* 8 LOOP ON TEST IN SWR<7:0>
```

```
.SBTTL TRAP CATCHER
```

```
=0
.*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A "+2"
.*AND "JSR PC,R0" SEQUENCE TO CATCH ILLEGAL INTERRUPTS,
.*AND INTERRUPTS TO THE WRONG VECTOR.
.*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
.*VECTORS
```

```
=4
.WORD IOTRD,200 ;HANDLE BUSS ERROR.
```

```
=140
170000,300 ;FALCON CPU FIX
```

```
=174
DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER.
SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER.
```

```
=100
.WORD 104,200,2 ;IF 'B EVENT' ON Q-BUS IS
;CONNECTED, WE NEED A WAY OF
;IGNORING ITS INTERRUPTS.
```

```
=200
JMP START
```

```
.SBTTL BASIC DEFINITIONS
```

```
.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
```





(1) .EQUIV SW06,SW6  
(1) .EQUIV SW05,SW5  
(1) .EQUIV SW04,SW4  
(1) .EQUIV SW03,SW3  
(1) .EQUIV SW02,SW2  
(1) .EQUIV SW01,SW1  
(1) .EQUIV SW00,SW0

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

(1) .EQUIV BIT09,BIT9  
(1) .EQUIV BIT08,BIT8  
(1) .EQUIV BIT07,BIT7  
(1) .EQUIV BIT06,BIT6  
(1) .EQUIV BIT05,BIT5  
(1) .EQUIV BIT04,BIT4  
(1) .EQUIV BIT03,BIT3  
(1) .EQUIV BIT02,BIT2  
(1) .EQUIV BIT01,BIT1  
(1) .EQUIV BIT00,BIT0

(1) ;\*BASIC "CPU" TRAP VECTOR ADDRESSES  
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS  
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS  
(1) 000014 TBITVEC=14 ;: "T" BIT  
(1) 000014 TRTVEC= 14 ;:TRACE TRAP  
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)  
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
(1) 000024 PWRVEC= 24 ;:POWER FAIL  
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) \*\*ERROR\*\*  
(1) 000034 TRAPVEC=34 ;: "TRAP" TRAP  
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR  
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR  
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR

47  
48 160150 ABASE= 160150  
49 000420 AVECT1= 420  
50 000200 APRIOR= 200  
51 000010 ACPUOP= 10  
52 000001 \$TN=1  
53

```
54      .SBTTL ACT11 HOOKS
(1)
(2)      ::*****
(1)      :HOOKS REQUIRED BY ACT11
(1)      $SVPC=.          ;SAVE PC
(1)      .=46
(1) 000046 000046      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1)      010044
(1)      .=52
(1) 000052 000052      .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)      000204          ;; RESTORE PC
(1)      000204
55
56      001000          .=1000
57
58      .SBTTL APT PARAMETER BLOCK
(1)
(2)      ::*****
(1)      :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      :*****
(1)      .SX=.          ;;SAVE CURRENT LOCATION
(1)      .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200      200          ;;FOR APT START UP
(1)      000044          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000      $APTHDR      ;;POINT TO APT HEADER BLOCK
(1)      001000          .=.SX          ;;RESET LOCATION COUNTER
(2)      :*****
(1)      :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)      :INTERFACE SPEC.
(1)
(1) 001000      $APTHD:
(1) 001000 000000      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001170      $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000074      $STMT: .WORD 60.          ;;RUN TIM OF LONGEST TEST
(1) 001006 000170      $PASTM: .WORD 120.        ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000170      $UNITM: .WORD 120.        ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000031      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
59
```



```

(2) 001210          SETABLE:          :::APT ENVIRONMENT TABLE
(2) 001210          $ENV: .BYTE      AENV          :::ENVIRONMENT BYTE
(2) 001211          $ENVM: .BYTE     AENVM         :::ENVIRONMENT MODE BITS
(2) 001212          $$WREG: .WORD    ASWREG        :::APT SWITCH REGISTER
(2) 001214          $USWR: .WORD     AUSWR         :::USER SWITCHES
(2) 001216          $CPUOP: .WORD    ACPUOP        :::CPU TYPE,OPTIONS
(2)                :*                BITS 15-11=CPU TYPE
(2)                :*                11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                :*                11/70=06,PDQ=07,Q=10
(2)                :*                BIT 10=REAL TIME CLOCK
(2)                :*                BIT 9=FLOATING POINT PROCESSOR
(2)                :*                BIT 8=MEMORY MANAGEMENT
(2) 001220          $MAMS1: .BYTE     AMAMS1        :::HIGH ADDRESS,M.S. BYTE
(2) 001221          $MTYP1: .BYTE     AMTYP1        :::MEM. TYPE,BLK#1
(2)                :*                MEM.TYPE BYTE -- (HIGH BYTE)
(2)                :*                900 NSEC CORE=001
(2)                :*                300 NSEC BIPOLAR=002
(2)                :*                500 NSEC MOS=003
(2) 001222          $MADR1: .WORD     AMADR1        :::HIGH ADDRESS,BLK#1
(2)                :*                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2) 001224          $MAMS2: .BYTE     AMAMS2        :::HIGH ADDRESS,M.S. BYTE
(2) 001225          $MTYP2: .BYTE     AMTYP2        :::MEM. TYPE,BLK#2
(2) 001226          $MADR2: .WORD     AMADR2        :::MEM.LAST ADDRESS,BLK#2
(2) 001230          $MAMS3: .BYTE     AMAMS3        :::HIGH ADDRESS,M.S.BYTE
(2) 001231          $MTYP3: .BYTE     AMTYP3        :::MEM. TYPE,BLK#3
(2) 001232          $MADR3: .WORD     AMADR3        :::MEM.LAST ADDRESS,BLK#3
(2) 001234          $MAMS4: .BYTE     AMAMS4        :::HIGH ADDRESS,M.S.BYTE
(2) 001235          $MTYP4: .BYTE     AMTYP4        :::MEM. TYPE,BLK#4
(2) 001236          $MADR4: .WORD     AMADR4        :::MEM.LAST ADDRESS,BLK#4
(2) 001240          $VECT1: .WORD     AVECT1        :::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001242          $VECT2: .WORD     AVECT2        :::INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001244          $BASE: .WORD     ABASE         :::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001246          $DEVN: .WORD     ADEVN         :::DEVICE MAP
(2) 001250          $CDW1: .WORD     ACDW1         :::CONTROLLER DESCRIPTION WORD#1
(2) 001252          $ETEND:          :::
(2)                .MEXIT

```

ERROR POINTER TABLE

```

(1) .SBTTL ERROR POINTER TABLE
(1)
(1) : *THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) : *THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) : *LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) : *NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) : *NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) : * EM ::POINTS TO THE ERROR MESSAGE
(1) : * DH ::POINTS TO THE DATA HEADER
(1) : * DT ::POINTS TO THE DATA
(1) : * DF ::POINTS TO THE DATA FORMAT
(1)
(1) SERRTB:
(1) 001252
64 :ITEM 1
65 001252 013256 EM1 :IBS FUNCTION ERROR
66 001254 014401 DH1 :TEST ERRPC IB ADDR
67 001256 014602 DT1 :$TESTN,$ERRPC,IBS
68 001260 014652 DFO :ALL NUMBERS ARE IN OCTAL FORM.
69 :ITEM 2
70 001262 013304 EM2 :IBD FUNCTION ERROR
71 001264 014401 DH1 :TEST ERRPC IB ADDR
72 001266 014602 DT1 :$TESTN,$ERRPC,IBS
73 001270 014652 DFO :ALL NUMBERS ARE IN OCTAL FORM.
74 :ITEM 3
75 001272 013332 EM3 :IBS DATA ERROR
76 001274 014447 DH3 :TEST ERRPC GOOD BAD
77 001276 014616 DT3 :$TESTN,$ERRPC,$GDDAT,$BDDAT
78 001300 014652 DFO :ALL NUMBERS ARE IN OCTAL FORM.
79 :ITEM 4
80 001302 013354 EM4 :IBD DATA ERROR
81 001304 014447 DH3 :TEST ERRPC GOOD BAD
82 001306 014616 DT3 :$TESTN,$ERRPC,$GDDAT,$BDDAT
83 001310 014652 DFO :ALL NUMBERS ARE IN OCTAL FORM.
84 :ITEM 5
85 001312 013376 EM5 :IBS/IBD ADDRESS ERROR
86 001314 014504 DH5 :TEST ERROR PC ADDRESS
87 001316 014630 DT5 :$TESTN,$ERRPC,IBS
88 001320 014652 DFO :ALL NUMBERS ARE IN OCTAL FORM.
89 :ITEM 6
90 001322 013426 EM6 :IBWC/IBCA DATA ERROR
91 001324 014447 DH3 :TEST ERRPC GOOD BAD
92 001326 014616 DT3 :$TESTN,$ERRPC,$GDDAT,$BDDAT
93 001330 014652 DFO :ALL NUMBERS ARE IN OCTAL FORM.
94 :ITEM 7
95 001332 013455 EM7 :INTERRUPT ERROR
96 001334 014535 DH7 :TEST ERRPC TO FROM ADDR.
97 001336 014640 DT7 :$TESTN,$ERRPC,TRTO,TRFRO
98 001340 014652 DFO :ALL NUMBERS ARE IN OCTAL FORM.
99 :ITEM 10
100 001342 013477 014447 014616 EM10,DH3,DT3,DFO :POSSIBLE -YA VARIATION ERROR
101 001350 014652
102 :ITEM 11
001352 014023 014447 014616 EM11,DH3,DT3,DFO :POSSIBLE 'ER1-INH' SWITCH ERROR
001360 014652

```

```

162 .SB1TL REG ADDRESS AND COMMON TAGS
163 :WARNING IF DEVICE # IS AT DIFFERENT ADDRESS OR VECTOR
164 :DO NOT PATCH THESE LOCATIONS - SEE PROGRAM DOCUMENTATION.
165 001362 160150 IBS: .WORD ABASE ;>NO <;CONTROL AND STATUS REGISTER.
166 001364 160152 IBD: .WORD ABASE+2 ;>PATCHES <;DATA REGISTER.
167 001366 160154 IBWC: .WORD ABASE+4 ;ADDRESS RESERVED FOR
168 001370 160156 IBCA: .WORD ABASE+6 ;FUTURE USE
169 001372 000420 VECTA: .WORD AVECT1 ;>ALLOWED <;VECTOR ADDRESS.
170 001374 000424 VECTB: .WORD AVECT1+4 ;>HERE! <;VECTOR ADDR. +4.
171 001376 000430 VECTC: .WORD AVECT1+10
172 001400 000434 VECTD: .WORD AVECT1+14
173 001402 160160 IBS2: .WORD ABASE+10
174 001404 160162 IBD2: .WORD ABASE+12
175 001406 000660 VECTA2: .WORD 660
176 001410 000664 VECTB2: .WORD 664
177 001412 000670 VECTC2: .WORD 670
178 001414 000674 VECTD2: .WORD 674
179
180 :VECTOR ADDRESSES +2 LOCATIONS.
181 001416 000422 PRA: .WORD AVECT1+2 ;NOTE: DO NOT ATTEMPT TO PATCH
182 001420 000426 PRB: .WORD AVECT1+6 ; THESE LOCATIONS IF A VECTOR
183 001422 000432 PRC: .WORD AVECT1+12 ; VARIES. ALTER LOCATION
184 001424 000436 PRD: .WORD AVECT1+16 ; 'SVECT1:'.
185 001426 000442 PRA2: .WORD AVECT1+22 ; IF TEST MODULE VECTOR IS
186 001430 000446 PRB2: .WORD AVECT1+26 ; DIFFERENT, YOU MUST CHANGE
187 001432 000452 PRC2: .WORD AVECT1+32 ; LOCATION 'VECTA2:'.
188 001434 000456 PRD2: .WORD AVECT1+36 ;
189
190 001436 000002 CPUDLY: 2 ;25 USEC LOOP FACTOR
191 001440 002 CPUSPD: .BYTE 2 ;0 (PDP-11)
192 001441 005 .BYTE 5 ;1 (PDP-11/04)
193 001442 004 .BYTE 4 ;2 (PDP-11/05)
194 001443 002 .BYTE 2 ;3 (PDP-11/20)
195 001444 011 .BYTE 9. ;4 (PDP-11/40)
196 001445 014 .BYTE 12. ;5 (PDP-11/45)
197 001446 014 .BYTE 12. ;6 (PDP-11/70)
198 001447 002 .BYTE 2 ;7 (PDP-)
199 001450 002 .BYTE 2 ;10 (PDP-11/03) ** DEFAULT VALUE **
200 001451 006 .BYTE 6 ;11 (PDP-11/23)
201 001452 006 .BYTE 6 ;12 (PDP-11/34)
202 001453 002 .BYTE 2 ;13 (PDP-11/)
203 001454 002 .BYTE 2 ;14 (PDP-11/)
204 001455 002 .BYTE 2 ;15 (PDP-11/)
205 001456 002 .BYTE 2 ;16 (PDP-11/)
206 001457 002 .BYTE 2 ;17 (PDP-11/)

```

```
208 .SBITL PROGRAM START
209
211
212 001460 START:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ::CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001460 012706 001100 MOV #CMTAG,R6 ::FIRST LOCATION TO BE CLEARED
(1) 001464 005026 CLR (R6)+ ::CLEAR MEMORY LOCATION
(1) 001466 022706 001140 CMP #SWR,R6 ::DONE?
(1) 001472 001374 BNE -6 ::LOOP BACK IF NO
(1) 001474 012706 001100 MOV #STACK,SP ::SETUP THE STACK POINTER
(1) ::INITIALIZE A FEW VECTORS
(1) 001500 012737 010426 000020 MOV #SCOPE,@IOTVEC ::IOT VECTOR FOR SCOPE ROUTINE
(1) 001506 012737 000340 000022 MOV #340,@IOTVEC+2 ::LEVEL 7
(1) 001514 012737 010120 000030 MOV #ERROR,@EMTVEC ::EMT VECTOR FOR ERROR ROUTINE
(1) 001522 012737 000340 000032 MOV #340,@EMTVEC+2 ::LEVEL 7
(1) 001530 012737 013176 000034 MOV #STRAP,@TRAPVEC ::TRAP VECTOR FOR TRAP CALLS
(1) 001536 012737 000340 000036 MOV #340,@TRAPVEC+2:LEVEL 7
(1) 001544 012737 012750 000024 MOV #SPWRDN,@PWRVEC ::POWER FAILURE VECTOR
(1) 001552 012737 000340 000026 MOV #340,@PWRVEC+2 ::LEVEL 7
(1) 001560 005037 001160 CLR $TIMES ::INITIALIZE NUMBER OF ITERATIONS
(1) 001564 005037 001162 CLR $ESCAPE ::CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001570 112737 000001 001115 MOVB #1,$ERMAX ::ALLOW ONE ERROR PER TEST
(1) 001576 012737 001576 001106 MOV #.,$LPADR ::INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001604 012737 001604 001110 MOV #.,$LPERR ::SETUP THE ERROR LOOP ADDRESS
(2) ::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ::EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001612 013746 000004 MOV @ERRVEC,-(SP) ::SAVE ERROR VECTOR
(2) 001616 012737 001652 000004 MOV #64,$ERRVEC ::SET UP ERROR VECTOR
(2) 001624 012737 177570 001140 MOV #DSWR,$SWR ::SETUP FOR A HARDWARE SWICH REGISTER
(2) 001632 012737 177570 001142 MOV #DDISP,$DISPLAY ::AND A HARDWARE DISPLAY REGISTER
(2) 001640 022777 177777 177272 CMP #-1,$SWR ::TRY TO REFERENCE HARDWARE SWR
(2) 001646 001012 BNE 66$ ::BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) 65$ BR 65$ ::AND THE HARDWARE SWR IS NOT = -1
(2) 001650 000403 BR 65$ ::BRANCH IF NO TIMEOUT
(2) 001652 012716 001660 64$: MOV #65$,(SP) ::SET UP FOR TRAP RETURN
(2) 001656 000002 RTI
(2) 001660 012737 000176 001140 65$: MOV #SWREG,$SWR ::POINT TO SOFTWARE SWR
(2) 001666 012737 000174 001142 MOV #DISPREG,$DISPLAY
(2) 001674 012637 000004 66$: MOV (SP)+,@ERRVEC ::RESTORE ERROR VECTOR
(1)
(2) 001700 005037 001176 CLR $PASS ::CLEAR PASS COUNT
(2) 001704 132737 000200 001211 BITB #APTSIZE,$ENVM ::TEST USER SIZE UNDER APT
(2) 001712 001403 BEQ 67$ ::YES,USE NON-APT SWITCH
(2) 001714 012737 001212 001140 MOV #$$SWREG,$SWR ::NO,USE APT SWITCH REGISTER
(2) 001722 67$:
213 001722 012737 000300 000022 MOV #300,IOTVEC+2 ::LOWER PS TO LEVEL 6 FOR FALCON CPU
214 001730 012737 000300 000032 MOV #300,EMTVEC+2
215 001736 012737 000300 000036 MOV #300,TRAPVEC+2
216 001744 012737 000300 000026 MOV #300,PWRVEC+2
217 001752 012737 013126 000004 MOV #IOTRD,$ERRVEC ::SET TO HANDLE BUS ERRORS.
218 001760 012737 000200 000006 MOV #200,$ERRVEC+2
219
220 001766 013737 001244 001362 MOV $BASE,$IBS ::GET BASE ADDR.
221 001774 013737 001362 001364 MOV $IBS,$IBD ::FIX DATA BUFFER=
222 002002 062737 000002 001364 ADD #2,$IBD ::CSR+2
```

INITIALIZE THE COMMON TAGS

```

223 002010 013737 001364 001366      MOV      IBD,IBWC
224 002016 062737 000002 001366      ADD      #2,IBWC
225 002024 013737 001366 001370      MOV      IBWC,IBCA
226 002032 062737 000002 001370      ADD      #2,IBCA
227 002040 013737 001240 001372      MOV      $VECT1,VECTA      ;GET VECTOR ADDR.
228 002046 042737 170000 001372      BIC      #170000,VECTA      ;STRIP JUNK
229 002054 013737 001362 014610      MOV      IBS,IBSA
230 002062 013737 001364 014612      MOV      IBD,IBDA
231 002070 013737 001402 001404      MOV      IBS2,IBD2
232 002076 062737 000002 001404      ADD      #2,IBD2
233 002104 013737 001372 001374      MOV      VECTA,VECTB
234 002112 062737 000004 001374      ADD      #4,VECTB
235 002120 013737 001374 001376      MOV      VECTB,VECTC
236 002126 062737 000004 001376      ADD      #4,VECTC
237 002134 013737 001376 001400      MOV      VECTC,VECTD
238 002142 062737 000004 001400      ADD      #4,VECTD
239 002150 013737 001406 001410      MOV      VECTA2,VECTB2
240 002156 062737 000004 001410      ADD      #4,VECTB2
241 002164 013737 001410 001412      MOV      VECTB2,VECTC2
242 002172 062737 000004 001412      ADD      #4,VECTC2
243 002200 013737 001412 001414      MOV      VECTC2,VECTD2
244 002206 062737 000004 001414      ADD      #4,VECTD2
245
246 002214 013737 001372 001416      MOV      VECTA,PRA      ;SET UP VECTOR+2 ADDRESSES.
247 002222 062737 000002 001416      ADD      #2,PRA
248 002230 013737 001416 001420      MOV      PRA,PRB
249 002236 062737 000004 001420      ADD      #4,PRB
250 002244 013737 001420 001422      MOV      PRB,PRC
251 002252 062737 000004 001422      ADD      #4,PRC
252 002260 013737 001422 001424      MOV      PRC,PRD
253 002266 062737 000004 001424      ADD      #4,PRD
254 002274 013737 001406 001426      MOV      VECTA2,PRA2
255 002302 062737 000002 001426      ADD      #2,PRA2
256 002310 013737 001426 001430      MOV      PRA2,PRB2
257 002316 062737 000004 001430      ADD      #4,PRB2
258 002324 013737 001430 001432      MOV      PRB2,PRC2
259 002332 062737 000004 001432      ADD      #4,PRC2
260 002340 013737 001432 001434      MOV      PRC2,PRD2
261 002346 062737 000004 001434      ADD      #4,PRD2
262 002354 013700 001216      RSTART: MOV      $CPUOP,R0      ;GET CPU TYPE
263 002360 042700 177760      BIC      #177760,R0      ;MASK OFF ANOTHER BITS
264 002364 116037 001440 001436      MOVB    CPUSPD(R0),CPUDLY ;GET DELAY FACTOR
265
266
(1) 002372 013777 001416 176772      MOV      PRA,@VECTA ;/-RESV-
(1) 002400 012777 004700 177010      MOV      #4700,@PRA ;/RESTORE VECTOR FOR
; /ILLEGAL INTRO.
267
(1) 002406 013777 001420 176760      MOV      PRB,@VECTB ;/-RESV-
(1) 002414 012777 004700 176776      MOV      #4700,@PRB ;/RESTORE VECTOR FOR
; /ILLEGAL INTRO.
268
(1) 002422 013777 001422 176746      MOV      PRC,@VECTC ;/-RESV-
(1) 002430 012777 004700 176764      MOV      #4700,@PRC ;/RESTORE VECTOR FOR
; /ILLEGAL INTRO.
269
(1) 002436 013777 001424 176734      MOV      PRD,@VECTD ;/-RESV-
(1) 002444 012777 004700 176752      MOV      #4700,@PRD ;/RESTORE VECTOR FOR
; /ILLEGAL INTRO.

```



