

# IBV11-A

DIAGNOSTIC (30K)  
CVIBBA0

AH-F016A-MC  
COPYRIGHT © 77-78  
FICHE 1 OF 1

DEC 1978  
**digital**  
MADE IN USA

The microfiche card contains 48 frames of diagnostic data arranged in 8 rows and 6 columns. Each frame displays text-based information, likely system logs or diagnostic codes, which are too small to be legible in this image. The data appears to be organized into lists and tables across the frames.



IDENTIFICATION

B 1

SEQ 0001

Product Code: AC-F015A-MC  
Product Name: CVIBBA0 IBV11-A (30K) Diag  
Date : AUG 1978  
Maintainer: Diagnostic Engineering

Copyright (C) 1977,1978  
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	Scope Loops
5.3	Program and/or Operator Action
5.3.1	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'
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



1.0 ABSTRACT  
-----

This program is a copy of 'MD-11-DVIBA' with different bus and vector addresses. This program allows the user to check-out or debug the IBV11-A, LSI/IB interface option. In order to check-out a greater portion of logic on this option, a second IBV11-A is needed. See section 2.1. When a second IBV11-A can be obtained in order to run this diagnostic, 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. 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', 'DA11' 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 be electrically second on the LSI-11-BUS. It 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.

1. Absolute loader must be in memory.
2. Place binary tape in reader.
3. Type address \*7500 (\* determine by location of loader).
4. Type 'G' (program will be loaded into memory).

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:	1250	171420	::BASE ADDRESS OF EQUIPMENT :: UNDER TEST
\$VECT1:	1244	00C420	::INTERRUPT VECTOR #1
\$WREG:	176	000000	::MANUAL SWR.
IBS2:	1366	171430	: ADDRESS OF SECOND IBV11-A.
VECTA2:	1372	660	: VECTOR ADDRESS OF SECOND IBV11-A.
\$CDW1:	1254	000000	::DEVICE DESCRIPTOR WORD #1 (if = 000001 to use 'KGM' in testing 1st IBV-11) (Default = 00000 to disable use of KGM in tests.

#### 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. 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	BELL ON ERROR
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 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 'DDT', '\$SWREG' should be altered to '060000' to loop on current test and inhibit error typeout, then type 'P' to continue program execution.

### 5.3 Program and/or Operator Action

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

CVIBB-A

SWR=000000 NEW=

2. Program now waits for the operator to enter a switch register setting (see section 8.6). If the program is restarted, only "CVIBB-A" is typed. 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 RLV 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.

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

0.1 Minutes (6 sec) Iteration Inhibited - No Errors  
0.5 Minutes (30 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
44	BASIC DEFINITIONS
51	ACT11 HOOKS
55	APT PARAMETER BLOCK
57	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
175	REG ADDRESS AND COMMON TAGS
207	PROGRAM START
211	INITIALIZE THE COMMON TAGS
262	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
265	T1 *TEST THE ADDRESSABILITY OF THE IBS, IBD REGISTERS
303	T2 *TEST THAT BASE ADDRESSES +4,+6 RESPOND WHEN ADDRESSED
332	T3 *TEST THAT IBS IS CLEAR AT INIT OF TESTING
343	T4 *TEST THAT IBD IS CLEAR AT INIT OF TESTING
361	T5 *TEST THAT BASE ADDRESSES +4,+6 RETURN ZERO WHEN READ
379	T6 *TEST THAT WE CAN SET TCS, TCS SETS CMD
400	T7 *TEST THAT EOP WILL SET
421	T10 *TEST THAT RE WILL SET + CLEAR
442	T11 *TEST THAT IBC WILL SET AND CLEAR
469	T12 *TEST THAT TON (BIT05) AND TKR SET AND CLEAR
491	T13 *MAKE SURE WE CAN SET AND CLEAR BIT06 (IE)
515	T14 *TEST THAT BIT 7 (ACC) CAN BE SET AND CLEARED
563	T15 *TEST THAT IBD BIT 0 CAN BET SET + CLEARED
564	T16 *TEST THAT IBD BIT 1 CAN BET SET + CLEARED
565	T17 *TEST THAT IBD BIT 2 CAN BET SET + CLEARED
566	T20 *TEST THAT IBD BIT 3 CAN BET SET + CLEARED
567	T21 *TEST THAT IBD BIT 4 CAN BET SET + CLEARED
568	T22 *TEST THAT IBD BIT 5 CAN BET SET + CLEARED
569	T23 *TEST THAT IBD BIT 6 CAN BET SET + CLEARED
570	T24 *TEST THAT IBD BIT 7 CAN BET SET + CLEARED
572	T25 *TEST THAT NO DATA GETS XFERRERD, IF NOT ENABLED
586	T26 *TEST IBD BITS DAC, AND DAV
662	T27 *TEST THAT REN SETS WHEN REM SETS, ALSO TEST CLEAR
664	T30 *TEST THAT IFC SETS WHEN IBS SETS, ALSO TEST CLEAR
666	T31 *TEST THAT ATN SETS WHEN TCS SETS, ALSO TEST CLEAR
668	T32 *TEST THAT EOI SETS WHEN EOP SETS, ALSO TEST CLEAR
670	T33 *TEST THAT RFD SET WHEN CSR CLEAR, CLEAR WHEN ACC SET
688	T34 *TEST THAT WE CAN GENERATE AN ER2
701	T35 *TEST THAT BUS INIT CLEARS ACC,TON,LON,REM,EIP,TCS
712	T36 *TEST IBC CLEARS ACC,TON,LON,REM AND EOP
725	T37 *TEST THAT BUS INIT INDIRECTLY CLEARS IBD
737	
738	INTERRUPT TESTS
739	
741	T40 *TEST THAT CMD CAN GENERATE AN INTERRUPT B
758	T41 *TEST THAT TKR AND LNR CAN GENERATE INTERRUPTS
791	T42 *TEST THAT ER2 CAN GENERATE AN INTERRUPT
812	
813	SECOND MODULE TESTS
814	
816	T43 *TEST THAT MODULE PASSES 'BIAKI'

CVIBB-A MACY11 27(654) 19-SEP-78 11:35  
 CVIBBA.P11 TABLE OF CONTENTS

SEQ 0014

853	T44	*TEST THAT SRQ CAN GENERATE AN INTERRUPT
880	T45	*TEST THAT ERROR1 IS GENERATED IF ATN IS ON THE IB BUS
895	T46	*TEST THAT ERROR 1 IS GENERATED IF IFC IS PUT ON IB BUS BY SECOUND MODULE
911	T47	*TEST THAT ERROR 1 IS GENERATED IF REN IS ON IB BUS
925	T50	*TEST THAT AN ERROR 1 CAN GENERATE AN INTERRUPT
961	T51	*TEST THAT DATA CAN BE XFERRERD BETWEEN THE MODULE UNDER TEST AND THE KGM
987	T52	*TEMP END OF TESTS
991		SYSMAC ROUTINES:
993		END OF PASS ROUTINE
995		ERROR HANDLER ROUTINE
996		ERROR MESSAGE TYPEOUT ROUTINE
997		SCOPE HANDLER ROUTINE
998		TTY INPUT ROUTINE
999		BINARY TO OCTAL (ASCII) AND TYPE
1001		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1002		TYPE ROUTINE
1003		APT COMMUNICATIONS ROUTINE
1004		POWER DOWN AND UP ROUTINES
1080		TRAP DECODER
(3)		TRAP TABLE
1082		MESSAGES AND TABLES

```
1          .NLIST MC,MD,CND
2          .LIST ME
3          .ENABL ABS
4          .ENABL AMA
5          .MCALL .HEADER,.SETUP,.SETTRAP,.TRMTRP,.$TRAP,.$RDOCT
6          .MCALL .$TYPBIN,TYPEOCS,.$POWER,.$CATCH,.$TYPOCT,.EQUAT
7          .MCALL .$CMTAG,.$SWRHI,.$EOP,.$ERROR,.$ERRTYP
8          .MCALL .$TYPDEC,.$SCOPE,.$READ,.$TYPE
9          .MCALL .$ACT11,.$APTHDR,.$APTTYPE
10         $SWR=167400
11
12         .TITLE CVIBB-A
13         ;*COPYRIGHT (C) 1978
14         ;*DIGITAL EQUIPMENT CORP.
15         ;*MAYNARD, MASS. 01754
16         ;*
17         ;*PROGRAM BY EDWARD C. BADGER MOD BY R. SHOOP
18         ;*
19         ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
20         ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
21         ;*
22         $TN=1
23
24         .SBTTL OPERATIONAL SWITCH SETTINGS
25         ;*
26         ;*          SWITCH          USE
27         ;*          -----          -----
28         ;*          15          HALT ON ERROR
29         ;*          14          LOOP ON TEST
30         ;*          13          INHIBIT ERROR TYPEOUTS
31         ;*          11          INHIBIT ITERATIONS
32         ;*          10          BELL ON ERROR
33         ;*          9          LOOP ON ERROR
34         ;*          8          LOOP ON TEST IN SWR<7:0>
35
36         .SBTTL TRAP CATCHER
37         ;*
38         ;*=0
39         ;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ".+2"
40         ;*AND "JSR PC,RO" SEQUENCE TO CATCH ILLEGAL INTERRUPTS,
41         ;*AND INTERRUPTS TO THE WRONG VECTOR.
42         ;*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
43         ;*VECTORS
44         ;*=4
45         .WORD IOTRD,200          ;HANDLE BUSS ERROR.
46         ;*=174
47         DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER.
48         SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER.
49         ;*=100
50         .WORD 104,200,2          ;IF 'B EVENT' ON Q-BUS IS
51         ;*CONNECTED, WE NEED A WAY OF
52         ;*IGNORING ITS INTERRUPTS.
53         ;*=200
54         JMP START
```



```

44      .SBTTL BASIC DEFINITIONS
(1)
(1)      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)      001100      STACK= 1100
(1)      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
(1)      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)      ;*MISCELLANEOUS DEFINITIONS
(1)      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
(1)      000012      LF= 12      ;;CODE FOR LINE FEED
(1)      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
(1)      000200      CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)      177776      PS= 177776  ;;PROCESSOR STATUS WORD
(1)      .EQUIV PS,PSW
(1)      177774      STKLMT= 177774 ;;STACK LIMIT REGISTER
(1)      177772      PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)      177570      DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1)      177570      DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)
(1)      ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)      000000      R0= %0      ;;GENERAL REGISTER
(1)      000001      R1= %1      ;;GENERAL REGISTER
(1)      000002      R2= %2      ;;GENERAL REGISTER
(1)      000003      R3= %3      ;;GENERAL REGISTER
(1)      000004      R4= %4      ;;GENERAL REGISTER
(1)      000005      R5= %5      ;;GENERAL REGISTER
(1)      000006      R6= %6      ;;GENERAL REGISTER
(1)      000007      R7= %7      ;;GENERAL REGISTER
(1)      000006      SP= %6      ;;STACK POINTER
(1)      000007      PC= %7      ;;PROGRAM COUNTER
(1)
(1)      ;*PRIORITY LEVEL DEFINITIONS
(1)      000000      PR0= 0      ;;PRIORITY LEVEL 0
(1)      000040      PR1= 40     ;;PRIORITY LEVEL 1
(1)      000100      PR2= 100    ;;PRIORITY LEVEL 2
(1)      000140      PR3= 140    ;;PRIORITY LEVEL 3
(1)      000200      PR4= 200    ;;PRIORITY LEVEL 4
(1)      000240      PR5= 240    ;;PRIORITY LEVEL 5
(1)      000300      PR6= 300    ;;PRIORITY LEVEL 6
(1)      000340      PR7= 340    ;;PRIORITY LEVEL 7
(1)
(1)      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
(1)      100000      SW15= 100000
(1)      040000      SW14= 40000
(1)      020000      SW13= 20000
(1)      010000      SW12= 10000
(1)      004000      SW11= 4000
(1)      002000      SW10= 2000
(1)      001000      SW09= 1000
(1)      000400      SW08= 400
(1)      000200      SW07= 200
(1)      000100      SW06= 100
(1)      000040      SW05= 40
(1)      000020      SW04= 20