```

271      .SBTTL  TYPE PROGRAM NAME
(1)      ::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002452 005227 177777      INC      #-1          ::FIRST TIME?
(1) 002456 001041             BNE      64$          ::BRANCH IF NO
(1) 002460 104401 002526      TYPE     ,65$        ::TYPE ASCIZ STRING
(2)      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002464 005737 000042      TST     @#42         ::ARE WE RUNNING UNDER XXDP/ACT?
(2) 002470 001012             BNE      66$          ::BRANCH IF YES
(2) 002472 123727 001210 000001  CMPB    $ENV,#1      ::ARE WE RUNNING UNDER APT?
(2) 002500 001406             BEQ     66$          ::BRANCH IF YES
(2) 002502 023727 001140 000176  CMP     SWR,#SWREG  ::SOFTWARE SWITCH REG SELECTED?
(2) 002510 001005             BNE     67$          ::BRANCH IF NO
(2) 002512 104406             GTSWR                    ::GET SOFT-SWR SETTINGS
(2) 002514 000403             BR      67$
(2) 002516 112737 000001 001134 66$:   MOVB    #1,$AUTOB   ::SET AUTO-MODE INDICATOR
(2) 002524             67$:
(1) 002524 000416             BR      64$          ::GET OVER THE ASCIZ
(1)      ::65$: .ASCIZ <CRLF>#CVIBAB IBV11A DIAGNOSTIC#<CRLF>
(1)      64$:
272 002562 000005             RESET
273 002564 004537 010100      JSR     R5,DEL50    ;DELAY FOR RESET PULSE TO FINISH
274 002570 001000             1000      ; <SO THAT ANY ERROR MESSAGES ARE NOT TRASHED>
275

```

```

277 -
(3)
(3)
(2) 002572 000240
(1) 002574 012737 000050 001160
(1) 002602 012737 002632 001106
278
279 002610 012737 000001 001102
280 002616 012737 000001 001174
281 002624 012737 002632 001110
282
283 002632 013746 000004
284 002636 012737 002664 000004
285
286
287
288 002644 005777 176512
289
290 002650 012737 002672 000004
291
292 002656 005777 176502
293
294 002662 000406
295 002664
(1) 002664 062706 000004
296

:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

(1)
(1) 002670 104005
297
298

:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$

300 002672
(1) 002672 062706 000004
301

:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

(1)
(1) 002676 104005
302

:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$

304 002700 012637 000004
305
(1) 002704 012746 000000
(1) 002710 012746 002716
(1) 002714 000002
(1) 002716
306
313

```

```

*****
*TEST 1 *TEST THE ADDRESSABILITY OF THE IBS, IBD REGISTERS
*****
TST1: NOP
MOV #50,$TIMES ;;DO 50 ITERATIONS
MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS

MOV #1,$STSTM ;SET TEST #1.
MOV #1,$TESTN ;DON'T FORGET APT!
MOV #1$,SLPERR

1$: MOV ERRVEC,-(SP) ;SAVE CONTENTS OF ADDR. 4
MOV #2$,ERRVEC ;SET TIME-OUT TRAP VECTOR TO HANDLE
;IN CASE WE TIME OUT WHEN
;WE ADDR. THE IBV-11

TST @IBS ;ADDR THE IBS, IF NO RESPONSE,
;WILL TRAP TO 2$ FROM HERE
MOV #3$,ERRVEC ;CHANGE FOR ADDRESSING THE IBD REG.

TST @IBD ;ADDR THE IBD REG.
;WE'LL TRAP TO 3$ FROM HERE IF BAD.

BR 4$

2$: ADD #4,$SP ;/ADD #4 TO STACK POINTER.

:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

ERROR 5 ;/MODULE FAULT DETECTED:
;IBS REGISTER COULD NOT BE
;ADDRESSED

:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$

3$: ADD #4,$SP ;/ADD #4 TO STACK POINTER.

:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

ERROR 5 ;/MODULE FAULT DETECTED:
;ADDRESSED

:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$

4$: MOV (SP)+,ERRVEC ;RESTORE CONTENTS OF LOC 4.
; /PR
MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
RTI ;/CAUSE A RETURN (PUTS NEW STATUS
; /IN STATUS REG.)

64$:

```

```

315          ::*****
(3)          ::*TEST 2          *TEST THAT BASE ADDRESSES +4,+6 RESPOND WHEN ADDRESSED
(4)          ::*
(4)          ::*EVEN THOUGH THE BASE ADDRESS +4 AND +6 ARE NOT USED,
(4)          ::*THE IBV11A SHOULD RESPOND TO THEM
(4)          ::*
(3)          ::*****
(2) 002716 000004          TST2: SCOPE
(1) 002720 012737 000010 001160      MOV      #10,$TIMES          ;;DO 10 ITERATIONS
316
317 002726 013746 000004          MOV      ERRVEC,-(SP)        ;SAVE CONTENTS OF ADDR 4.
318 002732 012737 002760 000004      MOV      #1$,ERRVEC        ;SET TIME OUT TRAP VECTOR TO HANDLE
319
320
321 002740 005777 176422          TST      @IBWC              ;IN CASE WE TIME OUT WHEN WE
322
323
324 002744 012737 002770 000004      MOV      #2$,ERRVEC        ;ADDRESS THE IBV-11 ADDRESSES +4,+6.
325
326 002752 005777 176412          TST      @IBCA              ;TEST BASE ADDRESS +4, IF NO RESPONSE
327 002756 000407
328
329 002760
(1) 002760 062706 000004          1$:      ADD      #4,$SP          ;WILL TRAP TO 1$ FROM HERE
330
          ::$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$
(1)
(1) 002764 104005          ERROR 5          ;/MODULE FAULT DETECTED:
331
332
          ::$$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
334 002766 000403          BR      3$
335
336 002770
(1) 002770 062706 000004          2$:      ADD      #4,$SP          ;/ADD #4 TO STACK POINTER.
337
          ::$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$
(1)
(1) 002774 104005          ERROR 5          ;/MODULE FAULT DETECTED:
338
339
          ::$$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
341 002776 012637 000004          3$:      MOV      (SP)+,ERRVEC    ;RESTORE CONTENTS OF LOC 4.
342
343

```

345  
(3)  
(3)  
(2) 003002 000004  
(1) 003004 012737 000001 001160  
346  
347 003012 000005  
348  
349 003014 005037 001124  
350 003020 017737 176336 001126  
351 003026 001401  
352

```
*****  
*TEST 3 *TEST THAT IBS IS CLEAR AT INIT OF TESTING  
*****  
TST3: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
  
RESET ;ISSUE SYSTEM INIT.  
  
CLR $GDDAT ;EXPECT ZERO CSR.  
MOV @IBS,$BDDAT ;READ CSR.  
BEQ TST4 ;;
```

;;\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 003030 104003  
353

ERROR 3 ;/MODULE FAULT DETECTED:  
;IBS NOT CLEAR ON INT.

;;\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

355  
356  
(3)  
(3)  
(2) 003032 000004  
(1) 003034 012737 000001 001160  
357  
358 003042 000005  
359  
360 003044 005037 001124  
361 003050 117737 176310 001126  
362 003056 001401  
363

```
*****  
*TEST 4 *TEST THAT IBD IS CLEAR AT INIT OF TESTING  
*****  
TST4: SCOPE  
MOV #1,$TIMES ;;DO 1 ITERATION  
  
RESET ;ISSUE SYSTEM INITIALIZE.  
  
CLR $GDDAT ;EXPECT ZERO CSR.  
MOVB @IBD,$BDDAT ;READ DBR  
BEQ TST5 ;;
```

;;\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 003060 104004  
364

ERROR 4 ;/MODULE FAULT DETECTED:  
;IBD NOT CLEAR ON INIT.

;;\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

367  
 374  
 375  
 (3)  
 (4)  
 (4)  
 (4)  
 (4)  
 (3)  
 (2) 003062 000004  
 (1) 003064 012737 000010 001160  
 376  
 377 003072 005037 001124  
 378 003076 017737 176264 001126  
 379 003104 001402  
 380

\*\*\*\*\*  
 : \*TEST 5 \*TEST THAT BASE ADDRESSES +4,+6 RETURN ZERO WHEN READ  
 : \*  
 : \*BASE ADDRESS +4 AND +6 SHOULD RETURN A ZERO WHEN  
 : \*READ, IN THIS TEST WE WILL TRY THAT.  
 : \*

\*\*\*\*\*  
 TST5: SCOPE  
 MOV #10,\$TIMES ;;DO 10 ITERATIONS  
 CLR \$GDDAT ;EXPECT ZERO RETURN  
 MOV @IBWC,\$BDDAT ;READ BASE ADDRESS+4  
 BEQ 1\$ ;IF ZERO - GOOD.

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003106 104006  
 381  
 382

ERROR 6 ;/MODULE FAULT DETECTED:  
 ;SHOULD HAVE READ BACK ZERO FROM  
 ;THIS ADDR.

384 003110 000405  
 385  
 386 003112 017737 176252 001126 1\$:  
 387 003120 001401  
 388

\*\*\*\*\*  
 BR TST6 ;  
 MOV @IBCA,\$BDDAT ;READ BASE ADDR+6, SHOULD BE ZERO  
 BEQ TST6 ;

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003122 104006  
 389  
 390

ERROR 6 ;/MODULE FAULT DETECTED:  
 ;SHOULD HAVE READ BACK ZERO FROM  
 ;BASE ADDR+6.

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$

392

394  
 (3)  
 (3)  
 (2) 003124 000004  
 395  
 396 003126 005077 176230  
 397 003132 052777 000001 176222  
 398 003140 012737 002001 001124  
 399 003146 017737 176210 001126  
 400 003154 023737 001124 001126  
 401 003162 000402  
 402

```

:*****
:*TEST 6 *TEST THAT WE CAN SET TCS (BIT00), TCS SETS CMD (BIT10)
:*****
TST6: SCOPE
CLR @IBS ;CLEAR CLR
BIS #BIT0,@IBS ;SET TCS.
MOV #BIT0!BIT10,$GDDAT ;EXPECT ONLY TCS AND CMD TO SET
MOV @IBS,$BDDAT ;READ THE IBS.
CMP $GDDAT,$BDDAT ;DID TCS AND CMD SET?
BR 1$ ;YES - CONTINUE

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003164 104003  
 403

```

ERROR 3 ;/MODULE FAULT DETECTED:
;TCS AND/OR CMD FAILED TO SET

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

405 003166 000412  
 406  
 407 003170 042777 000001 176164 1\$:  
 408 003176 005037 001124  
 409 003202 017737 176154 001126  
 410 003210 001401  
 411

```

BR TST7 ;:
BIC #BIT0,@IBS ;CLEAR TCS.
CLR $GDDAT ;EXPECT TCS AND CMD TO CLEAR
MOV @IBS,$BDDAT ;READ IBS, DID THEY CLEAR?
BEQ TST7 ;:

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003212 104003  
 412

```

ERROR 3 ;/MODULE FAULT DETECTED:
;TCS AND/OR CMD FAILED TO CLEAR.

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

414  
 415  
 (3)  
 (3)  
 (2) 003214 000004  
 416  
 417 003216 005077 176140  
 418 003222 052777 000002 176132  
 419 003230 012737 000002 001124  
 420 003236 017737 176120 001126  
 421 003244 023737 001124 001126  
 422 003252 001402  
 423

```

:*****
:*TEST 7 *TEST THAT EOP (BIT01) WILL SET
:*****
TST7: SCOPE

```

```

CLR @IBS ;CLEAR CSR.
BIS #BIT1,@IBS ;SET EOP
MOV #BIT1,$GDDAT ;EXPECT ONLY EOP TO SET.
MOV @IBS,$BDDAT ;READ IBS
CMP $GDDAT,$BDDAT ;DID EOP SET?
BEQ 1$ ;YES - CONTINUE

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003254 104003  
 424

```

ERROR 3 ;/MODULE FAULT DETECTED:
;EOP BIT SETTING ERROR.

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

\*TEST THAT EOP (BIT01) WILL SET

```

426 003256 000412 BR TST10 ;;
427
428 003260 042777 000002 176074 1$: BIC #BIT1,@IBS ;CLEAR EOP
429 003266 005037 001124 CLR $GDDAT ;EXPECT A ZERO CSR.
430 003272 017737 176064 001126 MOV @IBS,$BDDAT ;READ IBS, IS IT CLEAR?
431 003300 001401 BEQ TST10 ;;
432

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 003302 104003 ERROR 3 ;/MODULE FAULT DETECTED:
433 ;IBS FAILED TO CLEAR

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

```

435
436 *****
(3) :*TEST 10 *TEST THAT REM (BIT02) WILL SET + CLEAR
(3) :*****
(2) 003304 000004 TST10: SCOPE
437
438 003306 005077 176050 CLR @IBS ;CLEAR CSR.
439 003312 052777 000004 176042 BIS #BIT02,@IBS ;SET REM
440 003320 012737 000004 001124 MOV #BIT02,$GDDAT ;EXPECT ONLY REM TO SET.
441 003326 017737 176030 001126 MOV @IBS,$BDDAT ;READ IBS.
442 003334 023737 001126 001124 CMP $BDDAT,$GDDAT ;DID REM AND ONLY REM SET?
443 003342 001402 BEQ 1$ ;YES - CONTINUE
444

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 003344 104003 ERROR 3 ;/MODULE FAULT DETECTED:
445 ;REM BIT SETTING ERROR.

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

```

447 003346 000412 BR TST11 ;;
448
449 003350 042777 000004 176004 1$: BIC #BIT02,@IBS ;CLEAR REM BIT.
450 003356 005037 001124 CLR $GDDAT ;EXPECT ZERO CSR.
451 003362 017737 175774 001126 MOV @IBS,$BDDAT ;READ IBS - IS IT CLEAR?
452 003370 001401 BEQ TST11 ;;
453

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 003372 104003 ERROR 3 ;/MODULE FAULT DETECTED:
454 ;IBS FAILED TO CLEAR.

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

456

458  
 (3)  
 (3)  
 (2) 003374 000004  
 459 003376 005077 175760  
 460 003402 052777 000010 175752  
 461 003410 012737 000010 001124  
 462 003416 017737 175740 001126  
 463 003424 023737 001124 001126  
 464 003432 001414  
 465 003434 012777 000010 175720  
 466 003442 017737 175714 001126  
 467 003450 023737 001124 001126  
 468 003456 001402  
 469

```

*****
*TEST 11 *TEST THAT IBC (BIT03) WILL SET AND CLEAR
*****
TST11: SCOPE
CLR @IBS ;CLEAR CSR.
BIS #BIT03,@IBS ;SET IBC
MOV #BIT03,$GDDAT ;EXPECT ONLY IBC TO BE SET
MOV @IBS,$BDDAT ;READ IBS.
CMP $GDDAT,$BDDAT ;DID IBS SET?
BEQ 1$ ;YES CONTINUE
MOV #BIT03,@IBS ;TRY SETTING IBC AGAIN.
MOV @IBS,$BDDAT ;MEMORY REFRESH MIGHT HAVE
CMP $GDDAT,$BDDAT ;GOT IN THE WAY.
BEQ 1$

```

:::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003460 104003  
 470  
 471 003462 000416  
 472 003464 004537 010100  
 473 003470 000006  
 474 003472 012737 002001 001124  
 475 003500 017737 175656 001126  
 476 003506 023737 001124 001126  
 477 003514 001401  
 478

```

ERROR 3 ;/MODULE FAULT DETECTED:
;IBS BIT SETTING ERROR.
BR TST12
JSR R5,DEL50 ;DELAY 150 US.
;WORD 6
MOV #BIT10!BIT0,$GDDAT ;EXP CMD AND TCS.
MOV @IBS,$BDDAT ;READ IBS - IS IT CLEAR?
CMP $GDDAT,$BDDAT
BEQ TST12 ;:

```

:::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003516 104010  
 479  
 480  
 481  
 (3)  
 (3)  
 (2) 003520 000004  
 482  
 483 003522 005077 175634  
 484 003526 052777 000040 175626  
 485 003534 012737 001040 001124  
 486 003542 017737 175614 001126  
 487 003550 023737 001124 001126  
 488 003556 001402  
 489  
 490

```

ERROR 10 ;/MODULE FAULT DETECTED:
;IBS NOT CLEAR AFTER IBC
;DOES "SCPUOP" INDICATE THE CORRECT CODE FOR THIS CPU ? OR

```

```

*****
*TEST 12 *TEST THAT TON (BIT05) AND TKR (BIT09) SET AND CLEAR
*****
TST12: SCOPE
CLR @IBS ;CLEAR THE CSR.
BIS #BIT5,@IBS ;SET TON.
MOV #BIT5!BIT9,$GDDAT ;EXPECT ONLY TON AND TKR TO SET.
MOV @IBS,$BDDAT ;READ CSR.
CMP $GDDAT,$BDDAT ;DID THEY BOTH SET?
BEQ 1$ ;YES - CONTINUE.

```

:::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$

(1)  
 (1) 003560 104003  
 491

```

ERROR 3 ;/MODULE FAULT DETECTED:
;ERROR IN SETTING TON BIT OR TKR BIT.

```



493 003562 000412

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$  
BR TST13 ::

494  
495 003564 042777 000040 175570 1S:

BIC #BIT5,@IBS ;WHEN TON CLEARED, TKR SHOULD CLEAR.

496 003572 005037 001124

CLR \$GDDAT ;EXPECT ZERO CSR.

497 003576 017737 175560 001126

MOV @IBS,\$BDDAT ;DID IT CLEAR?

498 003604 001401

BEQ TST13 ::

499

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 003606 104003

ERROR 3 ;/MODULE FAULT DETECTED:  
;CSR FAILED TO CLEAR.

500

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$

502

\*TEST THAT LON (BIT04) WILL SET AND CLEAR

504  
(3)  
(3)  
(2) 003610 000004  
505 003612 005077 175544  
506 003616 052777 000020 175536  
507 003624 012737 000020 001124  
508 003632 017737 175524 001126  
509 003640 023737 001124 001126  
510 003646 001402  
511

```
*****  
: *TEST 13 *TEST THAT LON (BIT04) WILL SET AND CLEAR  
:*****  
TST13: SCOPE  
CLR @IBS ;CLEAR THE CSR.  
BIS #BIT4,@IBS ;SET LON  
MOV #BIT4,$GDDAT ;EXPECT ONLY LON TO BE SET.  
MOV @IBS,$BDDAT ;READ CSR  
CMP $GDDAT,$BDDAT ;DID THEY BOTH SET ?  
BEQ 1$ ;;YES - CONTINUE
```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 003650 104003  
512

ERROR 3 ;/MODULE FAULT DETECTED:  
;ERROR IN SETTING LON BIT

514 003652 000412  
515 003654 042777 000020 175500 1\$:  
516 003662 005037 001124  
517 003666 017737 175470 001126  
518 003674 001401  
519

```
*****  
: *TEST 14 *TEST THAT LON (BIT04) WILL SET AND CLEAR  
:*****  
TST14: SCOPE  
BR TST14 ;  
BIC #BIT4,@IBS ;WHEN LON CLEARED, LNK SHOULD CLEAR  
CLR $GDDAT ;EXPECT ZERO CSR  
MOV @IBS,$BDDAT ;DID IT CLEAR ?  
BEQ TST14 ;;YES - CONTINUE
```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 003676 104003  
520

ERROR 3 ;/MODULE FAULT DETECTED:  
;CSR FAILED TO CLEAR LON BIT

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

523 ::*****
(3) ::*TEST 14 *TEST THAT IE (BIT06) SET AND CLEAR
(3) ::*****
(2) 003700 000004 TST14: SCOPE
524
525 :/PR
(1) 003702 012746 000300 MOV #300,-(SP) ;/SET CPU PRIORITY ON RETURN
(1) 003706 012746 003714 MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
(1) 003712 000002 RTI ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 003714 64$: ;/IN STATUS REG.)
526
527 003714 005077 175442 CLR @IBS ;CLEAR CSR.
528
529 003720 052777 000100 175434 BIS #BIT6,@IBS ;SET IE.
530 003726 012737 000100 001124 MOV #BIT6,$GDDAT ;EXPECT ONLY BIT 6 TO SET.
531 003734 017737 175422 001126 MOV @IBS,$BDDAT ;READ IBS.
532 003742 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID IE SET?
533 003750 001402 BEQ 1$ ;YES - CONTINUE.
534

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 003752 104003 ERROR 3 ;/MODULE FAULT DETECTED:
535 ;ERROR IN SETTING IE BIT.

```

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$

```

537 003754 000412 BR TST15 ;;
538
539 003756 005037 001124 1$: CLR $GDDAT ;EXPECT ZERO CSR AFTER.
540 003762 042777 000100 175372 BIC #BIT6,@IBS ;IE IS CLEARED.
541 003770 017737 175366 001126 MOV @IBS,$BDDAT ;READ CSR - IS IT CLEAR?
542 003776 001401 BEQ TST15 ;;
543

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 004000 104003 ERROR 3 ;/MODULE FAULT DETECTED:
544 ;FAILED TO CLEAR CSR.

```

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$

546

```

548 (3) *****
549 (3) *TEST 15 *TEST THAT ACC (BIT7) CAN BE SET AND CLEARED
550 (2) 004002 000004 TST15: SCOPE
551 004004 005077 175352 CLR @IBS ;CLEAR CSR.
552 004010 052777 000200 175344 BIS #BIT7,@IBS ;SET ACC.
553 004016 012737 000200 001124 MOV #BIT7,$GDDAT ;EXPECT ONLY ACC TO SET.
554 004024 017737 175332 001126 MOV @IBS,$BDDAT ;READ IBS.
555 004032 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID ACC SET?
556 004040 001402 BEQ 1$ ;YES - CONTINUE.

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1) 004042 104003 ERROR 3 ;/MODULE FAULT DETECTED:
557 ;FAILURE IN SETTING BIT 7 (ACC).

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

```

559 004044 000412 BR TST16 ;:
560
561 004046 042777 000200 175306 1$: BIC #BIT7,@IBS ;TRY CLEARING ACC.
562 004054 005037 001124 CLR $GDDAT ;EXPECT ZERO CSR.
563 004060 017737 175276 001126 MOV @IBS,$BDDAT ;READ IBS, IS IT CLEAR?
564 004066 001401 BEQ TST16 ;:
565

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1) 004070 104003 ERROR 3 ;/MODULE FAULT DETECTED:
566 ;IBS FAILED TO CLEAR.

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

568

```

570 *****
(3) *TEST 16 * TEST THAT SRQ (BIT15) CAN BE SET AND CLEARED (IF SWR10=1)
(3) *****
(2) TST16: SCOPE
571 004072 000004 CLR @IBS :CLEAR CSR
572 004074 005077 175262 BIS #BIT15,@IBS :SET SRQ
573 004100 052777 100000 175254 MOV #BIT15,$GDDAT :LOAD EXPECTED VALUE
574 004106 012737 100000 001124 BIT #BIT10,@SWR :TEST IF 'ER1-INHIBIT' SWITCH IS SET (CLOSED-ON)
575 004114 032777 002000 175016 BNE 1$ :BR IF SWITCH IS SET
576 004122 001002 CLR $GDDAT :RELOAD EXPECTED
577 004124 005037 001124 001126 1$: MOV @IBS,$BDDAT :READ IBS
578 004130 017737 175226 001126 CMP $GDDAT,$BDDAT :DID SRQ SET?
579 004136 023737 001124 001126 BEQ 2$ ::YES - CONTINUE
580

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1) ERROR 11 :/MODULE FAULT DETECTED:
(1) 004146 104011 :'SRQ' BIT IN THE IBS REGISTER IS IN ERROR
581 :IS SWR10 ('ER1-INHIBIT SWITCH') CORRECT ?
582

```

```

584 ***** ERROR *****
585 BR TST17
586 BIC #BIT15,@IBS :CLEAR SRQ
587 CLR $GDDAT :CLEAR EXPECTED
588 MOV @IBS,$BDDAT :READ IBS BITS
589 BPL TST17 :BR IF CLEARED

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1) ERROR 3 :/MODULE FAULT DETECTED:
(1) 004174 104003 :FAILED TO CLEAR IBS 'SRQ' BIT
590

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

618

```
620
(5)
(4)
(4)
(3) 004176 000004
(1)
(1)
(1) 004200 012777 000060 175154
(1) 004206 012737 000001 001124
(1) 004214 013777 001124 175142
(1)
(1) 004222 117737 175136 001126
(1) 004230 123737 001124 001126
(1) 004236 001402
(1)
(2)

:::SSSSSSSSSS>>> ERROR <<<SSSSSSSSSS

(2)
(2) 004240 104004
(1)

ERROR 4 :/MODULE FAULT DETECTED:
:/ERROR IN SETTING IBD BIT 0.

:::SSSSSSSSSS^^^ ERROR ^^SSSSSSSSSS
(3) 004242 000412
(1) 004244 005037 001124 175106 1$: BR TST20
(1) 004250 042777 000001 175106 CLR $GDDAT
(1) 004256 117737 175102 001126 BIC #BIT0,@IBD
(3) 004264 001401 MOVB @IBD,$BDDAT
(2) BEQ TST20

:/EXPECT ZERO IBD WHEN
:/BIT 0 IS CLEARED.
:/READ IBD, IS IT CLEAR?

:::SSSSSSSSSS>>> ERROR <<<SSSSSSSSSS

(2)
(2) 004266 104004
(1)

ERROR 4 :/MODULE FAULT DETECTED:
:/FAILED TO CLEAR IBD.

:::SSSSSSSSSS^^^ ERROR ^^SSSSSSSSSS

(1)
```

622  
(5)  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(2)  
  
(2)  
(2)  
(1)  
  
(3)  
(1)  
(1)  
(1)  
(1)  
(3)  
(2)  
  
(2)  
(2)  
(1)  
  
(1)

004270 000004  
  
004272 012777 000060 175062  
004300 012737 000002 001124  
004306 013777 001124 175050  
  
004314 117737 175044 001126  
004322 123737 001124 001126  
004330 001402  
  
004332 104004  
  
004334 000412  
004336 005037 001124  
004342 042777 000002 175014  
004350 117737 175010 001126  
004356 001401

```
*****  
*TEST 20 *TEST THAT IBD BIT 1 CAN BE SET + CLEARED  
*****  
TST20: SCOPE  
  
:/MACRO BDT  
:/SET TON AND LON.  
:/WE'RE GONNA TEST BIT 1.  
:/SET THE BIT.  
  
:/READ THE IBD.  
:/DID IT GET THRU OK?  
:/YES - CONTINUE.  
  
:;$$$$$$$$$>>> ERROR <<<$$$$$$$$$$  
  
ERROR 4 :/MODULE FAULT DETECTED:  
:/ERROR IN SETTING IBD BIT 1.  
  
:;$$$$$$$$$*** ERROR ***$$$$$$$$$$  
BR TST21 :;  
CLR $GDDAT :/EXPECT ZERO IBD WHEN  
BIC #BIT1,@IBD :/BIT 1 IS CLEARED.  
MOVB @IBD,$BDDAT :/READ IBD, IS IT CLEAR?  
BEQ TST21 :;  
  
:;$$$$$$$$$>>> ERROR <<<$$$$$$$$$$  
  
ERROR 4 :/MODULE FAULT DETECTED:  
:/FAILED TO CLEAR IBD.  
  
:;$$$$$$$$$*** ERROR ***$$$$$$$$$$
```

```

624
(5)
(4)
(4)
(3) 004362 000004
(1)
(1)
(1) 004364 012777 000060 174770      MOV    #BIT4!BITS,@IBS    ;/MACRO BDT
(1) 004372 012737 000004 001124      MOV    #BIT2,$GDDAT      ;/SET TON AND LON.
(1) 004400 013777 001124 174756      MOV    $GDDAT,@IBD      ;/WE'RE GONNA TEST BIT 2.
(1)
(1) 004406 117737 174752 001126      MOVB   @IBD,$BDDAT      ;/SET THE BIT.
(1) 004414 123737 001124 001126      CMPB   $GDDAT,$BDDAT    ;/READ THE IBD.
(1) 004422 001402                    BEQ    1$                ;/DID IT GET THRU OK?
(1)
(2)

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(2)
(2) 004424 104004      ERROR 4                ;/MODULE FAULT DETECTED:
(1)

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

```

(3) 004426 000412
(1) 004430 005037 001124 174722 1$: BR    TST22             ;/EXPECT ZERO IBD WHEN
(1) 004434 042777 000004 174722 BIC   #BIT2,@IBD      ;/BIT 2 IS CLEARED.
(1) 004442 117737 174716 001126 MOVB  @IBD,$BDDAT    ;/READ IBD, IS IT CLEAR?
(3) 004450 001401      BEQ    TST22         ;/
(2)

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(2)
(2) 004452 104004      ERROR 4                ;/MODULE FAULT DETECTED:
(1)

```

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

(1)



```
626
(5)
(4)
(4)
(3) 004454 000004
(1)
(1)
(1) 004456 012777 000060 174676
(1) 004464 012737 000010 001124
(1) 004472 013777 001124 174664
(1)
(1) 004500 117737 174660 001126
(1) 004506 123737 001124 001126
(1) 004514 001402
(1)
(2)
(2) 004516 104004
(1)
(3) 004520 000412
(1) 004522 005037 001124
(1) 004526 042777 000010 174630
(1) 004534 117737 174624 001126
(3) 004542 001401
(2)
(2) 004544 104004
(1)
(1)
```

```

*****
*TEST 22 *TEST THAT IBD BIT 3 CAN BE SET + CLEARED
*****
TST22: SCOPE

MOV #BIT4!BIT5,@IBS ;/MACRO BDT
MOV #BIT3,$GDDAT ;/SET TON AND LON.
MOV $GDDAT,@IBD ;/WE'RE GONNA TEST BIT 3.
; /SET THE BIT.

MOVB @IBD,$BDDAT ;/READ THE IBD.
CMPB $GDDAT,$BDDAT ;/DID IT GET THRU OK?
BEQ 1$ ;/YES - CONTINUE.

::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

ERROR 4 ;/MODULE FAULT DETECTED:
; /ERROR IN SETTING IBD BIT 3.

::$$$$$$$$$^^^ ERROR ^^ $$$$$$$$$$
BR TST23 ;
CLR $GDDAT ;/EXPECT ZERO IBD WHEN
BIC #BIT3,@IBD ;/BIT 3 IS CLEARED.
MOVB @IBD,$BDDAT ;/READ IBD, IS IT CLEAR?
BEQ TST23 ;

::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

ERROR 4 ;/MODULE FAULT DETECTED:
; /FAILED TO CLEAR IBD.

::$$$$$$$$$^^^ ERROR ^^ $$$$$$$$$$
```

```

628
(5)
(4)
(4)
(3) 004546 000004
(1)
(1)
(1) 004550 012777 000060 174604
(1) 004556 012737 000020 001124
(1) 004564 013777 001124 174572
(1)
(1) 004572 117737 174566 001126
(1) 004600 123737 001124 001126
(1) 004606 001402
(1)
(2)

```

```

:*****
:*TEST 23 *TEST THAT IBD BIT 4 CAN BE SET + CLEARED
:*****
TST23: SCOPE

```

```

:/MACRO BDT
:/SET TON AND LON.
MOV #BIT4,BIT5,@IBD ;/WE'RE GONNA TEST BIT 4.
MOV #BIT4,$GDDAT ;/SET THE BIT.
MOV $GDDAT,@IBD
MOVB @IBD,$BDDAT ;/READ THE IBD.
CMPB $GDDAT,$BDDAT ;/DID IT GET THRU OK?
BEQ 1$ ;/YES - CONTINUE.

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(2)
(2) 004610 104004
(1)

```

```

ERROR 4 ;/MODULE FAULT DETECTED:
;/ERROR IN SETTING IBD BIT 4.

```

```

(3) 004612 000412
(1) 004614 005037 001124
(1) 004620 042777 000020 174536
(1) 004626 117737 174532 001126
(3) 004634 001401
(2)

```

```

:*****
:*** ERROR ***
BR TST24 ;/EXPECT ZERO IBD WHEN
CLR $GDDAT ;/BIT 4 IS CLEARED.
BIC #BIT4,@IBD ;/READ IBD, IS IT CLEAR?
MOVB @IBD,$BDDAT
BEQ TST24 ;/

```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(2)
(2) 004636 104004
(1)

```

```

ERROR 4 ;/MODULE FAULT DETECTED:
;/FAILED TO CLEAR IBD.

```

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$

(1)

630  
(5)  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(2)  
  
(2)  
(2)  
(1)  
  
(3)  
(1)  
(1)  
(1)  
(1)  
(3)  
(2)  
  
(2)  
(2)  
(1)  
  
(1)

004640 000004  
  
004642 012777 000060 174512  
004650 012737 000040 001124  
004656 013777 001124 174500  
  
004664 117737 174474 001126  
004672 123737 001124 001126  
004700 001402  
  
004702 104004  
  
004704 000412  
004706 005037 001124  
004712 042777 000040 174444  
004720 117737 174440 001126  
004726 001401

```
::*****  
:*TEST 24 *TEST THAT IBD BIT 5 CAN BE SET + CLEARED  
:*****  
TST24: SCOPE  
  
:/MACRO BDT  
:/SET TON AND LON.  
:/WE'RE GONNA TEST BIT 5.  
:/SET THE BIT.  
  
:/READ THE IBD.  
:/DID IT GET THRU OK?  
:/YES - CONTINUE.  
  
:/$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$  
  
ERROR 4 :/MODULE FAULT DETECTED:  
:/ERROR IN SETTING IBD BIT 5.  
  
:/$$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$  
BR TST25 :/  
CLR $GDDAT :/EXPECT ZERO IBD WHEN  
BIC #BIT5,@IBD :/BIT 5 IS CLEARED.  
MOVB @IBD,$BDDAT :/READ IBD, IS IT CLEAR?  
BEQ TST25 :/  
  
:/$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$  
  
ERROR 4 :/MODULE FAULT DETECTED:  
:/FAILED TO CLEAR IBD.  
  
:/$$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
```

632  
(5)  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(2)  
  
(2)  
(2)  
(1)  
  
(3)  
(1)  
(1)  
(1)  
(3)  
(2)  
  
(2)  
(2)  
(1)  
  
(1)

004732 000004  
  
004734 012777 000060 174420  
004742 012737 000100 001124  
004750 013777 001124 174406  
  
004756 117737 174402 001126  
004764 123737 001124 001126  
004772 001402  
  
004774 104004  
  
004776 000412  
005000 005037 001124  
005004 042777 000100 174352  
005012 117737 174346 001126  
005020 001401  
  
005022 104004

```
::*****  
:*TEST 25 *TEST THAT IBD BIT 6 CAN BE SET + CLEARED  
:*****  
TST25: SCOPE  
  
:/MACRO BDT  
:/SET TON AND LON.  
:/WE'RE GONNA TEST BIT 6.  
:/SET THE BIT.  
  
:/READ THE IBD.  
:/DID IT GET THRU OK?  
:/YES - CONTINUE.  
  
:/$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$  
  
ERROR 4 :/MODULE FAULT DETECTED:  
:/ERROR IN SETTING IBD BIT 6.  
  
:/$$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$  
BR TST26 :/  
CLR $GDDAT :/EXPECT ZERO IBD WHEN  
BIC #BIT6,@IBD :/BIT 6 IS CLEARED.  
MOVB @IBD,$BDDAT :/READ IBD, IS IT CLEAR?  
BEQ TST26 :/  
  
:/$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$  
  
ERROR 4 :/MODULE FAULT DETECTED:  
:/FAILED TO CLEAR IBD.  
  
:/$$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
```

```

634
(5)
(4)
(4)
(3) 005024 000004
(1)
(1)
(1) 005026 012777 000060 174326
(1) 005034 012737 000200 001124
(1) 005042 013777 001124 174314
(1)
(1) 005050 117737 174310 001126
(1) 005056 123737 001124 001126
(1) 005064 001402
(1)
(2)

```

```

:*****
:*TEST 26 *TEST THAT IBD BIT 7 CAN BE SET + CLEARED
:*****
TST26: SCOPE

```

```

:/MACRO BDT
:/SET TON AND LON.
:/WE'RE GONNA TEST BIT 7.
:/SET THE BIT.

MOVB @IBD,$BDDAT
CMPB $GDDAT,$BDDAT
BEQ 1$

:/READ THE IBD.
:/DID IT GET THRU OK?
:/YES - CONTINUE.

```

:::SSSSSSSSSS>>> ERROR <<<SSSSSSSSSS

```

(2)
(2) 005066 104004
(1)

```

```

ERROR 4
:/MODULE FAULT DETECTED:
:/ERROR IN SETTING IBD BIT 7.

```

```

(3) 005070 000412
(1) 005072 005037 001124
(1) 005076 042777 000200 174260
(1) 005104 117737 174254 001126
(3) 005112 001401
(2)

```

```

:***** ERROR *****
BR TST27
CLR $GDDAT
BIC #BIT7,@IBD
MOVB @IBD,$BDDAT
BEQ TST27

:/EXPECT ZERO IBD WHEN
:/BIT 7 IS CLEARED.
:/READ IBD, IS IT CLEAR?

```

:::SSSSSSSSSS>>> ERROR <<<SSSSSSSSSS

```

(2)
(2) 005114 104004
(1)

```

```

ERROR 4
:/MODULE FAULT DETECTED:
:/FAILED TO CLEAR IBD.

```

:::SSSSSSSSSS^ ERROR ^^^SSSSSSSSSS

(1) 635

637  
(3)  
(3)  
(2) 005116 000004

::\*\*\*\*\*  
:\*TEST 27 \*TEST THAT NO DATA GETS XFERRERD, IF NOT ENABLED  
:\*\*\*\*\*  
TST27: SCOPE

638  
639 005120 005077 174236  
640 005124 112777 000252 174232  
641 005132 005037 001124  
642 005136 117737 174222 001126  
643 005144 001401  
644

CLR @IBS :CLEAR CSR  
MOVB #252,@IBD :TRY XFERRING DATA  
CLR \$GDDAT :NO DATA SHOULD XFERR  
MOVB @IBD,\$BDDAT :READ BUFFER REG.  
BEQ TST30 ::

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005146 104002  
645  
646  
647  
648

ERROR 2 :/MODULE FAULT DETECTED:  
:DATA WAS XFERRERD THROUGH IBD  
:EVEN THOUGH TON AND LON CLEARED.  
:SIGNAL 'ENB XFER L' PROBABLY  
:STUCK LOW.