```

```
(1) 000010 SW03= 10
(1) 000004 SW02= 4
(1) 000002 SW01= 2
(1) 000001 SW00= 1
(1) .EQUIV SW09,SW9
(1) .EQUIV SW08,SW8
(1) .EQUIV SW07,SW7
(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)
(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)
(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
45
46 171420 ABASE= 171420
47 000420 AVECT1= 420
48 000200 APRIOR= 200
49 000001 $TN=1
50
51 .SBTTL ACT11 HOOKS
(1)
(2) ;;*****
(1) :HOOKS REQUIRED BY ACT11
(1) $SVPC=. ;SAVE PC
(1) 000046 000046 ;=46
(1) 000046 007066 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1) 000052 000052 ;=52
(1) 000052 000000 .WORD 0 ;;2)SET LOC.52 TO ZERO
(1) 000052 000204 ;=$SVPC ;; RESTORE PC
52
53 001000 ;=1000
54
55 .SBTTL APT PARAMETER BLOCK
(1)
(2) ;;*****
(1) :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2) ;;*****
(1) 001000 .SX=. ;;SAVE CURRENT LOCATION
(1) 000024 000024 ;=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200 200 ;;FOR APT START UP
(1) 000044 000044 ;=44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000 $APTHDR ;;POINT TO APT HEADER BLOCK
(1) 001000 ;=.$X ;;RESET LOCATION COUNTER
(2) ;;*****
(1) :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1) :INTERFACE SPEC.
(1)
(1) $APTHD:
(1) 001000 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001174 $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)
56
```



```

57          .SBTTL COMMON TAGS
(1)
(2)          ::*****
(1)          ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(1)          ::*USED IN THE PROGRAM.
(1)
(1)          001100          .=1100
(1) 001100          $CMTAG:          ::START OF COMMON TAGS
(1) 001100          000000          .WORD          0
(1) 001102          000          $TSTNM: .BYTE          0          ::CONTAINS THE TEST NUMBER
(1) 001103          000          $ERFLG: .BYTE          0          ::CONTAINS ERROR FLAG
(1) 001104          000000          $ICNT:  .WORD          0          ::CONTAINS SUBTEST ITERATION COUNT
(1) 001106          000000          $LPADR: .WORD          0          ::CONTAINS SCOPE LOOP ADDRESS
(1) 001110          000000          $LPERR: .WORD          0          ::CONTAINS SCOPE RETURN FOR ERRORS
(1) 001112          000000          $ERTTL: .WORD          0          ::CONTAINS TOTAL ERRORS DETECTED
(1) 001114          000          $ITEMB: .BYTE          0          ::CONTAINS ITEM CONTROL BYTE
(1) 001115          001          $ERMAX: .BYTE          1          ::CONTAINS MAX. ERRORS PER TEST
(1) 001116          000000          $ERRPC: .WORD          0          ::CONTAINS PC OF LAST ERROR INSTRUCTION
(1) 001120          000000          $GDADR: .WORD          0          ::CONTAINS ADDRESS OF 'GOOD' DATA
(1) 001122          000000          $BDADR: .WORD          0          ::CONTAINS ADDRESS OF 'BAD' DATA
(1) 001124          000000          $GDDAT: .WORD          0          ::CONTAINS 'GOOD' DATA
(1) 001126          000000          $BDDAT: .WORD          0          ::CONTAINS 'BAD' DATA
(1) 001130          000000          .WORD          0          ::RESERVED--NOT TO BE USED
(1) 001132          000000          .WORD          0
(1) 001134          000          $AUTOB: .BYTE          0          ::AUTOMATIC MODE INDICATOR
(1) 001135          000          $INTAG: .BYTE          0          ::INTERRUPT MODE INDICATOR
(1) 001136          000000          .WORD          0
(1) 001140          177570          SWR:      .WORD          DSWR          ::ADDRESS OF SWITCH REGISTER
(1) 001142          177570          DISPLAY: .WORD          DDISP          ::ADDRESS OF DISPLAY REGISTER
(1) 001144          177560          $TKS:      177560          ::TTY KBD STATUS
(1) 001146          177562          $TKB:      177562          ::TTY KBD BUFFER
(1) 001150          177564          $TPS:      177564          ::TTY PRINTER STATUS REG. ADDRESS
(1) 001152          177566          $TPB:      177566          ::TTY PRINTER BUFFER REG. ADDRESS
(1) 001154          000          $NULL:    .BYTE          0          ::CONTAINS NULL CHARACTER FOR FILLS
(1) 001155          002          $FILLS:   .BYTE          2          ::CONTAINS # OF FILLER CHARACTERS REQUIRED
(1) 001156          012          $FILLC:   .BYTE          12         ::INSERT FILL CHARS. AFTER A 'LINE FEED'
(1) 001157          000          $TPFLG:   .BYTE          0          ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(1) 001160          000000          $TIMES:   0          ::MAX. NUMBER OF ITERATIONS
(1) 001162          000000          $ESCAPE:  0          ::ESCAPE ON ERROR ADDRESS
(1) 001164          177607          $BELL:    .ASCIZ          <207><377><377> ::CODE FOR BELL
(1) 001170          077          $QUES:    .ASCII          /?/          ::QUESTION MARK
(1) 001171          015          $CRLF:    .ASCII          <15>          ::CARRIAGE RETURN
(1) 001172          000012          $LF:      .ASCIZ          <12>          ::LINE FEED
(2)          ::*****
(2)          .SBTTL APT MAILBOX-ETABLE
(2)          ::*****
(2)          .EVEN
(2) 001174          $MAIL:          ::APT MAILBOX
(2) 001174          000000          $MSGTY:   .WORD          AMSGTY  ::MESSAGE TYPE CODE
(2) 001176          000000          $FATAL:   .WORD          AFATAL   ::FATAL ERROR NUMBER
(2) 001200          000000          $TESTN:   .WORD          ATESTN  ::TEST NUMBER
(2) 001202          000000          $PASS:    .WORD          APASS    ::PASS COUNT
(2) 001204          000000          $DEVCT:   .WORD          ADEVCT  ::DEVICE COUNT
  