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

650

652  
(3)  
(3)  
(2) 005150 000004  
653  
654 005152 005077 174204  
655 005156 005077 174202  
656 005162 032777 000400 174174  
657 005170 001002  
658

```
*****
*TEST 30 *TEST IBD-DAC (BIT08) AND IBD-DAV (BIT09) CAN GE SET + CLEARED
*****
TST30: SCOPE
:
: <<*ALSO TEST IBS-LNR (BIT08)>>
CLR @IBS ;CLEAR CSR.
CLR @IBD ;CLEAR DATA REG.
BIT #BIT8,@IBD ;IS DAC SET?
BNE 1$ ;YES (GOOD)
```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005172 104002  
659  
660 005174 000503  
661 005176 005037 001124  
662 005202 017737 174154 001126  
663 005210 032737 000400 001126  
664 005216 001402  
665

```
ERROR 2 ;/MODULE FAULT DETECTED:
;IBD-DAC NOT SET.
BR TST31
CLR $GDDAT ;CLEAR EXPECTED
MOV @IBS,$BDDAT ;READ STATUS
BIT #BIT8,$BDDAT ;IS LNR CLEARED?
BEQ 2$ ;BR IF LNR IS CLEARED
```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005220 104003  
666  
667 005222 000470  
668 005224 052777 000260 174130  
669 005232 012777 000252 174124  
670 005240 017737 174120 001126  
671 005246 032737 001000 001126  
672 005254 001002  
673

```
ERROR 3 ;/MODULE FAULT DETECTED:
;IBS-LNR SET IN ERROR
BR TST31
BIS #BIT5!BIT4!BIT7,@IBS ;SET TON, LON AND ACC BITS
MOV #252,@IBD ;PUT DATA IN IBD.
MOV @IBD,$BDDAT ;READ IBD.
BIT #BIT9,$BDDAT ;DID DAV SET?
BNE 3$ ;YES - CONTINUE.
```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005256 104002  
674  
675 005260 000451  
676 005262 012737 000660 001124  
677 005270 017737 174066 001126  
678 005276 032737 000400 001126  
679 005304 001002  
680

```
ERROR 2 ;/MODULE FAULT DETECTED:
;DAV FAILED TO SET.
BR TST31
MOV #660,$GDDAT ;LOAD EXPECTED
MOV @IBS,$BDDAT ;READ CSR
BIT #BIT8,$BDDAT ;TEST IF LNR SET
BNE 4$ ;BR IF IBS-LNR WAS SET
```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005306 104003  
681  
682 005310 000435  
683 005312 012777 000060 174042  
684 005320 105777 174040

```
ERROR 3 ;/MODULE FAULT DETECTED:
;IBS-LNR FAILED TO SET
BR TST31
MOV #BIT4!BIT5,@IBS ;CLEAR ACC BIT
TSTB @IBD ;READ LOW BYTE OF IBD.
```

685 005324 032777 000400 174032  
686 005332 001402  
687

BIT #BIT8,@IBD ;DID DAC CLEAR?  
BEQ 5\$ ;YES - CONTINUE.

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005334 104002  
688  
689 005336 000422  
690 005340 032777 001000 174016 5\$:  
691 005346 001402  
692

ERROR 2 ;/MODULE FAULT DETECTED:  
;DAC FAILED TO CLEAR.  
BR TST31 ;;  
BIT #BIT9,@IBD ;DID DAV CLEAR?  
BEQ 6\$ ;;

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005350 104002  
693  
694 005352 000414  
695 005354 005077 174002 6\$:  
696 005360 005037 001124  
697 005364 017737 173772 001126  
698 005372 023737 001124 001126  
699 005400 001401  
700

ERROR 2 ;/MODULE FAULT DETECTED:  
;IBD-DAV FAILED TO CLEAR.  
BR TST31 ;;  
CLR @IBS ;CLEAR STATUS  
CLR \$GDDAT ;CLEAR EXPECTED  
MOV @IBS,\$BDDAT ;READ CSR  
CMP \$GDDAT,\$BDDAT ;COMPARE  
BEQ TST31 ;;

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 005402 104001  
701  
744  
745

ERROR 1 ;/MODULE FAULT DETECTED:  
;CSR FAILED TO CLEAR  
;/MACRO -SIGC-

:::\*\*\*\*\*  
;\*TEST 31 \*TEST THAT IBD - REN SETS WHEN IBS - REM SETS, ALSO TEST CLEAR  
:\*\*\*\*\*  
TST31: SCOPE

(3) 005404 000004  
(1)  
(1) 005406 005077 173750  
(1) 005412 052777 000004 173742  
(1) 005420 032777 010000 173736  
(1) 005426 001011  
(1) 005430 052777 000004 173724  
(1)  
(1)  
(1) 005436 032777 010000 173720  
(1) 005444 001002  
(2)

CLR @IBS ;/CLEAR CSR.  
BIS #BIT2,@IBS ;/SET REN, SHOULD SET REN.  
BIT #BIT12,@IBD ;/DID REN SET?  
BNE 1\$ ;/YES - LETS TRY CLEARING IT.  
BIS #BIT2,@IBS ;/SET REN, MEMORY  
;/REFRESH COULD HAVE  
;/INTERRUPTED US.  
BIT #BIT12,@IBD ;/DID REN SET THIS TIME?  
BNE 1\$

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(2)  
(2) 005446 104002  
(1)

ERROR 2 ;/MODULE FAULT DETECTED:  
;/REN FAILED TO SET WHEN REN SET.



(3) 005450 000410  
 (1) 005452 042777 000004 173702 1\$:  
 (1) 005460 032777 010000 173676  
 (3) 005466 001401  
 (2)

```

:::$$$$$$$$$$$^ ERROR ^$$$$$$$$$$$
BR      TST32      ::
BIC     #BIT2,@IBS  :/CLEAR REN, SHOULD CLEAR REN.
BIT     #BIT12,@IBD :/DID REN CLEAR?
BEQ     TST32      ::
  
```

:::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$

(2)  
 (2) 005470 104002  
 (1)

```

ERROR 2      :/MODULE FAULT DETECTED:
           :/REN FAILED TO CLEAR.
  
```

:::\$\$\$\$\$\$\$\$\$\$\$^ ERROR ^\$\$\$\$\$\$\$\$\$\$\$

(1)  
 746

748  
 (1)  
 (5)  
 (4)  
 (4)  
 (3) 005472 000004  
 (1)  
 (1) 005474 005077 173662  
 (1) 005500 052777 000010 173654  
 (1) 005506 032777 020000 173650  
 (1) 005514 001011  
 (1) 005516 052777 000010 173636  
 (1)  
 (1)  
 (1) 005524 032777 020000 173632  
 (1) 005532 001002  
 (2)

:/MACRO -SIGC-  
 :\*\*\*\*\*  
 :\*TEST 32 \*TEST THAT IBD - IFC SETS WHEN IBS - IBC SETS, ALSO TEST CLEAR  
 :\*\*\*\*\*  
 TST32: SCOPE

CLR @IBS ;/CLEAR CSR.  
 BIS #BIT3,@IBS ;/SET IBC, SHOULD SET IFC.  
 BIT #BIT13,@IBD ;/DID IFC SET?  
 BNE 1\$ ;/YES - LETS TRY CLEARING IT.  
 BIS #BIT3,@IBS ;/SET IBC, MEMORY  
 ;/REFRESH COULD HAVE  
 ;/INTERRUPTED US.  
 BIT #BIT13,@IBD ;/DID IFC SET THIS TIME?  
 BNE 1\$

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(2)  
 (2) 005534 104002  
 (1)

ERROR 2 ;/MODULE FAULT DETECTED:  
 ;/IFC FAILED TO SET WHEN IBC SET.

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$  
 BR TST33 ;:

(3) 005536 000410  
 (1)  
 (1) 005540 1\$:  
 (1) 005540 004537 010100 2\$:  
 (1) 005544 000006  
 (1)  
 (1) 005546 032777 020000 173610  
 (3) 005554 001401  
 (2)

JSR R5,DEL50 ;/DELAY SOME TIME  
 .WORD 6  
 BIT #BIT13,@IBD ;/IBC CLEAR, DID IFC CLEAR?  
 BEQ TST33 ;:

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(2)  
 (2) 005556 104002  
 (1)

ERROR 2 ;/MODULE FAULT DETECTED:  
 ;/IFC FAILED TO CLEAR.

:::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

(1)  
 749

```
751
(1)
(5)
(4)
(4)
(3) 005560 000004
(1)
(1) 005562 005077 173574 CLR @IBS ;/CLEAR CSR.
(1) 005566 052777 000001 173566 BIS #BIT0,@IBS ;/SET TCS, SHOULD SET ATN.
(1) 005574 032777 040000 173562 BIT #BIT14,@IBD ;/DID ATN SET?
(1) 005602 001011 BNE 1$ ;/YES - LETS TRY CLEARING IT.
(1) 005604 052777 000001 173550 BIS #BIT0,@IBS ;/SET TCS, MEMORY
(1) ;/REFRESH COULD HAVE
(1) ;/INTERRUPTED US.
(1) 005612 032777 040000 173544 BIT #BIT14,@IBD ;/DID ATN SET THIS TIME?
(1) 005620 001002 BNE 1$
(2)
:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$
(2) 005622 104002 ERROR 2 ;/MODULE FAULT DETECTED:
(1) ;/ATN FAILED TO SET WHEN TCS SET.
:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
(3) 005624 000410 BR TST34 ;/
(1) 005626 042777 000001 173526 1$: BIC #BIT0,@IBS ;/CLEAR TCS, SHOULD CLEAR ATN.
(1) 005634 032777 040000 173522 BIT #BIT14,@IBD ;/DID ATN CLEAR?
(3) 005642 001401 BEQ TST34 ;/
(2)
:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$
(2) 005644 104002 ERROR 2 ;/MODULE FAULT DETECTED:
(1) ;/ATN FAILED TO CLEAR.
:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
(1)
752
```

```

754
(1)
(5)
(4)
(4)
(3) 005646 000004
(1)
(1) 005650 005077 173506 CLR @IBS ;/CLEAR CSR.
(1) 005654 052777 000002 173500 BIS #BIT1,@IBS ;/SET EOP, SHOULD SET EOI.
(1) 005662 032777 100000 173474 BIT #BIT15,@IBD ;/DID EOI SET?
(1) 005670 001011 BNE 1$ ;/YES - LETS TRY CLEARING IT.
(1) 005672 052777 000002 173462 BIS #BIT1,@IBS ;/SET EOP, MEMORY
(1) ;/REFRESH COULD HAVE
(1) ;/INTERRUPTED US.
(1) 005700 032777 100000 173456 BIT #BIT15,@IBD ;/DID EOI SET THIS TIME?
(1) 005706 001002 BNE 1$
(2)

:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

(2)
(2) 005710 104002 ERROR 2 ;/MODULE FAULT DETECTED:
(1) ;/EOI FAILED TO SET WHEN EOP SET.

:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
(3) 005712 000410 BR TST35 ;/
(1) 005714 042777 000002 173440 1$: BIC #BIT1,@IBS ;/CLEAR EOP, SHOULD CLEAR EOI.
(1) 005722 032777 100000 173434 BIT #BIT15,@IBD ;/DID EOI CLEAR?
(3) 005730 001401 BEQ TST35 ;/
(2)

:::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

(2)
(2) 005732 104002 ERROR 2 ;/MODULE FAULT DETECTED:
(1) ;/EOI FAILED TO CLEAR.

:::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$

(1)
755

```

757  
 758  
 (3)  
 (3)  
 (2) 005734 000004  
 759 005736 005077 173420  
 760 005742 052777 100000 173412  
 761 005750 032777 002000 173162  
 762 005756 001406  
 763 005760 032777 004000 173376  
 764 005766 001002  
 765

\*\*\*\*\*  
 : \*TEST 35 \*TEST THAT IBD-SRQ SETS WHEN IBS-SRQ SETS, ALSO TEST CLEAR (IF SWR10=1)  
 \*\*\*\*\*

TST35: SCOPE  
 CLR @IBS ;CLEAR CSR  
 BIS #BIT15,@IBS ;SET SRQ IN IBS SHOULD SET SRQ IN IBD  
 BIT #BIT10,@SWR ;TEST IF 'ER1-INHIBIT' SWITCH IS SET <CLOSED-ON>  
 BEQ 1\$ ;BR IF SWITCH IS CLEARED  
 BIT #BIT11,@IBD ;DID SRQ IN IBD SET THIS TIME ?  
 BNE 1\$ ;BR IF YES

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 005770 104002  
 766  
 767

ERROR 2 ;/MODULE FAULT DETECTED:  
 ;SRQ IN IBD DID NOT SET WHEN SRQ IN IBS SET  
 ; CHECK ER1-INHIBIT SWITCH

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

769 005772 000410  
 770 005774 042777 100000 173360 1\$:  
 771 006002 032777 004000 173354  
 772 006010 001401  
 773

BR 2\$ ;  
 BIC #BIT15,@IBS ;REMOVE IBS-SRQ BIT  
 BIT #BIT11,@IBD ;DID SRQ IN IBD STAY CLEARED ?  
 BEQ 2\$ ;BR IF STILL CLEARED

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 006012 104002  
 774  
 775

ERROR 2 ;/MODULE FAULT DETECTED:  
 ;SRQ IN IBD WAS SET WHEN SRQ IN IBS CLEARED  
 ; CHECK ER-1 INHIBIT SWITCH

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$  
 CLR @IBS ;ENSURE CLEARED STATUS

777 006014 005077 173342 2\$:

```

779 ::*****
(3) ::*TEST 36 *TEST THAT RFD SET WHEN CSR CLEAR, CLEAR WHEN ACC SET
(3) ::*****
(2) 006020 000004 TST36: SCOPE
780
781 006022 005077 173334 CLR @IBS ;CLEAR CSR.
782 006026 032777 002000 173330 BIT #BIT10,@IBD ;DID RFD SET?
783 006034 001002 BNE 1$ ;YES CONTINUE.
784

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 006036 104002 ERROR 2 ;/MODULE FAULT DETECTED:
785 ;RFD FAILED TO SET.

```

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

```

787 006040 000410 BR TST37 ;:
788
789 006042 052777 000200 173312 1$: BIS #BIT7,@IBS ;NOW SET ACC,RFD SHOULD CLEAR.
790 006050 032777 002000 173306 BIT #BIT10,@IBD ;DID IT CLEAR?
791 006056 001401 BEQ TST37 ;:
792

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 006060 104002 ERROR 2 ;/MODULE FAULT DETECTED:
793 ;RFD FAILED TO CLEAR.

```

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

```

795
796
797 ::*****
(3) ::*TEST 37 *TEST THAT WE CAN GENERATE AN ER2
(3) ::*****
(2) 006062 000004 TST37: SCOPE
798
799 006064 005077 173272 CLR @IBS ;CLEAR THE STATUS REG.
800 006070 052777 000041 173264 BIS #BITS!BIT0,@IBS ;SET TON; THIS SHOULD CAUSE AN
801 ;ERROR SENCE NO LISTENERS ARE ON
802 006076 105077 173262 CLRB @IBD ;AND WE SENT DATA TO THE BUS.
803 ;BUS.
804 006102 032777 040000 173252 BIT #BIT14,@IBS ;DID ER2 SET?
805 006110 001001 BNE TST40 ;:
806

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 006112 104001 ERROR 1 ;/MODULE FAULT DETECTED:
807 ;ER2 FAILED TO SET.

```

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

809

811  
(3)  
(3)  
(2) 006114 000004  
(1) 006116 012737 000005 001160  
812  
813 006124 012777 000367 173230  
814 006132 000005  
815 006134 105777 173222  
816 006140 001401  
817

```
::*****  
:*TEST 40 *TEST THAT BUS INIT CLEARS ACC,TON,LON,REM,EIP,TCS  
:*****  
TST40: SCOPE  
MOV #5,$TIMES ;;DO 5 ITERATIONS  
  
MOV #367,@IBS ;SET ACC,TON,LON,REM,EOP, AND TCS.  
RESET ;ISSUE SYS INIT.  
TSTB @IBS ;DID THEY ALL CLEAR?  
BEQ TST41 ;:
```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 006142 104001  
818  
819

ERROR 1 ;/MODULE FAULT DETECTED:  
;BUS INIT FAILED TO CLEAR CSR.

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

821  
822  
(3)  
(3)  
(2) 006144 000004  
(1) 006146 012737 000005 001160  
823  
824 006154 012777 000266 173200  
825 006162 052777 000010 173172  
826 006170 004537 010100  
827 006174 000006  
828 006176 032777 000266 173156  
829 006204 001401  
830

```
::*****  
:*TEST 41 *TEST IBC CLEARS ACC,TON,LON,REM AND EOP  
:*****  
TST41: SCOPE  
MOV #5,$TIMES ;;DO 5 ITERATIONS  
  
MOV #266,@IBS ;SET ACC,TON,LON,REM, AND EOP.  
BIS #BIT3,@IBS ;SET IBC, THIS SHOULD CLEAR ABOVE BITS.  
JSR R5,DEL50 ;DELAY 125 USEC  
.WORD 6  
BIT #266,@IBS ;DID THEY CLEAR?  
BEQ TST42 ;:
```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 006206 104001  
831  
832  
833

ERROR 1 ;/MODULE FAULT DETECTED:  
;ACC,TON,LON,REM, AND/OR EOP  
;FAILED TO CLEAR ON IBC

834  
(3)  
(3)  
(2) 006210 000004  
(1) 006212 012737 000005 001160  
835  
836 006220 012777 000260 173134  
837 006226 012777 000377 173130  
838 006234 000005  
839 006236 105777 173122  
840 006242 001401  
841

```
::*****  
:*TEST 42 *TEST THAT BUS INIT INDIRECTLY CLEARS IBD  
:*****  
TST42: SCOPE  
MOV #5,$TIMES ;;DO 5 ITERATIONS  
  
MOV #BIT7!BIT5!BIT4,@IBS ;SET ACC,TON, AND LON.  
MOV #377,@IBD ;LOAD IBD  
RESET ;ISSUE SYS INIT.  
TSTB @IBD ;DID IT CLEAR?  
BEQ TST43 ;:
```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1) 006244 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
842 ;FAILED TO CLEAR LOW BYTE OF IBD ON  
843 ;SYSTEM INIT.

:::\$\$\$\$\$\$\$\$\$\$\$^ ERROR ^\$\$\$\$\$\$\$\$\$\$\$

845  
846  
847  
848  
849  
850  
(3)  
(3)  
(2)  
851  
852  
853  
854  
855  
856  
(1)  
(1)  
(1)  
(1)  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867

```
.SBTTL  
.SBTTL INTERRUPT TESTS  
.SBTTL  
:*****  
:*TEST 43 *TEST THAT CMD CAN GENERATE AN INTERRUPT  
:*****  
TST43: SCOPE  
CLR @IBS ;CLEAR THE CSR.  
MOV #200,@PRC  
MOV #1$,@VECTC ;SET UP INTERRUPT VECTOR  
BIS #BIT0!BIT6,@IBS ;SET TCS, SHOULD CAUSE  
;/PR  
MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN  
MOV #64$,-(SP) ;/SHOW RETURN ADDRESS  
RTI ;/CAUSE A RETURN (PUTS NEW STATUS  
;/IN STATUS REG.)  
64$: NOP ;CMD TO SET AND GIVE US AN  
NOP ;INTERRUPT.  
NOP  
NOP  
NOP  
NOP  
NOP  
NOP  
NOP  
NOP  
NOP  
NOP
```

:::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$

(1) 006334 104001 ERROR 1 ;/MODULE FAULT DETECTED:  
868 ;CMD FAILED TO GENERATE AN INTERRUPT.

:::\$\$\$\$\$\$\$\$\$\$\$^ ERROR ^\$\$\$\$\$\$\$\$\$\$\$

```
BR 2$  
1$: ADD #4,SP ;/ADD #4 TO STACK POINTER.  
2$: CLR @IBS ;CLEAR INTERRUPT  
;/-RESV-  
(1) 006350 013777 001422 173020 MOV PRC,@VECTC ;/RESTORE VECTOR FOR  
(1) 006356 012777 004700 173036 MOV #4700,@PRC ;/ILLEGAL INTRO.
```

874  
875  
(3)  
(3)

:\*\*\*\*\*  
:\*TEST 44 \*TEST THAT TCR AND LNR CAN GENERATE INTERRUPTS  
:\*\*\*\*\*



```

(2) 006364 000004 TST44: SCOPE
876 006366 012777 000200 173026 MOV #200,@PRC
877 006374 012777 006500 172774 MOV #1$,@VECTC ;SET UP INTERRUPT VECTOR FOR TKR INTERRUPT
878 006402 012777 000060 172752 MOV #BIT4!BIT5,@IBS ;SET TON AND LON
879 006410 052777 000100 172744 BIS #BIT6,@IBS ;ALLOW INTERRUPT
880 ;/PR
(1) 006416 012746 000000 MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
(1) 006422 012746 006430 MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
(1) 006426 000002 RTI ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006430 64$: ;/IN STATUS REG.)
881 006430 000240 NOP
882 006432 000240 NOP
883 006434 000240 NOP
884 006436 000240 NOP
885 006440 000240 NOP
886 006442 000240 NOP
887 006444 000240 NOP
888 006446 000240 NOP
889 006450 000240 NOP
890 006452 000240 NOP
891

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

```

(1)
(1) 006454 104001 ERROR 1 ;/MODULE FAULT DETECTED:
892 ;FAILED TO GENERATE A TKR INTERRUPT.

```

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

```

894 006456 005077 172700 CLR @IBS ;CLR CSR
895 ;/-RESV-
(1) 006462 013777 001422 172706 MOV PRC,@VECTC ;/RESTORE VECTOR FOR
(1) 006470 012777 004700 172724 MOV #4700,@PRC ;/ILLEGAL INTRO.
896 006476 000453 BR TST45 ;
897
898 006500 1$: ADD #4,SP ;/ADD #4 TO STACK POINTER.
(1) 006500 062706 000004 ;/-RESV-
(1) 006504 013777 001422 172664 MOV PRC,@VECTC ;/RESTORE VECTOR FOR
(1) 006512 012777 004700 172702 MOV #4700,@PRC ;/ILLEGAL INTRO.
900 006520 012777 000200 172676 MOV #200,@PRD
901 006526 012777 006602 172644 MOV #2$,@VECTD ;SET UP FOR LNR INTERRUPT.
902 006534 105277 172624 INCB @IBD ;SEND DATA - CLRS TKR SETS LNR
903 ;FOR INTERRUPT
904 ;/PR
(1) 006540 012746 000000 MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
(1) 006544 012746 006552 MOV #65$,-(SP) ;/SHOW RETURN ADDRESS
(1) 006550 000002 RTI ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006552 65$: ;/IN STATUS REG.)
905 006552 000240 NOP
906 006554 000240 NOP
907 006556 000240 NOP
908 006560 000240 NOP
909 006562 000240 NOP
910 006564 000240 NOP
911 006566 000240 NOP

```

912 006570 000240  
913 006572 000240  
914 006574 000240  
915

NOP  
NOP  
NOP

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 006576 104001  
916

ERROR 1 ;/MODULE FAULT DETECTED:  
;/FAILED TO GENERATE LNR INTERRUPT

918 006600 000402  
919

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$  
BR 3\$

920 006602 2\$:  
(1) 006602 062706 000004  
921 006606 005077 172550 3\$:

ADD #4,SP ;/ADD #4 TO STACK POINTER.  
CLR @IBS ;/CLEAR THE STATUS REG.

922  
(1) 006612 013777 001424 172560  
(1) 006620 012777 004700 172576  
923

;/-RESV-  
MOV PRD,@VECTD ;/RESTORE VECTOR FOR  
MOV #4700,@PRD ;/ILLEGAL INTRO.

```

925          ::*****
(3)          ::*TEST 45      *TEST THAT ER2 CAN GENERATE AN INTERRUPT
(3)          ::*****
(2) 006626 000004          TST45: SCOPE
926
927 006630 005077 172526          CLR      @IBS          ;START WITH CSR CLEAR
928 006634 012777 000200 172554          MOV      #200,@PRA
929 006642 012777 006734 172522          MOV      #1$,@VECTA          ;SET UP INTERRUPT VECTOR
930          ;/PR
(1) 006650 012746 000200          MOV      #200,-(SP)          ;/SET CPU PRIORITY ON RETURN
(1) 006654 012746 006662          MOV      #64$,-(SP)          ;/SHOW RETURN ADDRESS
(1) 006660 000002          RTI          ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006662          ;/IN STATUS REG.)
931 006662 052777 000140 172472          BIS      #BITS!BIT6,@IBS          ;SET TON - NO LISTNERS ON
932 006670 105077 172470          CLR      @IBD          ;BUS BUT DATA PUT ON
933 006674 000240          NOP          ;BUS - THEREFORE AN INTERRUPT
934 006676 000240          NOP          ;SHOULD BE POSTED.
935          ;/PR
(1) 006700 012746 000000          MOV      #0,-(SP)          ;/SET CPU PRIORITY ON RETURN
(1) 006704 012746 006712          MOV      #65$,-(SP)          ;/SHOW RETURN ADDRESS
(1) 006710 000002          RTI          ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006712          ;/IN STATUS REG.)
936 006712 000240          NOP
937 006714 000240          NOP
938 006716 000240          NOP
939 006720 000240          NOP
940 006722 000240          NOP
941 006724 000240          NOP
942 006726 000240          NOP
943
944

```

::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$\$

```

(1)
(1) 006730 104001          ERROR 1          ;/MODULE FAULT DETECTED:
945          ;/FAILED TO INTERRUPT ON ERROR2

```

::\$\$\$\$\$\$\$\$\$\$\$\$^ ERROR ^\$\$\$\$\$\$\$\$\$\$\$\$

```

947 006732 000402          BR      2$
948 006734          1$:
(1) 006734 062706 000004          ADD      #4,SP          ;/ADD #4 TO STACK POINTER.
949 006740 005077 172416          2$: CLR      @IBS          ;/CLEAR CSR
950          ;/RESV-
(1) 006744 013777 001416 172420          MOV      PRA,@VECTA ;/RESTORE VECTOR FOR
(1) 006752 012777 004700 172436          MOV      #4700,@PRA          ;/ILLEGAL INTRO.
951

```

```

953          ::*****
(3)          ::*TEST 46      *TEST THAT SRQ CAN GENERATE AN INTERRUPT (SWR10=1)
(3)          ::*****
(2) 006760 000004          TST46: SCOPE
954 006762 005077 172374  CLR      @IBS          :START WITH CSR CLEAR
955 006766 032777 002000 172144  BIT      #BIT10,@SWR      :TEST IF 'ER1 INH' SWITCH IS SET
956 006774 001450          BEQ      TST47          :;BR IF OPERATOR SAID YES
957          :;/PR
(1) 006776 012746 000200          MOV      #200,-(SP)      :;/SET CPU PRIORITY ON RETURN
(1) 007002 012746 007010          MOV      #64$,-(SP)     :;/SHOW RETURN ADDRESS
(1) 007006 000002          RTI          :;/CAUSE A RETURN (PUTS NEW STATUS
(1) 007010          64$:          :;/IN STATUS REG.)
958 007010 012777 000200 172402  MOV      #200,@PRB
959 007016 012777 007072 172350  MOV      #1$,@VECTB     :SET UP SRQ INTERRUPT VECTOR
960 007024 052777 100100 172330  BIS      #BIT6!BIT15,@IBS :SET SRQ AND IE
961 007032 000240          NGP
962 007034 000240          NOP
963 007036 000240          NOP          :AN INTERRUPT SHOULD BE POSTED
964          :;/PR
(1) 007040 012746 000000          MOV      #0,-(SP)
(1) 007044 012746 007052          MOV      #65$,-(SP)
(1) 007050 000002          RTI          :;/SET CPU PRIORITY ON RETURN
(1) 007052          65$:          :;/SHOW RETURN ADDRESS
965 007052 000240          NOP          :;/CAUSE A RETURN (PUTS NEW STATUS
966 007054 000240          NOP          :;/IN STATUS REG.)
967 007056 000240          NOP
968 007060 000240          NOP
969 007062 000240          NOP
970 007064 000240          NOP
971

```

:::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$\$

```

(1)
(1) 007066 104001          ERROR 1          :/MODULE FAULT DETECTED:
972          :/FAILED TO INTERRUPT ON SRQ
973          :/IS SWR10 CORRECT ?

```

:::\$\$\$\$\$\$\$\$\$\$\$\$^ ERROR ^\$\$\$\$\$\$\$\$\$\$\$\$

```

975 007070 000402          BR      2$
976 007072          1$:          ADD      #4,SP          :/ADD #4 TO STACK POINTER.
(1) 007072 062706 000004          CLR      @IBS          :/CLEAR CSR
977 007076 005077 172260          2$:          :;/-RESV-
978          MOV      PRB,@VECTB :/RESTORE VECTOR FOR
(1) 007102 013777 001420 172264  MOV      #4700,@PRB     :/ILLEGAL INTRO.
(1) 007110 012777 004700 172302
979
980          .SBTTL
981          .SBTTL SECOND MODULE TESTS
982          .SBTTL

```

```

984
(3)
(3)
(2) 007116 000004
997
998
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
999 007120 005737 001250
1000 007124 001002
1001 007126 000137 007756
1002 007132 005077 172224
1003 007136 005077 172240
1004 007142 012777 000200 172262
1005 007150 012777 007206 172234
1006
(1) 007156 012746 000000
(1) 007162 012746 007170
(1) 007166 000002
(1) 007170
1007 007170 012777 000140 172204
1008 007176 000240
1009 007200 000240
1010

```

```

:*****
:*TEST 47 *TEST THAT MODULE PASSES 'BIAKI'
:*****
TST47: SCOPE

```

```

:*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A
:*SECOUND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET (CLOSED/ON).
:*ADDRESS OF THE SECOUND MODULE IS IN LOCATION 'IBS2' VECTOR
:*ADDRESS IS IN LOCATION 'VECTA2'.
:* THE SECOUND IBV-11 SHOULD BE ELECTRICALLY SECOUND ON Q BUSS.
:*TO INHIBIT THE USE OF TESTING WITH A SECOUND MODULE, MAKE
:*LOCATION 'SCDW1' ZERO.
:*THE SWITCH 'ER1 INH' SHOULD BE CLEARED (OPEN/OFF) FOR THE
:*MODULE UNDER TEST

```

```

TST SCDW1 ;TESTING WITH
BNE 3$ ;SECOUND IBV11?
JMP EOP ;NO-END PASS.
3$: CLR @IBS ;CLEAR CSR.
CLR @IBS2 ;CLEAR SECOUND MODULE.
MOV #200,@PRC2
MOV #1$,@VECTC2 ;SET UP VECTOR ADDR.
; /PR
MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
RTI ;/CAUSE A RETURN (PUTS NEW STATUS
; /IN STATUS REG.)
64$: MOV #BIT6!BIT5,@IBS2 ;SET INTR ENABLE AND TON ON SECOUND
NOP ;IBV - SHOULD CAUSE A TKR INTERRUPT.
NOP

```

::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$\$

```

(1)
(1) 007202 104001
1011
1012
1013

```

```

ERROR 1 ;/MODULE FAULT DETECTED:
;ASSUMING SECOUND MODULE IS GOOD,
;MODULE (IBV-11) UNDER TEST FAILED
;TO PASS Q BUSS SIGNAL 'BTAKI'

```

::\$\$\$\$\$\$\$\$\$\$\$\$^ ERROR ^\$\$\$\$\$\$\$\$\$\$\$\$

```

1015 007204 000402
1016 007206
(1) 007206 062706 000004
1017 007212 005077 172164
1018
(1) 007216 013777 001432 172166
(1) 007224 012777 004700 172200
1019

```

```

BR 2$
1$: ADD #4,SP ;/ADD #4 TO STACK POINTER.
2$: CLR @IBS2 ;CLEAR SECOUND MODULE
; /-RESV-
MOV PRC2,@VECTC2 ;/RESTORE VECTOR FOR
MOV #4700,@PRC2 ;/ILLEGAL INTRO.

```

```

1021
(3)
(3)
(2) 007232 000004
1022
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
1023
1024 007234 005077 172122
1025 007240 005077 172136
1026
1027 007244 012777 000200 172146
1028 007252 012777 007316 172114
1029
(1) 007260 012746 000000
(1) 007264 012746 007272
(1) 007270 000002
(1) 007272
1030 007272 012777 000100 172062
1031 007300 052777 100000 172074
1032
1033
1034 007306 000240
1035 007310 000240
1036

(1)
(1) 007312 104001
1037
1038
1039

1041 007314 000402
1042
1043 007316
(1) 007316 062706 000004
1044
1045 007322
(1)
(1) 007322 013777 001420 172044
(1) 007330 012777 004700 172062
1046 007336 005077 172020
1047 007342 005077 172034
1048

```

```

*****
*TEST 50 *TEST THAT SRQ CAN GENERATE AN INTERRUPT
*****
TST50: SCOPE
: *WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A
: *SECOND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET (CLOSED/ON).
: *ADDRESS OF THE SECOND MODULE IS IN LOCATION 'IBS2' VECTOR
: *ADDRESS IS IN LOCATION 'VECTA2'.
: * THE SECOND IBV-11 SHOULD BE ELECTRICALLY SECOND ON Q BUSS.
: *TO INHIBIT THE USE OF TESTING WITH A SECOND MODULE, MAKE
: *LOCATION 'SCDW1' ZERO.
: *THE SWITCH 'ER1 INH' SHOULD BE CLEARED (OPEN/OFF) FOR THE
: *MODULE UNDER TEST
: *
CLR @IBS ;CLEAR CSRS.
CLR @IBS2
MOV #200,@PRB ;SET UP INTERRUPT VECTOR.
MOV #1$,@VECTB
;/PR
MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
RTI ;/CAUSE A RETURN (PUTS NEW STATUS
;/IN STATUS REG.)
MOV #100,@IBS ;ENABLE INTERRUPTS
BIS #BIT15,@IBS2 ;SETTING SRQ IN THE 'CDW' MODULE
;WILL PUT SRQ ON THE IB BUS
;IS ER1 INH SW IS SET.

NOP
NOP

::$$$$$$$$$>>> ERROR <<<$$$$$$$$$

ERROR 1 ;/MODULE FAULT DETECTED:
;SRQ FAILED TO GENERATE AN INTERRUPT
;THIS ERROR OCCURS IS THE CABLE BETWEEN UNITS IS MISSING
; OR IF S1-8 ON THE KGM IS CLEARED (OPEN-OFF)

::$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
BR 2$

1$: ADD #4,SP ;/ADD #4 TO STACK POINTER.

2$: ;/-RESV-
MOV PRB,@VECTB ;/RESTORE VECTOR FOR
MOV #4700,@PRB ;/ILLEGAL INTRO.
CLR @IBS ;CLEAR CSRS
CLR @IBS2

```

1050  
(3)  
(3)  
(2) 007346 000004

\*\*\*\*\*  
: \*TEST 51 \*TEST THAT ERROR 1 IS GENERATED IF ATN IS ON THE IB BUS  
\*\*\*\*\*

1051  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)

TST51: SCOPE  
: \*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A  
: \*SECOND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET (CLOSED/ON).  
: \*ADDRESS OF THE SECOND MODULE IS IN LOCATION 'IBS2' VECTOR  
: \*ADDRESS IS IN LOCATION 'VECTA2'.  
: \* THE SECOND IBV-11 SHOULD BE ELECTRICALLY SECONDED ON Q BUSS.  
: \*TO INHIBIT THE USE OF TESTING WITH A SECOND MODULE, MAKE  
: \*LOCATION 'SCDW1' ZERO.  
: \*THE SWITCH 'ER1 INH' SHOULD BE CLEARED (OPEN/OFF) FOR THE  
: \*MODULE UNDER TEST

1052 007350 005077 172026  
1053 007354 032777 002000 171556  
1054 007362 001407  
1055 007364 005737 001176  
1056 007370 001002  
1057 007372 104401 014303  
1058 007376 000137 007756  
1059 007402 005277 171774  
1060  
1061  
1062  
1063 007406 032777 020000 171746  
1064 007414 001001  
1065

: \*  
CLR @IBS2 ;CLR CSR OF 2ND MODULE.  
BIT #BIT10,@SWR ;TEST IF 'ER1 INH' SWITCH IS SET (CLOSED/ON)  
BEQ 1\$ ;BR IF 'ER1 INH' SWITCH IS CLEARED  
TST \$PASS ;TEST IF FIRST PASS  
BNE 2\$ ;BR IF NOT  
TYPE ,WARN1 ;INFORM OPERATOR ABOUT MISSING SEVERAL TESTS  
JMP EOP ;BYPASS REMAINDER OF TEST  
INC @IBS2 ;ASSERT ATN ON IB BUS  
;ASSERTED ATN ON IBV UNDER TEST-  
;THIS SHOULD CAUSE AN ERROR 1  
;BECAUSE THE 2ND IBV HAS ATN SET.  
;DID ERROR 1 SET?  
:  
BIT #BIT13,@IBS  
BNE TST52  
::

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 007416 104001  
1066

ERROR 1 ;/MODULE FAULT DETECTED:  
;FAILED TO GENERATE ERROR 1

1068  
 (3)  
 (3)  
 (2) 007420 000004  
 (1) 007422 012737 000005 001160  
 1069  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 1070 007430 005077 171726  
 1071 007434 012777 000010 171740  
 1072 007442 032777 020000 171712  
 1073  
 1074 007450 001010  
 1075 007452 012777 000010 171722  
 1076 007460 032777 020000 171674  
 1077 007466 001001  
 1078

```

:*****
:*TEST 52 *TEST THAT ERROR 1 IS GENERATED IF IFC IS ON IB BUS BY SECOUND MODULE
:*****
TST52: SCOPE
MOV #5,$TIMES ;;DO 5 ITERATIONS
:*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A
:*SECOUND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET (CLOSED/ON).
:*ADDRESS OF THE SECOUND MODULE IS IN LOCATION 'IBS2' VECTOR
:*ADDRESS IS IN LOCATION 'VECTA2'.
:* THE SECOUND IBV-11 SHOULD BE ELECTRICALLY SECOUND ON Q BUSS.
:*TO INHIBIT THE USE OF TESTING WITH A SECOUND MODULE, MAKE
:*LOCATION 'SCDW1' ZERO.
:*THE SWITCH 'ER1 INH' SHOULD BE CLEARED (OPEN/OFF) FOR THE
:*MODULE UNDER TEST
:*
CLR @IBS ;CLEAR CSR
MOV #BIT3,@IBS2 ;ASSERT IFC FROM TESTOR
BIT #BIT13,@IBS ;DID ERROR 1 GET SET?
;IF SO - NEXT TEST
;
BNE TST53 ;
MOV #BIT3,@IBS2 ;IF NOT WE'LL TRY AGAIN SENCE MEMORY
BIT #BIT13,@IBS ;REFRESH COULD HAVE GO IN THE WAY.
BNE TST53 ;

```

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
 (1) 007470 104001  
 1079  
 1080  
 1081

```

ERROR 1 ;/MODULE FAULT DETECTED:
;ERROR 1 FAILED TO SET WHEN
;IFC WAS ON IB-BUS AND MODULE
;UNDER TEST DIDN'T PUT IT THERE.

```



1083  
(3)  
(3)  
(2) 007472 000004

\*\*\*\*\*  
\*TEST 53 \*TEST THAT ERROR 1 IS GENERATED IF REN IS ON IB BUS  
\*\*\*\*\*  
TST53: SCOPE

1084  
1085  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)

;\*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A  
;\*SECOND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET (CLOSED/ON).  
;\*ADDRESS OF THE SECOND MODULE IS IN LOCATION 'IBS2' VECTOR  
;\*ADDRESS IS IN LOCATION 'VECTA2'.  
;\* THE SECOND IBV-11 SHOULD BE ELECTRICALLY SECONDED ON Q BUSS.  
;\*TO INHIBIT THE USE OF TESTING WITH A SECOND MODULE, MAKE  
;\*LOCATION 'SCDW1' ZERO.  
;\*THE SWITCH 'ER1 INH' SHOULD BE CLEARED (OPEN/OFF) FOR THE  
;\*MODULE UNDER TEST  
;\*

1086  
1087 007474 005077 171662  
1088 007500 005077 171676  
1089 007504 052777 000004 171670  
1090  
1091 007512 032777 020000 171642  
1092 007520 001001  
1093

CLR @IBS ;CLEAR CSRS.  
CLR @IBS2  
BIS #BIT2,@IBS2 ;ASSERT REN ON IB BUS FROM 2ND  
;MODULE. 1ST IBV-11 SHOULD  
BIT #BIT13,@IBS ;GENERATE AN ERROR 1;DID IT??  
BNE TST54 ;;

::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 007522 104001  
1094

ERROR 1 ;/MODULE FAULT DETECTED:  
;FAILED TO GENERATE AN ERROR 1.

::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$

1096