```

(2)	001206	000000	\$UNIT: .WORD	AUNIT	:: I/O UNIT NUMBER
(2)	001210	000000	\$MSGAD: .WORD	AMSGAD	:: MESSAGE ADDRESS
(2)	001212	000000	\$MSGLG: .WORD	AMSGLG	:: MESSAGE LENGTH
(2)	001214		\$ETABLE:		:: APT ENVIRONMENT TABLE
(2)	001214	000	\$ENV: .BYTE	AENV	:: ENVIRONMENT BYTE
(2)	001215	000	\$ENVM: .BYTE	AENVM	:: ENVIRONMENT MODE BITS
(2)	001216	000000	\$SWREG: .WORD	ASWREG	:: APT SWITCH REGISTER
(2)	001220	000000	\$USWR: .WORD	AUSWR	:: USER SWITCHES
(2)	001222	000000	\$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)	001224	000	\$MAMS1: .BYTE	AMAMS1	:: HIGH ADDRESS, M.S. BYTE
(2)	001225	000	\$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)	001226	000000	\$MADR1: .WORD	AMADR1	:: HIGH ADDRESS, BLK#1
(2)			::*		MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001230	000	\$MAMS2: .BYTE	AMAMS2	:: HIGH ADDRESS, M.S. BYTE
(2)	001231	000	\$MTYP2: .BYTE	AMTYP2	:: MEM. TYPE, BLK#2
(2)	001232	000000	\$MADR2: .WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
(2)	001234	000	\$MAMS3: .BYTE	AMAMS3	:: HIGH ADDRESS, M.S. BYTE
(2)	001235	000	\$MTYP3: .BYTE	AMTYP3	:: MEM. TYPE, BLK#3
(2)	001236	000000	\$MADR3: .WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
(2)	001240	000	\$MAMS4: .BYTE	AMAMS4	:: HIGH ADDRESS, M.S. BYTE
(2)	001241	000	\$MTYP4: .BYTE	AMTYP4	:: MEM. TYPE, BLK#4
(2)	001242	000000	\$MADR4: .WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
(2)	001244	000420	\$VECT1: .WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
(2)	001246	000000	\$VECT2: .WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
(2)	001250	171420	\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001252	000000	\$DEVN: .WORD	ADEVN	:: DEVICE MAP
(2)	001254	000000	\$CDW1: .WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
(2)	001256		\$ETEND:		
(2)			.MEXIT		

```

(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) $ERRTB:
(1) 001256
58
62
63 ;ITEM 1
64
65 001256 012232 EM1 ;IBS FUNCTION ERROR
66 001260 012453 DH1 ;TEST ERRPC IB ADDR
67 001262 012654 DT1 ;$TESTN,$ERRPC,IBS
68 001264 012724 DF0 ;ALL NUMBERS ARE IN OCTAL FORM.
69
70 ;ITEM 2
71
72 001266 012260 EM2 ;IBD FUNCTION ERROR
73 001270 012453 DH1 ;TEST ERRPC IB ADDR
74 001272 012654 DT1 ;$TESTN,$ERRPC,IBS
75 001274 012724 DF0 ;ALL NUMBERS ARE IN OCTAL FORM.
76
77 ;ITEM 3
78
79 001276 012306 EM3 ;IBS DATA ERROR
80 001300 012521 DH3 ;TEST ERRPC GOOD BAD
81 001302 012670 DT3 ;$TESTN,$ERRPC,$GDDAT,$BDDAT
82 001304 012724 DF0 ;ALL NUMBERS ARE IN OCTAL FORM.
83
84 ;ITEM 4
85
86 001306 012330 EM4 ;IBD DATA ERROR
87 001310 012521 DH3 ;TEST ERRPC GOOD BAD
88 001312 012670 DT3 ;$TESTN,$ERRPC,$GDDAT,$BDDAT
89 001314 012724 DF0 ;ALL NUMBERS ARE IN OCTAL FORM.
90
91 ;ITEM 5
92
93 001316 012352 EM5 ;IBS/IBD ADDRESS ERROR
94 001320 012556 DH5 ;TEST ERROR PC ADDRESS
95 001322 012702 DT5 ;$STNM,$ERRPC,IBS
96 001324 012724 DF0 ;ALL NUMBERS ARE IN OCTAL FORM.
97
98 ;ITEM 6
99
  
```



100	001326	012402	EM6	:IBWC/IBCA DATA ERROR
101	001330	012521	DH3	:TEST ERRPC GOOD BAD
102	001332	012670	DT3	:\$TESTN,\$ERRPC,\$GDDAT,\$BDDAT
103	001334	012724	DF0	:ALL NUMBERS ARE IN OCTAL FORM.
104				
105			:ITEM 7	
106	001336	012431	EM7	:INTERRUPT ERROR
107	001340	012607	DH7	:TEST ERRPC TO FROM ADDR.
108	001342	012712	DT7	:TSTNM,#ERRPC,TRTO,TRFRO
109	001344	012724	DF0	:ALL NUMBERS ARE IN OCTAL FORM.

110  
125  
136  
140  
148  
154  
174

.SBTTL REG ADDRESS AND COMMON TAGS  
 :WARNING IF DEVICE # IS AT DIFFERENT ADDRESS OR VECTOR  
 :DO NOT PATCH THESE LOCATIONS - SEE PROGRAM DOCUMENTATION.

179	001346	171420	IBS:	.WORD	ABASE	;>NO	<;CONTROL AND STATUS REGISTER.
180	001350	171422	IBD:	.WORD	ABASE+2	;>PATCHES	<;DATA REGISTER.
181	001352	171424	IBWC:	.WORD	ABASE+4		:ADDRESS RESERVED FOR
182	001354	171426	IBCA:	.WORD	ABASE+6		:FUTURE USE
183	001356	000420	VECTA:	.WORD	AVECT1	;>ALLOWED	<;VECTOR ADDRESS.
184	001360	000424	VECTB:	.WORD	AVECT1+4	;>HERE!	<;VECTOR ADDR. +4.
185	001362	000430	VECTC:	.WORD	AVECT1+10		
186	001364	000434	VECTD:	.WORD	AVECT1+14		
187	001366	171430	IBS2:	.WORD	ABASE+10		
188	001370	171432	IBD2:	.WORD	ABASE+12		
189	001372	000660	VECTA2:	.WORD	660		
190	001374	000664	VECTB2:	.WORD	664		
191	001376	000670	VECTC2:	.WORD	670		
192	001400	000674	VECTD2:	.WORD	674		

:VECTOR ADDRESSES +2 LOCATIONS.

196	001402	000422	PRA:	.WORD	AVECT1+2		:NOTE: DO NOT ATTEMPT TO PATCH
197	001404	000426	PRB:	.WORD	AVECT1+6		THESE LOCATIONS IF A VECTOR
198	001406	000432	PRC:	.WORD	AVECT1+12		VARYIES . ALTER LOCATION
199	001410	000436	PRD:	.WORD	AVECT1+16		'\$VECT1:'.
200							
201	001412	000442	PRA2:	.WORD	AVECT1+22		: IF TEST MODULE VECTOR IS
202	001414	000446	PRB2:	.WORD	AVECT1+26		DIFFERENT, YOU MUST CHANGE
203	001416	000452	PRC2:	.WORD	AVECT1+32		LOCATION 'VECTA2:'.
204	001420	000456	PRD2:	.WORD	AVECT1+36		

.SBTTL PROGRAM START

210  
211 001422 START:  
 (1) .SBTTL INITIALIZE THE COMMON TAGS

```

(1)          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001422 012706 001100      MOV    # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 001426 005026             CLR    (R6)+            ;;CLEAR MEMORY LOCATION
(1) 001430 022706 001140      CMP    #SWR,R6 ;;DONE?
(1) 001434 001374             BNE    -6              ;;LOOP BACK IF NO
(1) 001436 012706 001100      MOV    #STACK,SP      ;;SETUP THE STACK POINTER
(1)          ;;INITIALIZE A FEW VECTORS
(1) 001442 012737 007464 000020  MOV    # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001450 012737 000340 000022  MOV    #340,@#IOTVEC+2 ;;LEVEL 7
(1) 001456 012737 007142 000030  MOV    # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001464 012737 000340 000032  MOV    #340,@#EMTVEC+2 ;;LEVEL 7
(1) 001472 012737 012152 000034  MOV    # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001500 012737 000340 000036  MOV    #340,@#TRAPVEC+2;LEVEL 7
(1) 001506 012737 011724 000024  MOV    # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 001514 012737 000340 000026  MOV    #340,@#PWRVEC+2 ;;LEVEL 7
(1) 001522 005037 001160             CLR    $TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001526 005037 001162             CLR    $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001532 112737 000001 001115  MOVB  #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
(1) 001540 012737 001540 001106  MOV    #.,$LPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001546 012737 001546 001110  MOV    #.,$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) 001554 013746 000004             MOV    @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 001560 012737 001614 000004  MOV    #64$,@#ERRVEC  ;;SET UP ERROR VECTOR
(2) 001566 012737 177570 001140  MOV    #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001574 012737 177570 001142  MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001602 022777 177777 177330  CMP    #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
(2) 001610 001012             BNE    66$           ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)          ;;AND THE HARDWARE SWR IS NOT = -1
(2) 001612 000403             BR     65$           ;;BRANCH IF NO TIMEOUT
(2) 001614 012716 001622          64$: MOV    #65$,(SP)      ;;SET UP FOR TRAP RETURN
(2) 001620 000002             RTI
(2) 001622 012737 000176 001140  65$: MOV    #SWREG,SWR  ;;POINT TO SOFTWARE SWR
(2) 001630 012737 000174 001142  MOV    #DISPREG,DISPLAY
(2) 001636 012637 000004          66$: MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 001642 005037 001202             CLR    $PASS         ;;CLEAR PASS COUNT
(2) 001646 132737 000200 001215  BITB  #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
(2) 001654 001403             BEQ    67$           ;;YES,USE NON-APT SWITCH
(2) 001656 012737 001216 001140  MOV    # $SWREG,SWR  ;;NO,USE APT SWITCH REGISTER
(2) 001664          67$:
212 001664 012737 012102 000004  MOV    #IOTRD,ERRVEC ;SET TO HANDLE BUS ERRORS.
213 001672 012737 000200 000006  MOV    #200,ERRVEC+2
214
215 001700 013737 001250 001346  MOV    $BASE,IBS     ;GET BASE ADDR.
216 001706 013737 001346 001350  MOV    IBS,IBD       ;FIX DATA BUFFER=
217 001714 062737 000002 001350  ADD    #2,IBD        ;CSR+2
218 001722 013737 001350 001352  MOV    IBD,IBWC
219 001730 062737 000002 001352  ADD    #2,IBWC
220 001736 013737 001352 001354  MOV    IBWC,IBCA
221 001744 062737 000002 001354  ADD    #2,IBCA
222 001752 013737 001244 001356  MOV    $VECT1,VECTA ;GET VECTOR ADDR.
223 001760 042737 170000 001356  BIC    #170000,VECTA ;STRIP JUNK
224 001766 013737 001346 012662  MOV    IBS,IBSA