```

1098
(3)
(3)
(2) 007524 000004
1099
1100
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
1101
1102 007526 005077 171650
1103 007532 005077 171624
1104
1105 007536 012777 000200 171652
1106 007544 012777 007610 171620
1107
1108 007552 052777 000100 171602
1109
(1) 007560 012746 000000
(1) 007564 012746 007572
(1) 007570 000002
(1) 007572
1110 007572 052777 000004 171602
1111 007600 000240
1112 007602 000240
1113

```

```

:*****
:*TEST 54 *TEST THAT AN ERROR 1 CAN GENERATE AN INTERRUPT
:*****
TST54: SCOPE

```

```

:*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A
:*SECOND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET (CLOSED/ON).
:*ADDRESS OF THE SECOND MODULE IS IN LOCATION 'IBS2' VECTOR
:*ADDRESS IS IN LOCATION 'VECTA2'.
:* THE SECOND IBV-11 SHOULD BE ELECTRICALLY SECOND ON Q BUSS.
:*TO INHIBIT THE USE OF TESTING WITH A SECOND MODULE, MAKE
:*LOCATION 'SCDW1' ZERO.
:*THE SWITCH 'ER1 INH' SHOULD BE CLEARED (OPEN/OFF) FOR THE
:*MODULE UNDER TEST
:*

```

```

CLR @IBS2 ;CLEAR CSRS.
CLR @IBS
MOV #200,@PRA
MOV #1$,@VECTA ;SET UP VECTOR ADDR.
BIS #BIT06,@IBS ;SET INTERRUPT ENABLE
; /PR
MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
RTI ;/CAUSE A RETURN (PUTS NEW STATUS
; /IN STATUS REG.)
BIS #BIT2,@IBS2 ;GENERATE AN ERROR 1 AS PER LAST TEST.
NOP
NOP

```

::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$\$

```

(1)
(1) 007604 104001
1114 ERROR 1 ;/MODULE FAULT DETECTED:
;ERROR 1 FAILED TO GENERATE AN INTR.

```

::\$\$\$\$\$\$\$\$\$\$\$\$^ ERROR ^\$\$\$\$\$\$\$\$\$\$\$\$

```

1116 007606 000402 BR 2$
1117
1118 007610 1$:
(1) 007610 062706 000004 ADD #4,SP ;/ADD #4 TO STACK POINTER.
1119 007614 2$:
(1)
(1) 007614 013777 001416 171550 MOV PRA,@VECTA ;/-RESV-
; /RESTORE VECTOR FOR
(1) 007622 012777 004700 171566 MOV #4700,@PRA ;/ILLEGAL INTRO.
1120 007630 005077 171546 CLR @IBS2 ;CLEAR CSRS.
1121 007634 005077 171522 CLR @IBS
;122
1133

```

1135  
(3)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(3)

```
*****
: *TEST 55      *TEST THAT DATA CAN BE XFERRED BETWEEN THE MODULE UNDER TEST AND THE KGM
: *           NOTE: KGM =KNOWN GOOD MODULE
: *IN THIS TEST WE'LL MAKE THE KGM A LISTENER
: * AND THE MODULE UNDER TEST A TALKER.
: *WE'VE ALREADY XFERRED DATA TO AND FROM THE IB-BUS
: *VIA THE MODULE UNDER TEST. THE ONLY UNKNOWN
: *IS THE CABLE CONNECTING THE KGM TO THE MODULE UNDER TEST,
: *AS WELL AS THE KGM.
: *
```

(2) 007640 000004  
1136  
1137 007642 012737 007662 001110  
1138 007650 012737 000000 001124  
1139 007656 005037 001126  
1140  
1141 007662 005077 171474  
1142 007666 005077 171510  
1143 007672 052777 000041 171462  
1144 007700 052777 000020 171474  
1145 007706 013777 001124 171450  
1146 007714 117737 171464 001126  
1147 007722 123737 001124 001126  
1148 007730 001402  
1149

```
*****
TST55: SCOPE
MOV #1$,SLPERR      ;SET ERROR LOOP.
MOV #0,$GDDAT       ;START PATTERN.
CLR $BDDAT
1$: CLR @IBS         ;CLEAR CSRS.
   CLR @IBS2
   BIS #BIT5!BIT0,@IBS ;SET TON AND TCS.
   BIS #BIT4,@IBS2    ;SET LON ON KGM.
   MOV $GDDAT,@IBD    ;SEND PATTERN.
   MOV @IBD2,$BDDAT   ;READ ATA FROM KGM.
   CMPB $GDDAT,$BDDAT ;DATA SENT = DATA RECEIVED?
   BEQ 2$             ;YES, CONTINUE
```

:::\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$

(1)  
(1) 007732 104004  
1150  
1151

```
ERROR 4 ;/MODULE FAULT DETECTED:
;ERROR - BAD DATA PASSED BETWEEN
;MODULE UNDER TEST AND KGM.
```

1153 007734 000407  
1154  
1155 007736 105237 001124  
1156 007742 001347  
1157  
1158 007744 005077 171412  
1159 007750 005077 171426  
1160

```
*****
BR TST56 ;:
2$: INCB $GDDAT ;CHANGE PATTERN.
   BNE 1$ ;IF NOT DONE, CONTINUE.
   CLR @IBS ;CLEAR CSR'S
   CLR @IBS2
```

1162  
(3)  
(3)  
(2) 007754 000004  
1163 007756  
1164  
1165  
(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1) 007756  
(1) 007756 000004  
(1) 007760 005037 001102  
(1) 007764 005037 001160  
(1) 007770 005237 001176  
(1) 007774 042737 100000 001176  
(1) 010002 005327  
(1) 010004 000001  
(1) 010006 003022  
(1) 010010 012737  
(1) 010012 000001  
(1) 010014 010004  
(1) 010016 104401 010063  
(2) 010022 013746 001176  
(2) 010026 104405  
(1) 010030 104401 010060  
(1) 010034 013700 000042  
(1) 010040 001405  
(1) 010042 000005  
(1) 010044 004710  
(1) 010046 000240  
(1) 010050 000240  
(1) 010052 000240  
(1) 010054  
(1) 010054 000137  
(1) 010056 002354  
(1) 010060 377 377 000  
(1) 010063 015 042412 042116  
(1) 010070 050040 051501 020123  
(1) 010076 000043

```
::*****  
:*TEST 56 *TEMP END OF TESTS  
*****  
TST56: SCOPE  
EOP:  
 .SBTTL SYSMAC ROUTINES:  
 .SBTTL END OF PASS ROUTINE  
  
*****  
:*INCREMENT THE PASS NUMBER ($PASS)  
:*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)  
:*IF THERES A MONITOR GO TO IT  
:*IF THERE ISN'T JUMP TO RSTART  
  
$EOP: SCOPE  
 CLR $STNM ::ZERO THE TEST NUMBER  
 CLR $TIMES ::ZERO THE NUMBER OF ITERATIONS  
 INC $PASS ::INCREMENT THE PASS NUMBER  
 BIC #100000,$PASS ::DON'T ALLOW A NEG. NUMBER  
 DEC (PC)+ ::LOOP?  
$EOPCT: .WORD 1  
 BGT $DOAGN ::YES  
 MOV (PC)+,a(PC)+ ::RESTORE COUNTER  
$SENDCT: .WORD 1  
 TYPE $SENDMG ::TYPE 'END PASS #'  
 MOV $PASS,-(SP) ::SAVE $PASS FOR TYPEOUT  
 TYPDS ::GO TYPE--DECIMAL ASCII WITH SIGN  
 TYPE $ENULL ::TYPE A NULL CHARACTER  
$GET42: MOV @#42,R0 ::GET MONITOR ADDRESS  
 BEQ $DOAGN ::BRANCH IF NO MONITOR  
 RESET ::CLEAR THE WORLD  
$SENDAD: JSR PC,(R0) ::GO TO MONITOR  
 NOP ::SAVE ROOM  
 NOP ::FOR  
 NOP ::ACT11  
$DOAGN: JMP a(PC)+ ::RETURN  
$RTNAD: .WORD RSTART  
$ENULL: .BYTE -1,-1,0 ::NULL CHARACTER STRING  
$SENDMG: .ASCIZ <15><12>/END PASS #/
```

```

1167                                     :/DELMA
(1)                                     :/ROUTINE TO PROVIDE DELAYS IN INCREMENTS OF 25 US
(1)                                     :/
(1)                                     :/ CALL=          JSR      R5,DEL50
(1)                                     :/              :.WORD   X      (# OF 25 US TO DELAY)
(1)                                     :/              :RETURNS HERE
(1)                                     :/
(1)                                     :/
(1) 010100 012500                      DEL50: MOV      (R5)+,R0          :/GET # OF 25 US DELAYS
(1) 010102 013701 001436              1$:  MOV      CPUDLY,R1        :/# FOR LOOP TO DO 25 US.
(1) 010106 005301                      2$:  DEC      R1              :/DEC IT
(1) 010110 001376                      BNE     2$                  :/WAIT FOR 25 US TIME?
(1) 010112 005300                      DEC     R0                  :/DONE # OF 25 US DELAY DESIRED?
(1) 010114 001372                      BNE     1$                  :/NO - NEXT ONE.
(1) 010116 000205                      RTS     R5                  :/YES - EXIT.
    
```

```

1168 .SBTTL ERROR HANDLER ROUTINE
(1)
(2) *****
(1) *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1) *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) *AND GO TO $ERRTYP ON ERROR
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) *SW15=1      HALT ON ERROR
(1) *SW13=1      INHIBIT ERROR TYPEOUTS
(1) *SW09=1      LOOP ON ERROR
(1) *CALL
(1) *          ERROR  N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1) 010120                      $ERROR:
(1) 010120 104407                      7$:  CKSWR                      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 010122 105237 001103              INCB   $ERFLG                 ;;SET THE ERROR FLAG
(1) 010126 001775                      BEQ    7$                     ;;DON'T LET THE FLAG GO TO ZERO
(1) 010130 013777 001102 171004      MOV    $TSTNM,@DISPLAY       ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 010136 005237 001112              INC    $ERTTL                 ;;INC THE ERROR COUNT
(1) 010142 011637 001116              MOV    (SP),$ERRPC           ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 010146 162737 000002 001116      SUB    #2,$ERRPC
(1) 010154 117737 170736 001114      MOVB  @ERRPC,$ITEMB         ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 010162 032777 020000 170750      BIT   #BIT13,@SWR           ;;SKIP TYPEOUT IF SET
(1) 010170 001004                      BNE   20$                     ;;SKIP TYPEOUTS
(1) 010172 004737 010272              JSR   PC,$ERRTYP            ;;GO TO USER ERROR ROUTINE
(1) 010176 104401 001165              TYPE  ,SCLF
(1) 010202                      20$:
(1) 010202 122737 000001 001210      CMPB  #APTENV,$ENV          ;;RUNNING IN APT MODE
(1) 010210 001007                      BNE   2$                     ;;NO,SKIP APT ERROR REPORT
(1) 010212 113737 001114 010224      MOVB  $ITEMB,21$           ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 010220 004737 012520              JSR   PC,$ATY4              ;;REPORT FATAL ERROR TO APT
(1) 010224 000                      21$:  .BYTE  0
(1) 010225 000                      .BYTE  0
(1) 010226 000777                      22$:  BR     22$                   ;;APT ERROR LOOP
(1) 010230 005777 170704              2$:  TST   @SWR
(1) 010234 100002                      BPL   3$                     ;;SKIP IF CONTINUE
(1) 010236 000000                      HALT
(1) 010240 104407                      CKSWR
(1) 010242 032777 001000 170670      3$:  BIT   #BIT09,@SWR          ;;TEST FOR CHANGE IN SOFT-SWR
    
```

ERROR HANDLER ROUTINE

(1) 010250 001402  
 (1) 010252 013716 001110  
 (1) 010256 005737 001162  
 (1) 010262 001402  
 (1) 010264 013716 001162  
 (1) 010270  
 (1) 010270 000002  
 1169  
 (1)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1) 010272  
 (1) 010272 104401 001165  
 (1) 010276 010046  
 (1) 010300 005000  
 (1) 010302 153700 001114  
 (1) 010306 001004  
 (1)  
 (2) 010310 013746 001116  
 (2)  
 (2) 010314 104402  
 (1) 010316 000426  
 (1) 010320 005300  
 (1) 010322 006300  
 (1) 010324 006300  
 (1) 010326 006300  
 (1) 010330 062700 001252  
 (1) 010334 012037 010344  
 (1) 010340 001404  
 (1) 010342 104401  
 (1) 010344 000000  
 (1) 010346 104401 001165  
 (1) 010352 012037 010362  
 (1) 010356 001404  
 (1) 010360 104401  
 (1) 010362 000000  
 (1) 010364 104401 001165  
 (1) 010370 011000  
 (1) 010372 001004  
 (1) 010374 012600  
 (1) 010376 104401 001165  
 (1) 010402 000207  
 (1) 010404  
 (2) 010404 013046  
 (2) 010406 104402  
 (1) 010410 005710  
 (1) 010412 001770  
 (1) 010414 104401 010422  
 (1) 010420 000771  
 (1) 010422 020040 000  
 (1)  
 1170  
 (1)

```

    BEQ      4$          ;;BR IF NO
    MOV      $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4$:  TST      $ESCAPE   ;;CHECK FOR AN ESCAPE ADDRESS
    BEQ      5$          ;;BR IF NONE
    MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
5$:
    RTI                      ;;RETURN
.SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

*****
*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
*****
$ERRTYP:
    TYPE    ,SCRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
    MOV     RO,-(SP)    ;;SAVE RO
    CLR     RO          ;;PICKUP THE ITEM INDEX
    BISB   @#$ITEMB,RO
    BNE    1$          ;;IF ITEM NUMBER IS ZERO, JUST
                        ;;TYPE THE PC OF THE ERROR
    MOV     $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
                        ;;ERROR ADDRESS
    TYP0C  ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1$:  BR      6$          ;;GET OUT
    DEC     RO          ;;ADJUST THE INDEX SO THAT IT WILL
    ASL     RO          ;;WORK FOR THE ERROR TABLE
    ASL     RO
    ASL     RO
    ADD     #$ERRTB,RO  ;;FORM TABLE POINTER
    MOV     (RO)+,2$    ;;PICKUP "ERROR MESSAGE" POINTER
    BEQ     3$          ;;SKIP TYPEOUT IF NO POINTER
    TYPE   ;;TYPE THE "ERROR MESSAGE"
    WORD   0           ;;"ERROR MESSAGE" POINTER GOES HERE
2$:  TYPE    ,SCRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
    MOV     (RO)+,4$    ;;PICKUP "DATA HEADER" POINTER
    BEQ     5$          ;;SKIP TYPEOUT IF 0
    TYPE   ;;TYPE THE "DATA HEADER"
    WORD   0           ;;"DATA HEADER" POINTER GOES HERE
3$:  TYPE    ,SCRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
    MOV     (RO),RO     ;;PICKUP "DATA TABLE" POINTER
    BNE    7$          ;;GO TYPE THE DATA
4$:  .WORD   0
    TYPE   ;;"CARRIAGE RETURN" & "LINE FEED"
5$:  MOV     (RO),RO
    BNE    7$          ;;GO TYPE THE DATA
6$:  MOV     (SP)+,RO   ;;RESTORE RO
    TYPE   ,SCRLF      ;;"CARRIAGE RETURN" & "LINE FEED"
    RTS    PC          ;;RETURN
7$:  MOV     @(RO)+,-(SP) ;;SAVE @(RO)+ FOR TYPEOUT
    TYP0C  ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
    TST     (RO)       ;;IS THERE ANOTHER NUMBER?
    BEQ     6$          ;;BR IF NO
    TYPE   ,8$         ;;TYPE TWO(2) SPACES
    BR      7$         ;;LOOP
8$:  .ASCIZ  / /       ;;TWO(2) SPACES
    .EVEN
.SBTTL  SCOPE HANDLER ROUTINE
    
```

```

(2)          ::*****
(1)          ::*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)          ::*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)          ::*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)          ::*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)          ::*SW14=1      LOOP ON TEST
(1)          ::*SW11=1      INHIBIT ITERATIONS
(1)          ::*SW09=1      LOOP ON ERROR
(1)          ::*SW08=1      LOOP ON TEST IN SWR<7:0>
(1)          ::*CALL
(1)          ::*          SCOPE          ;;SCOPE=IOT
(1)          $SCOPE:
(1) 010426          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(1) 010426 104407          CKSWR
(2) 010430 104407          BIT          #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
(1) 010432 032777 040000 170500 1$:      SOVER          ;;YES IF SW14=1
(1) 010440 001114          BNE          TESTER#####
(1)          :#####START OF CODE FOR THE XOR
(1) 010442 000416          $XTSTR: BR          6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)          :THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1) 010444 013746 000004          MOV          @WERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 010450 012737 010470 000004          MOV          #5$,@WERRVEC          ;;SET FOR TIMEOUT
(1) 010456 005737 177060          TST          @#177060          ;;TIME OUT ON XOR?
(1) 010462 012637 000004          MOV          (SP)+,@WERRVEC          ;;RESTORE THE ERROR VECTOR
(1) 010466 000463          BR          $SVLAD          ;;GO TO THE NEXT TEST
(1) 010470 022626          5$:      CMP          (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
(1) 010472 012637 000004          MOV          (SP)+,@WERRVEC          ;;RESTORE THE ERROR VECTOR
(1) 010476 000423          BR          7$          ;;LOOP ON THE PRESENT TEST
(1) 010500          6$:      :#####END OF CODE FOR THE XOR TESTER#####
(1) 010500 032777 000400 170432          BIT          #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
(1) 010506 001404          BEQ          2$          ;;BR IF NO
(1) 010510 127737 170424 001102          CMPB          @SWR,$TSTNM          ;;ON THE RIGHT TEST? SWR<7:0>
(1) 010516 001465          BEQ          $OVER          ;;BR IF YES
(1) 010520 105737 001103          2$:      TSTB          $ERFLG          ;;HAS AN ERROR OCCURRED?
(1) 010524 001421          BEQ          3$          ;;BR IF NO
(1) 010526 123737 001115 001103          CMPB          $ERMAX,$ERFLG          ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 010534 101015          BHI          3$          ;;BR IF NO
(1) 010536 032777 001000 170374          BIT          #BIT09,@SWR          ;;LOOP ON ERROR?
(1) 010544 001404          BEQ          4$          ;;BR IF NO
(1) 010546 013737 001110 001106          7$:      MOV          $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 010554 000446          BR          $OVER
(1) 010556 105037 001103          4$:      CLRB          $ERFLG          ;;ZERO THE ERROR FLAG
(1) 010562 005037 001160          CLR          $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 010566 000415          BR          1$          ;;ESCAPE TO THE NEXT TEST
(1) 010570 032777 004000 170342          3$:      BIT          #BIT11,@SWR          ;;INHIBIT ITERATIONS?
(1) 010576 001011          BNE          1$          ;;BR IF YES
(1) 010600 005737 001176          TST          $PASS          ;;IF FIRST PASS OF PROGRAM
(1) 010604 001406          BEQ          1$          ;;          INHIBIT ITERATIONS
(1) 010606 005237 001104          INC          $ICNT          ;;INCREMENT ITERATION COUNT
(1) 010612 023737 001160 001104          CMP          $TIMES,$ICNT          ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 010620 002024          BGE          $OVER          ;;BR IF MORE ITERATION REQUIRED
(1) 010622 012737 000001 001104          1$:      MOV          #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
(1) 010630 013737 010706 001160          MOV          $MXCNT,$TIMES          ;;SET NUMBER OF ITERATIONS TO DO
(1) 010636 105237 001102          $SVLAD: INCB          $TSTNM          ;;COUNT TEST NUMBERS
(1) 010642 113737 001102 001174          MOVB          $TSTNM,$TESTN          ;;SET TEST NUMBER IN APT MAILBOX
(1) 010650 011637 001106          MOV          (SP),$LPADR          ;;SAVE SCOPE LOOP ADDRESS
    