```



```

225 001774 013737 001350 012664      MOV      IBD,IBDA
226 002002 013737 001366 001370      MOV      IBS2,IBD2
227 002010 062737 000002 001370      ADD      #2,IBD2
228 002016 013737 001356 001360      MOV      VECTA,VECTB
229 002024 062737 000004 001360      ADD      #4,VECTB
230 002032 013737 001360 001362      MOV      VECTB,VECTC
231 002040 062737 000004 001362      ADD      #4,VECTC
232 002046 013737 001362 001364      MOV      VECTC,VECTD
233 002054 062737 000004 001364      ADD      #4,VECTD
234 002062 013737 001372 001374      MOV      VECTA2,VECTB2
235 002070 062737 000004 001374      ADD      #4,VECTB2
236 002076 013737 001374 001376      MOV      VECTB2,VECTC2
237 002104 062737 000004 001376      ADD      #4,VECTC2
238 002112 013737 001376 001400      MOV      VECTC2,VECTD2
239 002120 062737 000004 001400      ADD      #4,VECTD2
240
241 002126 013737 001356 001402      MOV      VECTA,PRA          ;SET UP VECTOR+2 ADDRESSES.
242 002134 062737 000002 001402      ADD      #2,PRA
243 002142 013737 001402 001404      MOV      PRA,PRB
244 002150 062737 000004 001404      ADD      #4,PRB
245 002156 013737 001404 001406      MOV      PRB,PRC
246 002164 062737 000004 001406      ADD      #4,PRC
247 002172 013737 001406 001410      MOV      PRC,PRD
248 002200 062737 000004 001410      ADD      #4,PRD
249 002206 013737 001372 001412      MOV      VECTA2,PRA2
250 002214 062737 000002 001412      ADD      #2,PRA2
251 002222 013737 001412 001414      MOV      PRA2,PRB2
252 002230 062737 000004 001414      ADD      #4,PRB2
253 002236 013737 001414 001416      MOV      PRB2,PRC2
254 002244 062737 000004 001416      ADD      #4,PRC2
255 002252 013737 001416 001420      MOV      PRC2,PRD2
256 002260 062737 000004 001420      ADD      #4,PRD2
257 002266
258
(1) 002266 013777 001402 177062      MOV      PRA,@VECTA ;/-RESV- ;/RESTORE VECTOR FOR
(1) 002274 012777 004700 177100      MOV      #4700,@PRA ;/ILLEGAL INTRO.
259
(1) 002302 013777 001404 177050      MOV      PRB,@VECTB ;/-RESV- ;/RESTORE VECTOR FOR
(1) 002310 012777 004700 177066      MOV      #4700,@PRB ;/ILLEGAL INTRO.
260
(1) 002316 013777 001406 177036      MOV      PRC,@VECTC ;/-RESV- ;/RESTORE VECTOR FOR
(1) 002324 012777 004700 177054      MOV      #4700,@PRC ;/ILLEGAL INTRO.
261
(1) 002332 013777 001410 177024      MOV      PRD,@VECTD ;/-RESV- ;/RESTORE VECTOR FOR
(1) 002340 012777 004700 177042      MOV      #4700,@PRD ;/ILLEGAL INTRO.
262
(1)
(1) 002346 005227 177777      .SBTTL   TYPE PROGRAM NAME
(1) 002352 001041      ;;TYPE  THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002354 104401 002422      INC      #-1          ;;FIRST TIME?
(1) 002354 104401 002422      BNE      64$          ;;BRANCH IF NO
(2) 002360 005737 000042      TYPE     ,65$        ;;TYPE ASCIZ STRING
(2) 002364 001012      .SBTTL   GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002366 123727 001214 000001      TST      @#42        ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002366 123727 001214 000001      BNE      66$          ;;BRANCH IF YES
(2) 002366 123727 001214 000001      CMPB    $ENV,#1     ;;ARE WE RUNNING UNDER APT?

```



```

(2) 002374 001406          BEQ    66$          ;;BRANCH IF YES
(2) 002376 023727 001140 000176  CMP    SWR,#SWREG    ;;SOFTWARE SWITCH REG SELECTED?
(2) 002404 001005          BNE    67$          ;;BRANCH IF NO
(2) 002406 104406          GTSWR                    ;;GET SOFT-SWR SETTINGS
(2) 002410 000403          BR     67$
(2) 002412 112737 000001 001134 66$:  MOVB   #1,$AUTOB    ;;SET AUTO-MODE INDICATOR
(2) 002420 000416          67$:  BR     64$          ;;GET OVER THE ASCIZ
(1) 002420 000416          ;;65$: .ASCIZ <CRLF>#CVIBBA IBV11A DIAGNOSTIC#<CRLF>
(1) 002456 000005          64$:  RESET
263 002456 000005
264
265
266
267
268
269
270
271 002520 013746 000004          1$:  MOV    ERRVEC,-(SP)  ;SAVE CONTENTS OF ADDR. 4
272 002524 012737 002552 000004  MOV    #2$,ERRVEC   ;SET TIME-OUT TRAP VECTOR TO HANDLE
273                                     ;IN CASE WE TIME OUT WHEN
274                                     ;WE ADDR. THE IBV-11
275
276 002532 005777 176610          TST    @IBS         ;ADDR THE IBS, IF NO RESPONSE,
277                                     ;WILL TRAP TO 2$ FROM HERE
278 002536 012737 002560 000004  MOV    #3$,ERRVEC   ;CHANGE FOR ADDRESSING THE IBD REG.
279
280 002544 005777 176600          TST    @IBD         ;ADDR THE IBD REG.
281                                     ;WE'LL TRAP TO 3$ FROM HERE IF BAD.
282 002550 000406          BR     4$
283 002552          2$:  ADD    #4,SP        ;/ADD #4 TO STACK POINTER.
(1) 002552 062706 000004
284
                ;;$$$$$$$$$>>> ERROR <<<$$$$$$$$$
(1) 002556 104005          ERROR 5             ;/MODULE FAULT DETECTED:
285                                     ;IBS REGISTER COULD NOT BE
286                                     ;ADDRESSED
                ;;$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$
288 002560          3$:  ADD    #4,SP        ;/ADD #4 TO STACK POINTER.
(1) 002560 062706 000004
289
                ;;$$$$$$$$$>>> ERROR <<<$$$$$$$$$
(1)

```

```

(1) 002564 104005          ERROR 5          ;/MODULE FAULT DETECTED:
290                                     ;ADDRESSED

                                     ;:$$$$$$$$$$$^^^ ERROR ^^ $$$$$$$$$$
292 002566 012637 000004  4$:  MOV      (SP)+,ERRVEC  ;RESTORE CONTENTS OF LOC 4.
293                                     ;/PR
(1) 002572 012746 000000  MOV      #0,-(SP)    ;/SET CPU PRIORITY ON RETURN
(1) 002576 012746 002604  MOV      #64$,-(SP)  ;/SHOW RETURN ADDRESS
(1) 002602 000002          RTI          ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 002604          64$:          ;/IN STATUS REG.)

                                     ;:*****
(3) *TEST 2          *TEST THAT BASE ADDRESSES +4,+6 RESPOND WHEN ADDRESSED
(4)
(4)
(4)
(4)
(4)
(3)
(2) 002604 000004          TST2:  SCOPE
(1) 002606 012737 000010 001160  MOV      #10,$TIMES  ;;DO 10 ITERATIONS
304
305 002614 013746 000004  MOV      ERRVEC,-(SP) ;SAVE CONTENTS OF ADDR 4.
306 002620 012737 002646 000004  MOV      #1$,ERRVEC  ;SET TIME OUT TRAP VECTOR TO HANDLE
307                                     ;IN CASE WE TIME OUT WHEN WE
308                                     ;ADDRESS THE IBV-11 ADDRESSES +4,+6.
309 002626 005777 176520  TST      @IBWC      ;TEST BASE ADDRESS +4, IF NO RESPONSE
310                                     ;WILL TRAP TO 1$ FROM HERE
311
312 002632 012737 002656 000004  MOV      #2$,ERRVEC  ;CHANGE FOR ADDRESSING +6 ADDR.
313
314 002640 005777 176510  TST      @IBCA      ;ADDR THE +6 ADDR. - TRAP IF BAD.
315 002644 000407          BR       3$          ;CONTINUE IF GOOD.
316
317 002646          1$:          ADD      #4,SP      ;/ADD #4 TO STACK POINTER.
(1) 002646 062706 000004

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

(1) 002652 104005          ERROR 5          ;/MODULE FAULT DETECTED:
319                                     ;BASE ADDR+4 COULD NOT
320                                     ;BE ADDRESSED.

                                     ;:$$$$$$$$$$$^^^ ERROR ^^ $$$$$$$$$$
322 002654 000403          BR       3$

323
324 002656          2$:          ADD      #4,SP      ;/ADD #4 TO STACK POINTER.
(1) 002656 062706 000004

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

```

```

(1)
(1) 002662 104005          ERROR 5          ;/MODULE FAULT DETECTED:
326                               ;BASE ADDR+6 COULD NOT
327                               ;BE ADDRESSED.

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

329
330 002664 012637 000004    3$:  MOV      (SP)+,ERRVEC  ;RESTORE CONTENTS OF LOC 4.
331
332                               ;*****
(3)                               ;*TEST 3          *TEST THAT IBS IS CLEAR AT INIT OF TESTING
(3)                               ;*****
(2) 002670 000004          TST3:  SCOPE
(1) 002672 012737 000001 001160  MOV      #1,$TIMES      ;;DO 1 ITERATION
333
334 002700 000005          RESET          ;ISSUE SYSTEM INIT.
335
336 002702 005037 001124    CLR      $GDDAT        ;EXPECT ZERO CSR.
337 002706 017737 176434 001126  MOV      @IBS,$BDDAT   ;READ CSR.
338 002714 001401          BEQ      TST4          ;;
339

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

(1)
(1) 002716 104003          ERROR 3          ;/MODULE FAULT DETECTED:
340                               ;IBS NOT CLEAR ON INT.

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

342
343                               ;*****
(3)                               ;*TEST 4          *TEST THAT IBD IS CLEAR AT INIT OF TESTING
(3)                               ;*****
(2) 002720 000004          TST4:  SCOPE
(1) 002722 012737 000001 001160  MOV      #1,$TIMES      ;;DO 1 ITERATION
344
345 002730 000005          RESET          ;ISSUE SYSTEM INITIALIZE.
346
347 002732 005037 001124    CLR      $GDDAT        ;EXPECT ZERO CSR.
348 002736 117737 176406 001126  MOV      @IBD,$BDDAT   ;READ DBR
349 002744 001401          BEQ      TST5          ;;
350

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

(1)
(1) 002746 104004          ERROR 4          ;/MODULE FAULT DETECTED:
351                               ;IBD NOT CLEAR ON INIT.

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

353
360
361                               ;*****

```



```

(3) ;*TEST 5 *TEST THAT BASE ADDRESSES +4,+6 RETURN ZERO WHEN READ
(4) ;*
(4) ;*BASE ADDRESS +4 AND +6 SHOULD RETURN A ZERO WHEN
(4) ;*READ, IN THIS TEST WE WILL TRY THAT.
(4) ;*
(3) ;:*****

```

```

(2) 002750 000004 TST5: SCOPE
(1) 002752 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
362
363 002760 005037 CLR $GDDAT ;EXPECT ZERO RETURN
364 002764 017737 176362 001126 MOV @IBWC,$BDDAT ;READ BASE ADDRESS+4
365 002772 001402 BEQ 1$ ;IF ZERO - GOOD.
366

```

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

```

(1)
(1) 002774 104006 ERROR 6 ;/MODULE FAULT DETECTED:
367 ;SHOULD HAVE READ BACK ZERO FROM
368 ;THIS ADDR.

```

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

```

370 002776 000405 BR TST6 ;;
371
372 003000 017737 176350 001126 1$: MOV @IBCA,$BDDAT ;READ BASE ADDR+6, SHOULD BE ZERO
373 003006 001401 BEQ TST6 ;;
374

```

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

```

(1)
(1) 003010 104006 ERROR 6 ;/MODULE FAULT DETECTED:
375 ;SHOULD HAVE READ BACK ZERO FROM
376 ;BASE ADDR+6.

```

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

```

378
379 ;:*****
(3) ;*TEST 6 *TEST THAT WE CAN SET TCS, TCS SETS CMD
(3) ;:*****
(2) 003012 000004 TST6: SCOPE
380
381 003014 005077 176326 CLR @IBS ;CLEAR CLR
382 003020 052777 000001 176320 BIS #BIT0,@IBS ;SET TCS.
383 003026 052737 002001 001124 BIS #BIT0!BIT10,$GDDAT ;EXPECT ONLY TCS AND CMD TO SET
384 003034 017737 176306 001126 MOV @IBS,$BDDAT ;READ THE IBS.
385 003042 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID TCS AND CMD SET?
386 003050 000402 BR 1$ ;YES - CONTINUE
387

```

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

```

(1)
(1) 003052 104003 ERROR 3 ;/MODULE FAULT DETECTED:

```

388 ;TCS AND/OR CMD FAILED TO SET

```
390 003054 000412      ::$$$$$$$$$$$^^^ ERROR ^^ $$$$$$$$$$$$  
391 BR TST7 ;;  
392 003056 042777 000001 176262 1$: BIC #BIT0,@IBS ;CLEAR TCS.  
393 003064 005037 001124 CLR $GDDAT ;EXPECT TCS AND CMD TO CLEAR  
394 003070 017737 176252 001126 MOV @IBS,$BDDAT ;READ IBS, DID THEY CLEAR?  
395 003076 001401 BEQ TST7 ;;  
396
```

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

```
(1)  
(1) 003100 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
397 ;TCS AND/OR CMD FAILED TO CLEAR.
```

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

```
399  
400 :*****  
(3) :*TEST 7 *TEST THAT EOP WILL SET  
(3) :*****  
(2) 003102 000004 TST7: SCOPE  
401  
402 003104 005077 176236 CLR @IBS ;CLEAR CSR.  
403 003110 052777 000002 176230 BIS #BIT1,@IBS ;SET EOP  
404 003116 012737 000002 001124 MOV #BIT1,$GDDAT ;EXPECT ONLY EOP TO SET.  
405 003124 017737 176216 001126 MOV @IBS,$BDDAT ;READ IBS  
406 003132 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID EOP SET?  
407 003140 001402 BEQ 1$ ;YES - CONTINUE  
408
```

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

```
(1)  
(1) 003142 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
409 ;EOP BIT SETTING ERROR.
```

```
411 003144 000412      ::$$$$$$$$$$$^^^ ERROR ^^ $$$$$$$$$$$$  
412 BR TST10 ;;  
413 003146 042777 000002 176172 1$: BIC #BIT1,@IBS ;CLEAR EOP  
414 003154 005037 001124 CLR $GDDAT ;EXPECT A ZERO CSR.  
415 003160 017737 176162 001126 MOV @IBS,$BDDAT ;READ IBS, IS IT CLEAR?  
416 003166 001401 BEQ TST10 ;;  
417
```

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

```
(?)  
(1) 003170 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
418 ;IBS FAILED TO CLEAR
```

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

```
420  
421 ::*****  
(3) : *TEST 10 *TEST THAT RE WILL SET + CLEAR  
(3) :*****  
(2) 003172 000004 TST10: SCOPE  
422  
423 003174 005077 176146 CLR @IBS ;CLEAR CSR.  
424 003200 052777 000004 176140 BIS #BIT02,@IBS ;SET REM  
425 003206 012737 000004 001124 MOV #BIT02,$GDDAT ;EXPECT ONLY REM TO SET.  
426 003214 017737 176126 001126 MOV @IBS,$BDDAT ;READ IBS.  
427 003222 023737 001126 001124 CMP $BDDAT,$GDDAT ;DID REM AND ONLY REM SET?  
428 003230 001402 BEQ 1$ ;YES - CONTINUE  
429
```

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

```
(1)  
(1) 003232 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
430 ;REM BIT SETTING ERROR.
```

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

```
432 003234 000412 BR TST11 ;:  
433  
434 003236 042777 000004 176102 1$: BIC #BIT02,@IBS ;CLEAR REM BIT.  
435 003244 005037 001124 CLR $GDDAT ;EXPECT ZERO CSR.  
436 003250 017737 176072 001126 MOV @IBS,$BDDAT ;READ IBS - IS IT CLEAR?  
437 003256 001401 BEQ TST11 ;:  
438
```

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

```
(1)  
(1) 003260 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
439 ;IBS FAILED TO CLEAR.
```

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

```
441  
442 ::*****  
(3) : *TEST 11 *TEST THAT IBC WILL SET AND CLEAR  
(3) :*****  
(2) 003262 000004 TST11: SCOPE  
443  
444 003264 005077 176056 CLR @IBS ;CLEAR CSR.  
445 003270 052777 000010 176050 BIS #BIT03,@IBS ;SET IBC  
446 003276 012737 000010 001124 MOV #BIT03,$GDDAT ;EXPECT ONLY IBC TO BE SET  
447 003304 017737 176036 001126 MOV @IBS,$BDDAT ;READ IBS.  
448 003312 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID IBS SET?  
449 003320 001414 BEQ 1$ ;YES CONTINUE  
450 003322 012777 000010 176016 MOV #BIT03,@IBS ;TRY SETTING IBC AGAIN.  
451 003330 017737 176012 001126 MOV @IBS,$BDDAT ;MEMORY REFRESH MIGHT HAVE  
452 003336 023737 001124 001126 CMP $GDDAT,$BDDAT ;GOT IN THE WAY.  
453 003344 001402 BEQ 1$  
454
```



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

(1) 003346 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
455 ;IBS BIT SETTING ERROR.