```





```
(1) 011066 104401 001165 14$: TYPE $CRLF      ::ECHO <CR> AND <LF>
(1) 011072 123727 001135 000001 CMPB $INTAG,#1  ::RE-ENABLE TTY KBD INTERRUPTS?
(1) 011100 001003 BNE 15$        ::BRANCH IF NOT
(1) 011102 012777 000100 170034 MOV #100,@STKS ::RE-ENABLE TTY KBD INTERRUPTS
(1) 011110 000002 15$: RTI          ::RETURN
(1) 011112 004737 012360 16$: JSR PC,$TYPEC  ::ECHO CHAR
(1) 011116 021627 000060 CMP (SP),#60    ::CHAR < 0?
(1) 011122 002420 BLT 18$        ::BRANCH IF YES
(1) 011124 021627 000067 CMP (SP),#67    ::CHAR > 7?
(1) 011130 003015 BGT 18$        ::BRANCH IF YES
(1) 011132 042726 000060 BIC #60,(SP)+  ::STRIP-OFF ASCII
(1) 011136 005766 000002 TST 2(SP)      ::IS THIS THE FIRST CHAR
(1) 011142 001403 BEQ 17$        ::BRANCH IF YES
(1) 011144 006316 ASL (SP)       ::NO, SHIFT PRESENT
(1) 011146 006316 ASL (SP)       :: CHAR OVER TO MAKE
(1) 011150 006316 ASL (SP)       :: ROOM FOR NEW ONE.
(1) 011152 005266 000002 17$: INC 2(SP)      ::KEEP COUNT OF CHAR
(1) 011156 056616 177776 BIS -2(SP),(SP) ::SET IN NEW CHAR
(1) 011162 000707 BR 7$          ::GET THE NEXT ONE
(1) 011164 104401 001164 18$: TYPE $QUES     ::TYPE ?<CR><LF>
(1) 011170 000720 BR 20$          ::SIMULATE CONTROL-U
(1) .DSABL LSB
```

\*\*\*\*\*

THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

\*CALL:

```
* RDCHR          ::INPUT A SINGLE CHARACTER FROM THE TTY
* RETURN HERE   ::CHARACTER IS ON THE STACK
*              ::WITH PARITY BIT STRIPPED OFF
```

```
(1) 011172 011646 $RDCHR: MOV (SP),-(SP)  ::PUSH DOWN THE PC
(1) 011174 016666 MOV 4(SP),2(SP)  ::SAVE THE PS
(1) 011202 105777 167736 1$: TSTB @STKS    ::WAIT FOR
(1) 011206 100375 BPL 1$          ::A CHARACTER
(1) 011210 117766 167732 000004 MOVB @STKB,4(SP)  ::READ THE TTY
(1) 011216 042766 177600 000004 BIC #^C<177>,4(SP) ::GET RID OF JUNK IF ANY
(1) 011224 026627 000004 000023 CMP 4(SP),#23    ::IS IT A CONTROL-S?
(1) 011232 001013 BNE 3$          ::BRANCH IF NO
(1) 011234 105777 167704 2$: TSTB @STKS    ::WAIT FOR A CHARACTER
(1) 011240 100375 BPL 2$          ::LOOP UNTIL ITS THERE
(1) 011242 117746 167700 MOVB @STKB,-(SP)  ::GET CHARACTER
(1) 011246 042716 177600 BIC #^C177,(SP)  ::MAKE IT 7-BIT ASCII
(1) 011252 022627 000021 CMP (SP)+,#21    ::IS IT A CONTROL-Q?
(1) 011256 001366 BNE 2$          ::IF NOT DISCARD IT
(1) 011260 000750 BR 1$          ::YES, RESUME
(1) 011262 026627 000004 000021 3$: CMP 4(SP),#$XON  ::IS IT A RANDOM XON?
(1) 011270 001744 BEQ 1$          ::BRANCH IF YES
(1) 011272 026627 000004 000140 CMP 4(SP),#140  ::IS IT UPPER CASE?
(1) 011300 002407 BLT 4$          ::BRANCH IF YES
(1) 011302 026627 000004 0C0175 CMP 4(SP),#175  ::IS IT A SPECIAL CHAR?
(1) 011310 003003 BGT 4$          ::BRANCH IF YES
(1) 011312 042766 000040 000004 BIC #40,4(SP)  ::MAKE IT UPPER CASE
(1) 011320 000002 4$: RTI          ::GO BACK TO USER
(2) *****
```

```

(1)          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)          ;*CALL:
(1)          ;*      RDLIN          ;:INPUT A STRING FROM THE TTY
(1)          ;*      RETURN HERE    ;:ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)          ;*                    ;:TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)          ;
(1) 011322 010346          $RDLIN: MOV      R3,-(SP)          ;:SAVE R3
(1) 011324 012703 011430 1$:      MOV      #$TTYIN,R3        ;:GET ADDRESS
(1) 011330 022703 011440 2$:      CMP      #$TTYIN+8.,R3      ;:BUFFER FULL?
(1) 011334 101405          BLOS     4$                    ;:BR IF YES
(1) 011336 104410          RDCHR                    ;:GO READ ONE CHARACTER FROM THE TTY
(1) 011340 112613          MOVB     (SP)+,(R3)        ;:GET CHARACTER
(1) 011342 122713 000177 10$:     CMPB     #177,(R3)        ;:IS IT A RUBOUT
(1) 011346 001003          BNE     3$                    ;:SKIP IF NOT
(1) 011350 104401 001164 4$:      TYPE     ,SQUES          ;:TYPE A '?'
(1) 011354 000763          BR      1$                    ;:CLEAR THE BUFFER AND LOOP
(1) 011356 111337 011426 3$:      MOVB     (R3),9$          ;:ECHO THE CHARACTER
(1) 011362 104401 011426          TYPE     ,9$
(1) 011366 122723 000015          CMPB     #15,(R3)+        ;:CHECK FOR RETURN
(1) 011372 001356          BNE     2$                    ;:LOOP IF NOT RETURN
(1) 011374 105063 177777          CLRB    -1(R3)            ;:CLEAR RETURN (THE 15)
(1) 011400 104401 001166          TYPE     ,LF             ;:TYPE A LINE FEED
(1) 011404 012603          MOV     (SP)+,R3          ;:RESTORE R3
(1) 011406 011646          MOV     (SP),-(SP)        ;:ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 011410 016666 000004 000002  MOV     4(SP),2(SP)        ;:FIRST ASCII CHARACTER ON IT
(1) 011416 012766 011430 000004  MOV     #$TTYIN,4(SP)
(1) 011424 000002          RTI                        ;:RETURN
(1) 011426 000          9$:      .BYTE    0          ;:STORAGE FOR ASCII CHAR. TO TYPE
(1) 011427 000          .BYTE    0          ;:TERMINATOR
(1) 011430 000010          $TTYIN: .BLKB 8.         ;:RESERVE 8 BYTES FOR TTY INPUT
(1) 011440 052536 005015 000  $CNTLU: .ASCIZ /^U/<15><12> ;:CONTROL 'U'
(1) 011445 136 006507 000012  $CNTLG: .ASCIZ /^G/<15><12> ;:CONTROL 'G'
(1) 011452 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 011460 020075 000
(1) 011463 040 047040 053505 $MNEW: .ASCIZ / NEW = /
(1) 011470 036440 000040
  
```



BINARY TO OCTAL (ASCII) AND TYPE

(1)	011636	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
(1)	011640	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
(1)	011644	052703	000040	5\$:	BIS	#' ,R3	::MAKE ASCII IF NOT ALREADY
(1)	011650	110337	011714		MOVB	R3,8\$	::SAVE FOR TYPING
(1)	011654	104401	011714		TYPE	8\$	::GO TYPE THIS DIGIT
(1)	011660	105337	011716	7\$:	DECB	\$OCNT	::COUNT BY 1
(1)	011664	003347			BGT	2\$	::BR IF MORE TO DO
(1)	011666	002402			BLT	6\$	::BR IF DONE
(1)	011670	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
(1)	011672	000744			BR	2\$	::GO DO THE LAST DIGIT
(1)	011674	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
(1)	011676	012604			MOV	(SP)+,R4	::RESTORE R4
(1)	011700	012603			MOV	(SP)+,R3	::RESTORE R3
(1)	011702	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
(1)	011710	012616			MOV	(SP)+,(SP)	
(1)	011712	000002			RTI		::RETURN
(1)	011714	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
(1)	011715	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
(1)	011716	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
(1)	011717	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
(1)	011720	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE





```

(1) 012270 105037 012476 CLR# $CHARCNT ::CLEAR CHARACTER COUNT
(1) 012274 000755 BR 2$ ::GET NEXT CHARACTER
(1) 012276 004737 012360 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
(1) 012302 123726 001156 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
(1) 012306 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
(1) 012310 013746 001154 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
(1) 7$: DECB 1(SP) ::AND THE NULL CHAR.
(1) 012320 002770 BLT 6$ ::DOES A NULL NEED TO BE TYPED?
(1) 012322 004737 012360 JSR PC,$TYPEC ::BR IF NO--GO POP THE NULL OFF OF STACK
(1) 012326 105337 012476 DECB $CHARCNT ::GO TYPE A NULL
(1) 012332 000770 BR 7$ ::DO NOT COUNT AS A COUNT
::LOOP

:HORIZONTAL TAB PROCESSOR
(1) 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
(1) 012340 004737 012360 9$: JSR PC,$TYPEC ::TYPE A SPACE
(1) 012344 132737 000007 012476 BITB #7,$CHARCNT ::BRANCH IF NOT AT
(1) 012352 001372 BNE 9$ ::TAB STOP
(1) 012354 005726 TST (SP)+ ::POP SPACE OFF STACK
(1) 012356 000724 BR 2$ ::GET NEXT CHARACTER

$TYPEC:
(1) 012360 105777 166560 TSTB @STKS ::CHAR IN KYBD BUFFER? :MJD001
(1) 012364 100022 BPL 10$ ::BR IF NOT :MJD001
(1) 012366 017746 166554 MOV @STKB,-(SP) ::GET CHAR :MJD001
(1) 012372 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS :MJD001
(1) 012376 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF :MJD001
(1) 012402 001012 BNE 102$ ::BR IF NOT :MJD001

101$:
(1) 012404 105777 166534 TSTB @STKS ::WAIT FOR CHAR :MJD001
(1) 012410 100375 BPL 101$ :MJD001
(1) 012412 117716 166530 MOVB @STKB,(SP) ::GET CHAR :MJD001
(1) 012416 042716 177600 BIC #177600,(SP) ::STRIP IT :MJD001
(1) 012422 122716 000021 CMPB #$XON,(SP) ::WAS IT XON? :MJD001
(1) 012426 001366 BNE 101$ ::BR IF NOT :MJD001

102$:
(1) 012430 005726 TST (SP)+ ::FIX STACK :MJD001
(1) 012432 105777 166512 10$: TSTB @STPS ::WAIT UNTIL PRINTER IS READY :MJD001
(1) 012436 100375 BPL 10$ :MJD001
(1) 012440 116677 000002 166504 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 012446 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
(1) 012454 001003 BNE 1$ ::BRANCH IF NO
(1) 012456 105037 012476 CLR# $CHARCNT ::YES--CLEAR CHARACTER COUNT
(1) 012462 000406 BR $TYPEX ::EXIT
(1) 012464 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
(1) 012472 001402 BEQ $TYPEX ::BRANCH IF YES
(1) 012474 105227 INCB (PC)+ ::COUNT THE CHARACTER
(1) 012476 000000 $CHARCNT: .WORD 0 ::CHARACTER COUNT STORAGE
(1) 012500 000207 $TYPEX: RTS PC

.SBTTL APT COMMUNICATIONS ROUTINE
*****
(1) $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
(2) $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
    
```

```

(1) 012516 000403 BR SATYC
(1) 012520 112737 000001 012746 SATY4: MOV #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) 012526 SATYC:
(3) 012526 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 012530 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(1) 012532 105737 012744 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
(1) 012536 001450 BEQ 5$ ;;IF NOT: BR
(1) 012540 122737 000001 001210 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 012546 001031 BNE 3$ ;;IF NOT: BR
(1) 012550 132737 000100 001211 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 012556 001425 BEQ 3$ ;;IF NOT: BR
(1) 012560 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
(1) 012564 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 012572 005737 001170 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) 012576 001375 BNE 1$ ;;IF NOT: WAIT
(1) 012600 010037 001204 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
(1) 012604 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
(1) 012606 001376 BNE 2$
(1) 012610 163700 001204 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 012614 006200 ASR R0 ;;GET MESSAGE LNTH IN WORDS
(1) 012616 010037 001206 MOV R0,$MSGLGIT ;;PUT LENGTH IN MAILBOX
(1) 012622 012737 000004 001170 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 012630 000413 BR 5$
(1) 012632 017637 000004 012656 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) 012640 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 012646 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 012652 004737 012146 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 012656 000000 4$: .WORD 0
(1) 012660 5$:
(1) 012660 105737 012746 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
(1) 012664 001416 BEQ 12$ ;;IF NOT: BR
(1) 012666 005737 001210 TST $ENV ;;RUNNING UNDER APT?
(1) 012672 001413 BEQ 12$ ;;IF NOT: BR
(1) 012674 005737 001170 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
(1) 012700 001375 BNE 11$ ;;IF NOT: WAIT
(1) 012702 017637 000004 001172 MOV @4(SP),$FATAL ;;GET ERROR #
(1) 012710 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 012716 005237 001170 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
(1) 012722 105037 012746 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
(1) 012726 105037 012745 CLRB $LFLG ;;CLEAR LOG FLAG
(1) 012732 105037 012744 CLRB $MFLG ;;CLEAR MESSAGE FLAG
(3) 012736 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 012740 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 012742 000207 RTS PC ;;RETURN
(1) 012744 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
(1) 012745 000 $LFLG: .BYTE 0 ;;LOG FLAG
(1) 012746 000 $FFLG: .BYTE 0 ;;FATAL FLAG
(1) 012750 .EVEN
(1) 000200 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTPOOL=100
(1) 000040 APTCSUP=040
    
```



1179  
(1)  
(2)  
(1)  
(1)  
(1)  
(3)  
(3)  
(3)  
(3)  
(3)  
(3)  
(3)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1)  
(3)  
(3)  
(3)  
(3)  
(3)  
(3)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)

012750 012737 013110 000024  
012756 012737 000340 000026  
012764 010046  
012766 010146  
012770 010246  
012772 010346  
012774 010446  
012776 010546  
013000 017746 166134  
013004 010637 013114  
013010 012737 013022 000024  
013016 000000  
013020 000776  
  
013022 012737 013110 000024  
013030 013706 013114  
013034 005037 013114  
013040 005237 013114  
013044 001375  
013046 012677 166066  
013052 012605  
013054 012604  
013056 012603  
013060 012602  
013062 012601  
013064 012600  
013066 012737 012750 000024  
013074 012737 000340 000026  
013102 104401  
013104 013116  
013106 000002  
013110 000000  
013112 000776  
013114 000000  
013116 005015 047520 042527  
013124 000122

.SBTTL POWER DOWN AND UP ROUTINES

::\*\*\*\*\*  
:POWER DOWN ROUTINE

```
$PWRDN: MOV    #$ILLUP,@#PWRVEC  ;;SET FOR FAST UP
        MOV    #340,@#PWRVEC+2  ;;PRIO:7
        MOV    R0,-(SP)          ;;PUSH R0 ON STACK
        MOV    R1,-(SP)          ;;PUSH R1 ON STACK
        MOV    R2,-(SP)          ;;PUSH R2 ON STACK
        MOV    R3,-(SP)          ;;PUSH R3 ON STACK
        MOV    R4,-(SP)          ;;PUSH R4 ON STACK
        MOV    R5,-(SP)          ;;PUSH R5 ON STACK
        MOV    @SWR,-(SP)        ;;PUSH @SWR ON STACK
        MOV    SP,$SAVR6         ;;SAVE SP
        MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
        HALT
        BR     .-2              ;;HANG UP
```

::\*\*\*\*\*  
:POWER UP ROUTINE

```
$PWRUP: MOV    #$ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
        MOV    $SAVR6,SP         ;;GET SP
        CLR    $SAVR6           ;;WAIT LOOP FOR THE TTY
1$:      INC    $SAVR6           ;;WAIT FOR THE INC
        BNE   1$                ;;OF WORD
        MOV   (SP)+,@SWR        ;;POP STACK INTO @SWR
        MOV   (SP)+,R5          ;;POP STACK INTO R5
        MOV   (SP)+,R4          ;;POP STACK INTO R4
        MOV   (SP)+,R3          ;;POP STACK INTO R3
        MOV   (SP)+,R2          ;;POP STACK INTO R2
        MOV   (SP)+,R1          ;;POP STACK INTO R1
        MOV   (SP)+,R0          ;;POP STACK INTO R0
        MOV    #$PWRDN,@#PWRVEC  ;;SET UP THE POWER DOWN VECTOR
        MOV    #340,@#PWRVEC+2  ;;PRIO:7
        TYPE   $POWER           ;;REPORT THE POWER FAILURE
        $PWRMG: .WORD $POWER    ;;POWER FAIL MESSAGE POINTER
        $ILLUP: HALT           ;;THE POWER UP SEQUENCE WAS STARTED
        BR     .-2              ;;BEFORE THE POWER DOWN WAS COMPLETE
        $SAVR6: 0               ;;PUT THE SP HERE
        $POWER: .ASCIZ <15><12>'POWER'
        .EVEN
```

```

1181
1182
1183      ;*
1184      ;*THIS ROUTINE WILL PROTECT THE PROGRAM
1185      ;*FROM INTERRUPTS.
1186      ;*
1187      ;*THE TRAP CATCHER IS SET UP FOR
1188      ;*      .WORD      +2
1189      ;*      .WORD      JSR      PC,R0
1190      ;*
1191      ;*ILLEGAL INTERRUPTS OR INTERRUPTS TO THE WRONG VECTOR
1192      ;*GOTO THE VECTOR AND PICK UP THE ".+2" AS AN ADDRESS
1193      ;*AND "4700" AS NEW STATUS.
1194      ;*THE .+2 AS A PC WILL CAUSE EXECUTION OF THE "JSR PC,R0" (AN ILLEGAL INSTR).
1195      ;*AND TRAP TO LOCATION "4". IN LOCATION 10 WE HAVE A
1196      ;*POINTER HERE. IF THIS CONDITION CAUSES A TRAP TO LOC. 4
1197      ;*WE WILL REPORT IT IN THE SAME MANNER THAT WE WOULD
1198      ;*REPORT ANY OTHER ERROR.
1199
1200      ;*IF A BUSS ERROR TRAP DID OCCUR AND CAUSE A TRAP TO 4,
1201      ;*WE WILL HALT.
1202 013126 011637 013172      IOTRD: MOV      (6),TRTO      ;GET WHERE WE CAME TO.
1203 013132 162737 000004 013172  SUB      #4,TRTO      ;FORM REAL ADDR.
1204
1205 013140 023727 013172 001000      CMP      TRTO,#1000    ;DID TRAP COME FROM LESS THAN ADDR. 1000?
1206 013146 003402
1207
1208 013150 000000      1$:      HALT
1209      ;NO!  MUST BE A BUSS ILLEGAL ADDR. TIME OUT.
1210      ; ADDRESS CONTAINED IN TRTO.
1211 013152 000776      BR      1$
1212 013154      2$:
1213
1214 013154 016637 000004 013174      MOV      4(6),TRFRO    ;GET TRAPPED FROM ADDR.
1215
1216 013162 062706 000004      ADD      #4,SP        ;/ADD #4 TO STACK POINTER.
1217
1218

```

::\$\$\$\$\$\$\$\$\$\$\$>>> ERROR <<<\$\$\$\$\$\$\$\$\$\$\$