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

457 003350 000416 BR TST12 ;:  
458  
459 003352 004537 007122 1\$: JSR R5,DEL50 ;DELAY 150 US.  
460 003356 000006 .WORD 6  
461  
462 003360 012737 002001 001124 MOV #BIT10!BIT0,\$GDDAT ;EXP CMD AND TCS.  
463 003366 017737 175754 001126 MOV @IBS,\$BDDAT ;READ IBS - IS IT CLEAR?  
464 003374 023737 001124 001126 CMP \$GDDAT,\$BDDAT  
465 003402 001401 BEQ TST12 ;:  
466

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

(1) 003404 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
467 ;IBS NOT CLEAR AFTER IBC

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

469  
(3) \*\*\*\*\*  
(3) \*TEST 12 \*TEST THAT TON (BIT05) AND TKR SET AND CLEAR  
TST12: SCOPE  
(2) 003406 000004  
470  
471 003410 005077 175732 CLR @IBS ;CLEAR THE CSR.  
472 003414 052777 000040 175724 BIS #BIT5,@IBS ;SET TON.  
473 003422 012737 001040 001124 MOV #BIT5!BIT9,\$GDDAT ;EXPECT ONLY TON AND TKR TO SET.  
474 003430 017737 175712 001126 MOV @IBS,\$BDDAT ;READ CSR.  
475 003436 023737 001124 001126 CMP \$GDDAT,\$BDDAT ;DID THEY BOTH SET?  
476 003444 001402 BEQ 1\$ ;YES - CONTINUE.  
477  
478

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

(1) 003446 104003 ERROR 3 ;/MODULE FAULT DETECTED:  
479 ;ERROR IN SETTING TON BIT.

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

481 003450 000412 BR TST13 ;:  
482  
483 003452 042777 000040 175666 1\$: BIC #BIT5,@IBS ;WHEN TON CLEARED, TKR SHOULD CLEAR.  
484 003460 005037 001124 CLR \$GDDAT ;EXPECT ZERO CSR.  
485 003464 017737 175656 001126 MOV @IBS,\$BDDAT ;DID IT CLEAR?  
486 003472 001401 BEQ TST13 ;:  
487

```

(1)
(1) 003474 104003          ERROR 3          :/MODULE FAULT DETECTED:
488                                     :CSR FAILED TO CLEAR.

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

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

490
491
(3)
(3)
(2) 003476 000004          :*****
TST13: SCOPE                :*TEST 13          *MAKE SURE WE CAN SET AND CLEAR BIT06 (IE)
492
493
(1) 003500 012746 000340          :/PR
(1) 003504 012746 003512          :/SET CPU PRIORITY ON RETURN
(1) 003510 000002          :/SHOW RETURN ADDRESS
(1) 003512          64$:          RTI          :/CAUSE A RETURN (PUTS NEW STATUS
                                     :/IN STATUS REG.)

494
495 003512 005077 175630          CLR @IBS          :CLEAR CSR.
496
497 003516 052777 000100 175622  BIS #BIT6,@IBS    :SET IE.
498 003524 012737 000100 001124  MOV #BIT6,$GDDAT  :EXPECT ONLY BIT 6 TO SET.
499 003532 017737 175610 001126  MOV @IBS,$BDDAT   :READ IBS.
500 003540 023737 001124 001126  CMP $GDDAT,$BDDAT :DID IE SET?
501 003546 001402          BEQ 1$           :YES - CONTINUE.
502

```

```

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

(1)
(1) 003550 104003          ERROR 3          :/MODULE FAULT DETECTED:
503                                     :ERROR IN SETTING IE BIT.

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

505 003552 000412          BR TST14          ;;
506
507 003554 005037 001124          1$:          CLR $GDDAT        :EXPECT ZERO CSR AFTER.
508 003560 042777 000100 175560  BIC #BIT6,@IBS   :IE IS CLEARED.
509 003566 017737 175554 001126  MOV @IBS,$BDDAT  :READ CSR - IS IT CLEAR?
510 003574 001401          BEQ TST14          ;;
511

```

```

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

(1)
(1) 003576 104003          ERROR 3          :/MODULE FAULT DETECTED:
512                                     :FAILED TO CLEAR CSR.

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

514
515
(3)
(3)
:*****
:*TEST 14          *TEST THAT BIT 7 (ACC) CAN BE SET AND CLEARED
:*****

```

```

(2) 003600 000004          TST14: SCOPE
516
517 003602 005077 175540      CLR @IBS          ;CLEAR CSR.
518 003606 052777 000200 175532  BIS #BIT7,@IBS   ;SET ACC.
519 003614 012737 000200 001124  MOV #BIT7,$GDDAT ;EXPECT ONLY ACC TO SET.
520 003622 017737 175520 001126  MOV @IBS,$BDDAT  ;READ IBS.
521 003630 023737 001124 001126  CMP $GDDAT,$BDDAT ;DID ACC SET?
522 003636 001402          BEQ 1$           ;YES - CONTINUE.
523

```

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

```

(1)
(1) 003640 104003          ERROR 3           ;/MODULE FAULT DETECTED:
524                                     ;FAILURE IN SETTING BIT 7 (ACC).

```

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

```

526 003642 000412          BR TST15          ;;
527
528 003644 042777 000200 175474 1$: BIC #BIT7,@IBS   ;TRY CLEARING ACC.
529 003652 005037 001124          CLR $GDDAT       ;EXPECT ZERO CSR.
530 003656 017737 175464 001126  MOV @IBS,$BDDAT ;READ IBS, IS IT CLEAR?
531 003664 001401          BEQ TST15        ;;
532

```

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

```

(1)
(1) 003666 104003          ERROR 3           ;/MODULE FAULT DETECTED:
533                                     ;IBS FAILED TO CLEAR.

```

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

```

535
562
563
(5)
(4)
(4)
(3) 003670 000004          *****
TST15: SCOPE          ;*TEST 15 *TEST THAT IBD BIT 0 CAN BE SET + CLEARED
(1)
(1)

```

```

(1) 003672 012777 000060 175446      MOV #BIT4!BIT5,@IBS ;/MACRO BDT
(1) 003700 012737 000001 001124      MOV #BIT0,$GDDAT   ;/SET TON AND LON.
(1) 003706 013777 001124 175434      MOV $GDDAT,@IBD    ;/WE'RE GONNA TEST BIT 0.
(1)                                     ;/SET THE BIT.
(1) 003714 117737 175430 001126      MOV @IBD,$BDDAT    ;/READ THE IBD.
(1) 003722 123737 001124 001126      CMPB $GDDAT,$BDDAT ;/DID IT GET THRU OK?
(1) 003730 001402          BEQ 1$           ;/YES - CONTINUE.
(1)
(2)

```

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

(2)



(2) 003732 104004 ERROR 4 ;/MODULE FAULT DETECTED:  
(1) ;/ERROR IN SETTING IBD BIT 0.

(3) 003734 000412  
(1) 003736 005037 001124 1\$: BR TST16 ;/;  
(1) 003742 042777 000001 175400 CLR \$GDDAT ;/EXPECT ZERO IBD WHEN  
(1) 003750 117737 175374 001126 BIC #BIT0,@IBD ;/BIT 0 IS CLEARED.  
(3) 003756 001401 MOVB @IBD,\$BDDAT ;/READ IBD, IS IT CLEAR?  
(2) BEQ TST16 ;/;

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

(2) 003760 104004 ERROR 4 ;/MODULE FAULT DETECTED:  
(1) ;/FAILED TO CLEAR IBD.

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

(1) 564  
(5) ;/\*\*\*\*\*  
(4) ;/\*TEST 16 \*TEST THAT IBD BIT 1 CAN BET SET + CLEARED  
(4) ;/\*\*\*\*\*  
(3) 003762 000004 TST16: SCOPE

(1) ;/MACRO BDT  
(1) 003764 012777 000060 175354 MOV #BIT4!BIT5,@IBS ;/SET TON AND LON.  
(1) 003772 012737 000002 001124 MOV #BIT1,\$GDDAT ;/WE'RE GONNA TEST BIT 1.  
(1) 004000 013777 001124 175342 MOV \$GDDAT,@IBD ;/SET THE BIT.  
(1) 004006 117737 175336 001126 MOVB @IBD,\$BDDAT ;/READ THE IBD.  
(1) 004014 123737 001124 001126 CMPB \$GDDAT,\$BDDAT ;/DID IT GET THRU OK?  
(1) 004022 001402 BEQ 1\$ ;/YES - CONTINUE.

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

(2) 004024 104004 ERROR 4 ;/MODULE FAULT DETECTED:  
(1) ;/ERROR IN SETTING IBD BIT 1.

(3) 004026 000412  
(1) 004030 005037 001124 1\$: BR TST17 ;/;  
(1) 004034 042777 000002 175306 CLR \$GDDAT ;/EXPECT ZERO IBD WHEN  
(1) 004042 117737 175302 001126 BIC #BIT1,@IBD ;/BIT 1 IS CLEARED.  
(3) 004050 001401 MOVB @IBD,\$BDDAT ;/READ IBD, IS IT CLEAR?  
(2) BEQ TST17 ;/;

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

(2) 004052 104004 ERROR 4 ;/MODULE FAULT DETECTED:

(1) ;FAILED TO CLEAR IBD.

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

(1) 565  
(5) :\*\*\*\*\*  
(4) :\*TEST 17 \*TEST THAT IBD BIT 2 CAN BET SET + CLEARED  
(4) :\*\*\*\*\*  
(3) 004054 000004 TST17: SCOPE

(1) (1) 004056 012777 000060 175262 MOV #BIT4!BIT5,@IBS ;/MACRO BDT  
(1) 004064 012737 000004 001124 MOV #BIT2,\$GDDAT ;/SET TON AND LON.  
(1) 004072 013777 001124 175250 MOV \$GDDAT,@IBD ;/WE'RE GONNA TEST BIT 2.  
(1) ;/SET THE BIT.  
(1) 004100 117737 175244 001126 MOVB @IBD,\$BDDAT ;/READ THE IBD.  
(1) 004106 123737 001124 001126 CMPB \$GDDAT,\$BDDAT ;/DID IT GET THRU OK?  
(1) 004114 001402 BEQ 1\$ ;/YES - CONTINUE.

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

(2) 004116 104004 ERROR 4 ;/MODULE FAULT DETECTED:  
(1) ;/ERROR IN SETTING IBD BIT 2.

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

(3) 004120 000412 BR TST20 ;/;  
(1) 004122 005037 001124 1\$: CLR \$GDDAT ;/EXPECT ZERO IBD WHEN  
(1) 004126 042777 000004 175214 BIC #BIT2,@IBD ;/BIT 2 IS CLEARED.  
(1) 004134 117737 175210 001126 MOVB @IBD,\$BDDAT ;/READ IBD, IS IT CLEAR?  
(3) 004142 001401 BEQ TST20 ;/;  
(2)

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

(2) 004144 104004 ERROR 4 ;/MODULE FAULT DETECTED:  
(1) ;/FAILED TO CLEAR IBD.

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

(1) 566  
(5) :\*\*\*\*\*  
(4) :\*TEST 20 \*TEST THAT IBD BIT 3 CAN BET SET + CLEARED  
(4) :\*\*\*\*\*  
(3) 004146 000004 TST20: SCOPE

(1) (1) 004150 012777 000060 175170 MOV #BIT4!BIT5,@IBS ;/MACRO BDT  
(1) 004156 012737 000010 001124 MOV #BIT3,\$GDDAT ;/SET TON AND LON.  
(1) 004164 013777 001124 175156 MOV \$GDDAT,@IBD ;/WE'RE GONNA TEST BIT 3.  
(1) ;/SET THE BIT.

```

(1) 004172 117737 175152 001126      MOVB   @IBD,$BDDAT    ;/READ THE IBD.
(1) 004200 123737 001124 001126      CMPB   $GDDAT,$BDDAT ;/DID IT GET THRU OK?
(1) 004206 001402                      BEQ    1$             ;/YES - CONTINUE.
(1)
(2)

```

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

```

(2) 004210 104004                      ERROR  4              ;/MODULE FAULT DETECTED:
(1)                                           ;/ERROR IN SETTING IBD BIT 3.

```

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

```

(3) 004212 000412                      BR     TST21          ;/
(1) 004214 005037 001124 175122 1$:  CLR    $GDDAT        ;/EXPECT ZERO IBD WHEN
(1) 004220 042777 000010 175122      BIC    #BIT3,@IBD    ;/BIT 3 IS CLEARED.
(1) 004226 117737 175116 001126      MOVB   @IBD,$BDDAT   ;/READ IBD, IS IT CLEAR?
(3) 004234 001401                      BEQ    TST21          ;/
(2)

```

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

```

(2) 004236 104004                      ERROR  4              ;/MODULE FAULT DETECTED:
(1)                                           ;/FAILED TO CLEAR IBD.

```

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

```

(1) 567
(5)
(4) :::*****
(4) *TEST 21 *TEST THAT IBD BIT 4 CAN BET SET + CLEARED
(3) 004240 000004 TST21: SCOPE
(1)

```

```

(1) 004242 012777 000060 175076      MOV    #BIT4!BIT5,@IBS ;/MACRO BDT
(1) 004250 012737 000020 001124      MOV    #BIT4,$GDDAT    ;/SET TON AND LON.
(1) 004256 013777 001124 175064      MOV    $GDDAT,@IBD     ;/WE'RE GONNA TEST BIT 4.
(1)                                           ;/SET THE BIT.
(1) 004264 117737 175060 001126      MOVB   @IBD,$BDDAT    ;/READ THE IBD.
(1) 004272 123737 001124 001126      CMPB   $GDDAT,$BDDAT  ;/DID IT GET THRU OK?
(1) 004300 001402                      BEQ    1$             ;/YES - CONTINUE.
(1)
(2)

```

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

```

(2) 004302 104004                      ERROR  4              ;/MODULE FAULT DETECTED:
(1)                                           ;/ERROR IN SETTING IBD BIT 4.

```

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

```

(3) 004304 000412                      BR     TST22          ;/
(1) 004306 005037 001124 1$:  CLR    $GDDAT        ;/EXPECT ZERO IBD WHEN

```



```

(1) 004312 042777 000020 175030      BIC    #BIT4,@IBD      ;/BIT 4 IS CLEARED.
(1) 004320 117737 175024 001126      MOVB   @IBD,$BDDAT    ;/READ IBD, IS IT CLEAR?
(3) 004326 001401                      BEQ    TST22          ;;
(2)

```

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

```

(2) 004330 104004                      ERROR  4              ;/MODULE FAULT DETECTED:
(1)                                           ;/FAILED TO CLEAR IBD.

```

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

(1)  
568

```

(5) :::*****
(4) :::*TEST 22      *TEST THAT IBD BIT 5 CAN BET SET + CLEARED
(4) :::*****
(3) 004332 000004      TST22:  SCOPE
(1)

```

```

(1) 004334 012777 000060 175004      MOV    #BIT4!BIT5,@IBS ;/MACRO BDT
(1) 004342 012737 000040 001124      MOV    #BIT5,$GDDAT    ;/SET TON AND LON.
(1) 004350 013777 001124 174772      MOV    $GDDAT,@IBD     ;/WE'RE GONNA TEST BIT 5.
(1)                                           ;/SET THE BIT.
(1) 004356 117737 174766 001126      MOVB   @IBD,$BDDAT    ;/READ THE IBD.
(1) 004364 123737 001124 001126      CMPB   $GDDAT,$BDDAT  ;/DID IT GET THRU OK?
(1) 004372 001402                      BEQ    1$             ;/YES - CONTINUE.
(1)
(2)

```

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

```

(2) 004374 104004                      ERROR  4              ;/MODULE FAULT DETECTED:
(1)                                           ;/ERROR IN SETTING IBD BIT 5.

```

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

```

(3) 004376 000412                      BR     TST23          ;;
(1) 004400 005037 001124 1$:          CLR    $GDDAT        ;/EXPECT ZERO IBD WHEN
(1) 004404 042777 000040 174736      BIC    #BIT5,@IBD     ;/BIT 5 IS CLEARED.
(1) 004412 117737 174732 001126      MOVB   @IBD,$BDDAT    ;/READ IBD, IS IT CLEAR?
(3) 004420 001401                      BEQ    TST23          ;;
(2)

```

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

```

(2) 004422 104004                      ERROR  4              ;/MODULE FAULT DETECTED:
(1)                                           ;/FAILED TO CLEAR IBD.

```

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

(1)  
569  
(5)

:::\*\*\*\*\*

```
(4) ;*TEST 23 *TEST THAT IBD BIT 6 CAN BET SET + CLEARED
(4) :*****
(3) 004424 000004 TST23: SCOPE
(1)
(1) ;/MACRO BDT
(1) 004426 012777 000060 174712 MOV #BIT4!BITS,@IBS ;/SET TON AND LON.
(1) 004434 012737 000100 001124 MOV #BIT6,$GDDAT ;/WE'RE GONNA TEST BIT 6.
(1) 004442 013777 001124 174700 MOV $GDDAT,@IBD ;/SET THE BIT.
(1)
(1) 004450 117737 174674 001126 MOVB @IBD,$BDDAT ;/READ THE IBD.
(1) 004456 123737 001124 001126 CMPB $GDDAT,$BDDAT ;/DID IT GET THRU OK?
(1) 004464 001402 BEQ 1$ ;/YES - CONTINUE.
(1)
(2)
```

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

```
(2)
(2) 004466 104004 ERROR 4 ;/MODULE FAULT DETECTED:
(1) ;/ERROR IN SETTING IBD BIT 6.
```

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

```
(3) 004470 000412 BR TST24 ;/
(1) 004472 005037 001124 1$: CLR $GDDAT ;/EXPECT ZERO IBD WHEN
(1) 004476 042777 000100 174644 BIC #BIT6,@IBD ;/BIT 6 IS CLEARED.
(1) 004504 117737 174640 001126 MOVB @IBD,$BDDAT ;/READ IBD, IS IT CLEAR?
(3) 004512 001401 BEQ TST24 ;/
(2)
```

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

```
(2)
(2) 004514 104004 ERROR 4 ;/MODULE FAULT DETECTED:
(1) ;/FAILED TO CLEAR IBD.
```

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

```
(1)
570
(5) :*****
(4) ;*TEST 24 *TEST THAT IBD BIT 7 CAN BET SET + CLEARED
(4) :*****
(3) 004516 000004 TST24: SCOPE
(1)
(1) ;/MACRO BDT
(1) 004520 012777 000060 174620 MOV #BIT4!BITS,@IBS ;/SET TON AND LON.
(1) 004526 012737 000200 001124 MOV #BIT7,$GDDAT ;/WE'RE GONNA TEST BIT 7.
(1) 004534 013777 001124 174606 MOV $GDDAT,@IBD ;/SET THE BIT.
(1)
(1) 004542 117737 174602 001126 MOVB @IBD,$BDDAT ;/READ THE IBD.
(1) 004550 123737 001124 001126 CMPB $GDDAT,$BDDAT ;/DID IT GET THRU OK?
(1) 004556 001402 BEQ 1$ ;/YES - CONTINUE.
(1)
(2)
```

```

(2)
(2) 004560 104004          ERROR 4          :/MODULE FAULT DETECTED:
(1)                                     :/ERROR IN SETTING IBD BIT 7.