```

(1)
(1) 013166 104007      ERROR 7      ;/MODULE FAULT DETECTED:
1219      ;ERROR! ILLEGAL INTERRUPT
1220      ;OR INTERRUPT TO WRONG
1221      ;VECTOR - IF TEST NUMBER
1222      ;IS LESS THAN 10, ITS LIKELY
1223      ;(BUT NOT EXCLUSIVELY) TO BE A
1224      ;DEVICE OTHER THAN THE IBV-11
1225      ;TO BLAME.
1226      ;IF THE INTERRUPT OCCURRED
1227      ;DURING AN INTERRUPT TEST, I'D
1228      ;SUSPECT A PROBLEM WITH THE
1229      ;IBV-11.
1230      ;IF THE ADDRESS THE INTERRUPT
1231      ;VECTOR TO IS WITHIN THE RANGE

```

1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249

:OF VECTORS ASSIGNED TO THE IBV-11,  
:THEN I'D SUSPECT THE IBV-11  
:INTERRUPTED ILLEGALLY.  
:IF THE ADDRESS THE INTERRUPT  
:VECTORED TO IS OUTSIDE OF THE  
:RANGE ASSIGNED TO THE IBV-11,  
:I'D SUSPECT THAT THE  
:IBV-11 PUT THE WRONG VECTOR ON  
:THE BUSS DURING THE INTERRUPT  
:PROCESS.  
:FOR THIS ERROR - DON'T  
:USE 'LOOP ON ERROR' OPTION.  
:ALSO EXPECT THE INTERRUPT TEST TO  
:REPORT THAT THE IBV-11 DIDN'T  
:INTERRUPT.  
:FOLLOW RECOMMENDED PROCEDURE  
:IN THE DOCUMENT (ON THIS DIAGNOSTIC)  
:FOR LOOPING ON ERROR

1251 013170 000002  
1252 013172 000000  
1253 013174 000000

:.:\$\$\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$  
RTI  
TRTO: .WORD 0 ;ADDR THAT WE INTERRUPTED TO  
TRFRO: .WORD 0 ;ADDR THAT WE INTERRUPTED FROM.



Line	Code	Code	Code	Code	Code	Description
1257						.SB1TL MESSAGES AND TABLES
1258						
1259	013256	005007	044415	051502	EM1:	.ASCIZ<7><12><15>#IBS FUNCTION ERROR#
	013264	043040	047125	052103		
	013272	047511	020116	051105		
	013300	047522	000122			
1260						
1261	013304	005007	044415	042102	EM2:	.ASCIZ<7><12><15>#IBD FUNCTION ERROR#
	013312	043040	047125	052103		
	013320	047511	020116	051105		
	013326	047522	000122			
1262						
1263	013332	005007	044415	051502	EM3:	.ASCIZ<7><12><15>#IBS DATA ERROR#
	013340	042040	052101	020101		
	013346	051105	047522	000122		
1264						
1265	013354	005007	044415	042102	EM4:	.ASCIZ<7><12><15>#IBD DATA ERROR#
	013362	042040	052101	020101		
	013370	051105	047522	000122		
1266						
1267	013376	100007	041111	027523	EM5:	.ASCIZ<7><200>#IBS/IBD ADDRESS ERROR#
	013404	041111	020104	042101		
	013412	051104	051505	020123		
	013420	051105	047522	000122		
1268	013426	100007	041111	041527	EM6:	.ASCIZ <7><200>#IBWC/IBCA DATA ERROR#
	013434	044457	041502	020101		
	013442	040504	040524	042440		
	013450	051122	051117	000		
1269						
1270	013455	007	044600	052116	EM7:	.ASCIZ <7><200>#INTERRUPT ERROR#
	013462	051105	052522	052120		
	013470	042440	051122	051117		
	013476	000				
1271						
1272	013477	007	046200	043517	EM10:	.ASCII <7><200>#LOGIC ERROR DETECTED WITH THE "IBC" SIGNAL#
	013504	041511	042440	051122		
	013512	051117	042040	052105		
	013520	041505	042524	020104		
	013526	044527	044124	052040		
	013534	042510	021040	041111		
	013542	021103	051440	043511		
	013550	040516	114			
1273	013553	200	051511	021040		.ASCII <200>#IS "\$CPUOP" OR "SWR12" CORRECT ?#
	013560	041444	052520	050117		
	013566	020042	051117	021040		
	013574	053523	030522	021062		
	013602	041440	051117	042522		
	013610	052103	037440			
1274	013614	022200	050103	047525		.ASCII <200>#\$CPUOP DEFAULTS TO PDP-11/03 CPU TYPE#
	013622	020120	042504	040506		
	013630	046125	051524	052040		
	013636	020117	042120	026520		
	013644	030461	030057	020063		
	013652	050103	020125	054524		
	013660	042520				
1275	013662	051600	051127	031061		.ASCII <200>#SWR12=0 INDICATES THE MODULE IS NOT A -YA VERSION#

	013670	030075	044440	042116	
	013676	041511	052101	051505	
	013704	052040	042510	046440	
	013712	042117	046125	020105	
	013720	051511	047040	052117	
	013726	040440	026440	040531	
	013734	053040	051105	044523	
	013742	047117			
1276	013744	051600	051127	031061	.ASCIZ <200>#SWR12=1 INDICATES THE MODULE IS A -YA VERSION#
	013752	030475	044440	042116	
	013760	041511	052101	051505	
	013766	052040	042510	046440	
	013774	042117	046125	020105	
	014002	051511	040440	026440	
	014010	040531	053040	051105	
	014016	044523	047117	000	
1277	014023	007	046200	043517	EM11: .ASCII <7><200>#LOGIC ERROR DETECTED WITH THE 'SRQ' SIGNAL#
	014030	041511	042440	051122	
	014036	051117	042040	052105	
	014044	041505	042524	020104	
	014052	044527	044124	052040	
	014060	042510	021040	051123	
	014066	021121	051440	043511	
	014074	040516	114		
1278	014077	200	051511	021040	.ASCII <200>#IS 'SWR10' ('ER1-INHIBIT <S1-8>') CORRECT ?#
	014104	053523	030522	021060	
	014112	024040	042442	030522	
	014120	044455	044116	041111	
	014126	052111	036040	030523	
	014134	034055	021076	020051	
	014142	047503	051122	041505	
	014150	020124	077		
1279	014153	200	053523	030522	.ASCII <200>#SWR10=0 INDICATES S1-8 IS CLEARED (OPEN-OFF)#
	014160	036460	020060	047111	
	014166	044504	040503	042524	
	014174	020123	030523	034055	
	014202	044440	020123	046103	
	014210	040505	042522	020104	
	014216	047450	042520	026516	
	014224	043117	024506		
1280	014230	051600	051127	030061	.ASCIZ <200>#SWR10=1 INDICATES S1-8 IS SET (CLOSED-ON)#
	014236	030475	044440	042116	
	014244	041511	052101	051505	
	014252	051440	026461	020070	
	014260	051511	051440	052105	
	014266	024040	046103	051517	
	014274	042105	047455	024516	
	014302	000			
1281	014303	200	042502	040503	WARN1: .ASCIZ <CRLF>#BECAUSE 'S1-8' IS SET, HARDWARE ERROR 1 IS NOT BEING TESTED#<CRLF
	014310	051525	020105	051442	
	014316	026461	021070	044440	
	014324	020123	042523	026124	
	014332	044040	051101	053504	
	014340	051101	020105	051105	
	014346	047522	020122	020061	
	014354	051511	047040	052117	

1282	014362	041040	044505	043516							
	014370	052040	051505	042524							
	014376	100104	000								
	014401	200	042524	052123	DH1:	.ASCIZ	<CRLF>#TEST	ERRPC	IB ADR	IBS	IBD#
	014406	020040	020040	051105							
	014414	050122	020103	020040							
	014422	041111	040440	051104							
	014430	020040	044440	051502							
	014436	020040	020040	044440							
	014444	042102	000								
1283											
1284	014447	200	042524	052123	DH3:	.ASCIZ	<CRLF>#TEST	ERRPC	GOOD	BAD#	
	014454	020040	020040	051105							
	014462	050122	020103	020040							
	014470	047507	042117	020040							
	014476	020040	040502	000104							
1285											
1286	014504	052200	051505	020124	DH5:	.ASCIZ	<CRLF>#TEST	ERRPC	IB ADDR#		
	014512	020040	042440	051122							
	014520	041520	020040	044440							
	014526	020102	042101	051104							
	014534	000									
1287											
1288	014535	200	042524	052123	DH7:	.ASCIZ	<CRLF>#TEST	ERRPC	TO	FROM ADDR.#	
	014542	020040	020040	051105							
	014550	050122	020103	020040							
	014556	047524	020040	020040							
	014564	020040	051106	046517							
	014572	040440	042104	027122							
	014600	000									
1289											
1290		014602					.EVEN				
1291											
1292	014602	001174	001116	001362	DT1:	.WORD	\$TESTN,\$ERRPC,IBS				
1293											
1294	014610	000000			IBSA:	.WORD	0				
1295											
1296	014612	000000	000000		IBDA:	.WORD	0,0				
1297											
1298	014616	001174	001116	001124	DT3:	.WORD	\$TESTN,\$ERRPC,\$GDDAT,\$BDDAT,0				
	014624	001126	000000								
1299											
1300	014630	001174	001116	001362	DT5:	.WORD	\$TESTN,\$ERRPC,IBS,0				
	014636	000000									
1301											
1302	014640	001174	001116	013172	DT7:	.WORD	\$TESTN,\$ERRPC,TRTO,TRFRO,0				
	014646	013174	000000								
1303											
1304	014652	000000	000000		DF0:	.WORD	0,0				
1305											
1306		000001					.END				







CROSS REFERENCE TABLE -- USER SYMBOLS

ERRVEC= 000004	46#	212*	217*	218*	283	284*	290*	304*	317	318*	324*	342*	1170*
GNS = ***** U	271	1255											
GTSWR = 104406	271	1255#											
HT = 000011	46#	1176											
IBCA 001370	168#	225*	226*	326	386								
IBD 001364	166#	221*	222*	223	230	292	361	620*	622*	624*	626*	628*	630*
	632*	634*	640*	642	655*	656	669*	670	684	685	690	745	748
	751	754	763	771	782	790	802*	837*	839	902*	932*	1145*	
IBDA 014612	230*	1296#											
IBD2 001404	174#	231*	232*	1146									
IBS 001362	165#	220*	221	229	288	350	396*	397*	399	407*	409	417*	418*
	420	428*	430	438*	439*	441	449*	451	459*	460*	462	465*	466
	475	483*	484*	486	495*	497	505*	506*	508	515*	517	527*	529*
	531	540*	541	550*	551*	553	561*	563	571*	572*	577	585*	587
	620*	622*	624*	626*	628*	630*	632*	634*	639*	654*	662	668*	677
	683*	695*	697	745*	748*	751*	754*	759*	760*	770*	777*	781*	789*
	799*	800*	804	813*	815	824*	825*	828	836*	852*	855*	872*	878*
	879*	894*	921*	927*	931*	949*	954*	960*	977*	1002*	1024*	1030*	1046*
	1063	1070*	1072	1076	1087*	1091	1103*	1108*	1121*	1141*	1143*	1158*	1292
	1300												
IBSA 014610	229*	1294#											
IBS2 001402	173#	231	1003*	1007*	1017*	1025*	1031*	1047*	1052*	1059*	1071*	1075*	1088*
	1089*	1102*	1110*	1120*	1142*	1144*	1159*						
IBWC 001366	167#	223*	224*	225	321	378							
IOTRD 013126	34	217	1202#										
IOTVEC= 000020	46#	212*	213*										
LF = 000012	46#	1176											
PIRQ = 177772	46#												
PIRQVE= 000240	46#												
PRA 001416	181#	246*	247*	248	266*	928*	950*	1105*	1119*				
PRA2 001426	185#	254*	255*	256									
PRB 001420	182#	248*	249*	250	267*	958*	978*	1027*	1045*				
PRB2 001430	186#	256*	257*	258									
PRC 001422	183#	250*	251*	252	268*	853*	873*	876*	895*	899*			
PRC2 001432	187#	258*	259*	260	1004*	1018*							
PRD 001424	184#	252*	253*	269*	900*	922*							
PRD2 001434	188#	260*	261*										
PRO = 000000	46#												
PR1 = 000040	46#												
PR2 = 000100	46#												
PR3 = 000140	46#												
PR4 = 000200	46#												
PR5 = 000240	46#												
PR6 = 000300	46#												
PR7 = 000340	46#												
PS = 177776	46#												
PSW = 177776	46#												
PWRVEC= 000024	46#	212*	216*	1179*									
RDCHR = 104410	1171	1255#											
RDLIN = 104411	1255#												
RESVEC= 000010	46#												
RSTART 002354	262#	1165											
STACK = 001100	46#	212											
START 001460	44	212#											
STKLMT= 177774	46#												
SWR 001140	60#	212*	271	574	761	955	1053	1168	1170	1171*	1179*		

SWREG	000176	39#	212	271	1171				
SW0	= 000001	46#							
SW00	= 000001	46#							
SW01	= 000002	46#							
SW02	= 000004	46#							
SW03	= 000010	46#							
SW04	= 000020	46#							
SW05	= 000040	46#							
SW06	= 000100	46#							
SW07	= 000200	46#							
SW08	= 000400	46#							
SW09	= 001000	46#							
SW1	= 000002	46#							
SW10	= 002000	46#							
SW11	= 004000	46#							
SW12	= 010000	46#							
SW13	= 020000	46#							
SW14	= 040000	46#							
SW15	= 100000	46#							
SW2	= 000004	46#							
SW3	= 000010	46#							
SW4	= 000020	46#							
SW5	= 000040	46#							
SW6	= 000100	46#							
SW7	= 000200	46#							
SW8	= 000400	46#							
SW9	= 001000	46#							
TBITVE	= 000014	46#							
TKVEC	= 000060	46#							
TPVEC	= 000064	46#							
TRAPVE	= 000034	46#	212*	215*					
TRFRO	013174	1214*	1253#	1302					
TRTO	013172	1202*	1203*	1205	1252#	1302			
TRTVEC	= 000014	46#							
TST1	002572	277#							
TST10	003304	426	431	436#					
TST11	003374	447	452	458#					
TST12	003520	471	477	481#					
TST13	003610	493	498	504#					
TST14	003700	514	518	523#					
TST15	004002	537	542	548#					
TST16	004072	559	564	570#					
TST17	004176	584	588	620#					
TST2	002716	315#							
TST20	004270	620	622#						
TST21	004362	622	624#						
TST22	004454	624	626#						
TST23	004546	626	628#						
TST24	004640	628	630#						
TST25	004732	630	632#						
TST26	005024	632	634#						
TST27	005116	634	637#						
TST3	003002	345#							
TST30	005150	643	652#						
TST31	005404	660	667	675	682	689	694	699	745#
TST32	005472	745	748#						





\$MAMS3	001230	60#																
\$MAMS4	001234	60#																
\$MBADR	001002	58#																
\$MFLG	012744	1177#*																
\$MNEW	011463	1171#																
\$MSGAD	001204	60#	1177*															
\$MSGLG	001206	60#	1177*															
\$MSGTY	001170	60#	1177*															
\$MSWR	011452	1171#																
\$MTYP1	001221	60#																
\$MTYP2	001225	60#																
\$MTYP3	001231	60#																
\$MTYP4	001235	60#																
\$MXCNT	010706	1170#																
\$NULL	001154	60#	1176															
\$NWTST=	000001	277#	315#	345#	356#	375#	394#	415#	436#	458#	481#	504#	523#	548#				
		570#	620#	622#	624#	626#	628#	630#	632#	634#	637#	652#	745#	748#				
		751#	754#	758#	779#	797#	811#	822#	834#	850#	875#	925#	953#	984#				
		1021#	1050#	1068#	1083#	1098#	1135#	1162#										
\$OCNT	011716	1173#*																
\$OMODE	011720	1173#*																
\$OVER	010672	1170#																
\$PASS	001176	60#	212*	1055	1165*	1170												
\$PASTM	001006	58#																
\$POWER	013116	1179#																
\$PWDN	012750	212	1179#															
\$PWRMG	013104	1179#																
\$PWRUP	013022	1179#																
\$QUES	001164	60#	1168	1171	1176													
\$RDCHR	011172	1171#	1255															
\$RDDEC=	***** U	1255																
\$RDLIN	011322	1171#	1255															
\$RDOCT=	***** U	1255																
\$RDSZ =	000010	1171#																
\$RTNAD	010056	1165#																
\$R2A =	***** U	1255																
\$SAVRE=	***** U	1255																
\$SAVR6	013114	1179#*																
\$SCOPE	010426	212	1170#															
\$SETUP=	000117	210#	212	271	1165	1168	1170	1171										
\$STUP =	177777	210#																
\$SVLAD	010636	1170#																
\$SVPC =	000204	54#																
\$SWR =	165400	10#	12	14	60	212	277	315	345	356	375	394	415	436				
		458	481	504	523	548	570	620	622	624	626	628	630	632				
		634	637	652	745	748	751	754	758	779	797	811	822	834				
		850	875	925	953	984	1021	1050	1068	1083	1098	1135	1162	1165				
		1168	1170	1179														
\$SWREG	001212	60#	212															
\$SWRMK=	000000	14	1170															
\$TESTN	001174	60#	280*	1170*	1292	1298	1300	1302										
\$TIMES	001160	60#	212*	277*	315*	345*	356*	375*	811*	822*	834*	1068*	1165*	1170*				
\$TKB	001146	60#	1171	1176														
\$TKS	001144	60#	1171*	1176														
\$TN =	000057	12#	52#	277#	315#	345#	351	356#	362	375#	384	387	394#	405				
		410	415#	426	431	436#	447	452	458#	471	477	481#	493	498				







TYPDEC	46#	1165															
TYPNAM	46#	271															
TYPNUM	46#																
TYPOCS	6#	46#															
TYPOCT	46#	1169	1171														
TYPTXT	46#																
WARN	985#	998	1022	1051	1069	1085	1100										
ZZ1	307#	315	368#	375	1123#	1135											
SSCMRE	60#																
SSCMTM	60#																
SSESCA	46#																
SSNEWT	46#	277	315	345	356	375	394	415	436	458	481	504	523	548	570		
	620	622	624	626	628	630	632	634	637	652	745	748	751	754	758		
	779	797	811	822	834	850	875	925	953	984	1021	1050	1068	1083	1098		
	1135	1162															
SSSET	1255#																
SSSETM	212#																
SSSKIP	46#	351	362	384	387	405	410	426	431	447	452	471	477	493	498		
	514	518	537	542	559	564	584	588	620	622	624	626	628	630	632		
	634	643	660	667	675	682	689	694	699	745	748	751	754	787	791		
	805	816	829	840	896	956	1064	1074	1077	1092	1153						
.EQUAT	6#	46															
.HEADE	5#	12															
.SETTR	5#																
.SETUP	5#	210															
.SWRHI	7#	14															
.SWRLO	14#																
.TRMTR	5#																
.SACT1	9#	54															
.SAPT8	60#																
.SAPTH	9#	58															
.SAPTY	9#	1177															
.SCATC	6#																
.SCMTA	7#	60															
.SEOP	7#	1165															
.SERRO	7#	1168															
.SERRT	7#	1169															
.SPOWE	6#	1179															
.SRDOC	5#																
.SREAD	8#	1171															
.SSCOP	8#	1170															
.STRAP	5#	1255															
.STYPB	6#																
.STYPD	8#	1175															
.STYPE	8#	1176															
.STYPO	6#	1173															

. ABS. 014656 000 OVR RW REL LCL D

ERRORS DETECTED: 0

CVIBAB, CVIBAB/CRF=CVIBAB  
 RUN-TIME: 24 10 1 SECONDS  
 RUN-TIME RATIO: 208/36=5.6

CVIBA-B MACY11 30G(1063) 25-FEB-83  
CVIBAB.P11 13-DEC-82 09:29

08:23 PAGE 49-2  
CROSS REFERENCE TABLE -- MACRO NAMES

6 8

SEQ 0097

CORE USED: 26K (51 PAGES)