```

```

(3) 004562 000412          BR      TST25          :
(1) 004564 005037 001124 1$: CLR      $GDDAT        :/EXPECT ZERO IBD WHEN
(1) 004570 042777 000200 174552 BIC     #BIT7,@IBD    :/BIT 7 IS CLEARED.
(1) 004576 117737 174546 001126 MOVB   @IBD,$BDDAT    :/READ IBD, IS IT CLEAR?
(3) 004604 001401          BEQ     TST25          :
(2)

```

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

```

(2)
(2) 004606 104004          ERROR 4          :/MODULE FAULT DETECTED:
(1)                                     :/FAILED TO CLEAR IBD.

```

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

```

(1)
571
572
(3)
(3)
(2) 004610 000004          TST25: SCOPE
573
574 004612 005077 174530          CLR     @IBS          :CLEAR CSR
575 004616 112777 000252 174524 MOVB   #252,@IBD     :TRY XFERRING DATA
576 004624 005037 001124          CLR     $GDDAT        :NO DATA SHOULD XFERR
577 004630 117737 174514 001126 MOVB   @IBD,$BDDAT    :READ BUFFER REG.
578 004636 001401          BEQ     TST26          :
579

```

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

```

(1)
(1) 004640 104002          ERROR 2          :/MODULE FAULT DETECTED:
580                                     :/DATA WAS XFERRED THROUGH IBD
581                                     :/EVEN THOUGH TON AND LON CLEARED.
582                                     :/SIGNAL 'ENB XFER L' PROBABLY
583                                     :/STUCK LOW.

```

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

```

585
586
(3)
(3)
(2) 004642 000004          TST26: SCOPE
587
588 004644 005077 174476          CLR     @IBS          :CLEAR CSR.
589 004650 005077 174474          CLR     @IBD          :CLEAR DATA REG.
590 004654 032777 000400 174466 BIT     #BIT8,@IBD    :IS DAC SET?

```



591 004662 001002 BNE 4\$ ;YES (GOOD)  
592

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

(1)  
(1) 004664 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
593 ;DAC NOT SET.

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

595 004666 000437 BR TST27 ;:  
596 004670  
597 004670 052777 000260 174450 4\$: BIS #BIT5!BIT4!BIT7,@IBS ;SET TON AND LON  
598 004676 012777 000252 174444 MOV #252,@IBD ;PUT DATA IN IBD.  
599 004704 017737 174440 001126 MOV @IBD,\$BDDAT ;READ IBD.  
600 004712 032737 001000 001126 BIT #BIT9,\$BDDAT ;DID DAV SET?  
601 004720 001002 BNE 1\$ ;YES - CONTINUE.  
602

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

(1)  
(1) 004722 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
603 ;DAV FAILED TO SET.

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

605 004724 000420 BR TST27 ;:  
606  
607 004726 012777 000060 174412 1\$: MOV #BIT4!BIT5,@IBS ;CLEAR ACC.  
608 004734 105777 174410 2\$: TSTB @IBD ;READ LOW BYTE OF IBD.  
609 004740 032777 000400 174402 BIT #BIT8,@IBD ;DID DAC CLEAR?  
610 004746 001402 BEQ 3\$ ;YES - CONTINUE.  
611

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

(1)  
(1) 004750 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
612 ;DAC FAILED TO CLEAR.

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

614 004752 000405 BR TST27 ;:  
615 004754 032777 001000 174366 3\$: BIT #BIT9,@IBD ;DID DAV CLEAR?  
616 004762 001401 BEQ TST27 ;:  
617

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

(1)  
(1) 004764 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
618 ;DAV FAILED TO CLEAR.

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

661

```
662  
(1) ;/MACRO -SIGC-  
(5) ::*****  
(4) :*TEST 27 *TEST THAT REN SETS WHEN REM SETS, ALSO TEST CLEAR  
(4) :*****  
(3) 004766 000004 TST27: SCOPE  
(1)  
(1) 004770 005077 174352 CLR @IBS ;/CLEAR CSR.  
(1) 004774 052777 000004 174344 BIS #BIT2,@IBS ;/SET REM, SHOULD SET REN.  
(1) 005002 032777 010000 174340 BIT #BIT12,@IBD ;/DID REN SET?  
(1) 005010 001011 BNE 1$ ;/YES - LETS TRY CLEARING IT.  
(1) 005012 052777 000004 174326 BIS #BIT2,@IBS ;/SET REM, MEMORY  
(1) ;/REFRESH COULD HAVE  
(1) ;/INTERRUPTED US.  
(1) 005020 032777 010000 174322 BIT #BIT12,@IBD ;/DID REN SET THIS TIME?  
(1) 005026 001002 BNE 1$  
(2)
```

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

```
(2)  
(2) 005030 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
(1) ;/REN FILED TO SET WHEN REM SET.
```

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

```
(3) 005032 000410 BR TST30 ;/  
(1) 005034 042777 000004 174304 1$: BIC #BIT2,@IBS ;/CLEAR REM, SHOULD CLEAR REN.  
(1) 005042 032777 010000 174300 BIT #BIT12,@IBD ;/DID REN CLEAR?  
(3) 005050 001401 BEQ TST30 ;/  
(2)
```

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

```
(2)  
(2) 005052 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
(1) ;/REN FAILED TO CLEAR.
```

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

```
(1)  
663 ;/MACRO -SIGC-  
664  
(1)
```

```
(5) ::*****  
(4) :*TEST 30 *TEST THAT IFC SETS WHEN IBS SETS, ALSO TEST CLEAR  
(4) :*****  
(3) 005054 000004 TST30: SCOPE  
(1)  
(1) 005056 005077 174264 CLR @IBS ;/CLEAR CSR.  
(1) 005062 052777 000010 174256 BIS #BIT3,@IBS ;/SET IBS, SHOULD SET IFC.  
(1) 005070 032777 020000 174252 BIT #BIT13,@IBD ;/DID IFC SET?  
(1) 005076 001011 BNE 1$ ;/YES - LETS TRY CLEARING IT.  
(1) 005100 052777 000010 174240 BIS #BIT3,@IBS ;/SET IBS, MEMORY  
(1) ;/REFRESH COULD HAVE  
(1) ;/INTERRUPTED US.
```

CVIBB-A MACY11 27(654)  
CVIBBA.P11 T30

19-SEP-78 11:35 PAGE 1-27  
\*TEST THAT IFC SETS WHEN IBS SETS, ALSO TEST CLEAR

SEQ 0042

(1) 005106 032777 020000 174234 BIT #BIT13,@IBD ;/DID IFC SET THIS TIME?  
(1) 005114 001002 BNE 1\$  
(2)

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

(2) 005116 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
(1) ;/IFC FILED TO SET WHEN IBS SET.

:::\$\$\$\$\$\$\$\$\$\*\*\* ERROR \*\*\*\$\$\$\$\$\$\$\$\$  
BR TST31 ;:

(3) 005120 000411  
(1) 005122 032777 000010 174216 1\$: BIT #BIT3,@IBS ;/WAIT FOR IBS TO CLEAR.  
(1) 005130 001374 BNE 1\$  
(1) 005132 032777 020000 174210 BIT #BIT13,@IBD ;/IBS CLEAR, DID IFC CLEAR?  
(3) 005140 001401 BEQ TST31 ;:  
(2)

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

(2) 005142 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
(1) ;/IFC FAILED TO CLEAR.

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

(1) 665 ;/MACRO -SIGC-  
(1) 666

:::\*\*\*\*\*  
:\*TEST 31 \*TEST THAT ATN SETS WHEN TCS SETS, ALSO TEST CLEAR  
:::\*\*\*\*\*  
TST31: SCOPE

(3) 005144 000004  
(1) 005146 005077 174174 CLR @IBS ;/CLEAR CSR.  
(1) 005152 052777 000001 174166 BIS #BIT0,@IBS ;/SET TCS, SHOULD SET ATN.  
(1) 005160 032777 040000 174162 BIT #BIT14,@IBD ;/DID ATN SET?  
(1) 005166 001011 BNE 1\$ ;/YES - LETS TRY CLEARING IT.  
(1) 005170 052777 000001 174150 BIS #BIT0,@IBS ;/SET TCS, MEMORY  
(1) ;/REFRESH COULD HAVE  
(1) ;/INTERRUPTED US.  
(1) 005176 032777 040000 174144 BIT #BIT14,@IBD ;/DID ATN SET THIS TIME?  
(1) 005204 001002 BNE 1\$  
(2)

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

(2) 005206 104002 ERROR 2 ;/MODULE FAULT DETECTED:  
(1) ;/ATN FILED TO SET WHEN TCS SET.

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



```

(3) 005210 000410 BR TST32 ;;
(1) 005212 042777 000001 174126 1$: BIC #BIT0,@IBS ;:/CLEAR TCS, SHOULD CLEAR ATN.
(1) 005220 032777 040000 174122 BIT #BIT14,@IBD ;:/DID ATN CLEAR?
(3) 005226 001401 BEQ TST32 ;;
(2)

```

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

```

(2) 005230 104002 ERROR 2 ;/MODULE FAULT DETECTED:
(1) ;/ATN FAILED TO CLEAR.

```

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

```

(1) 667 ;/MACRO -SIGC-
(1) 668

```

```

(5) :*****
(4) :*TEST 32 *TEST THAT EOI SETS WHEN EOP SETS, ALSO TEST CLEAR
(4) :*****
(3) 005232 000004 TST32: SCOPE

```

```

(1) 005234 005077 174106 CLR @IBS ;/CLEAR CSR.
(1) 005240 052777 000002 174100 BIS #BIT1,@IBS ;/SET EOP, SHOULD SET EOI.
(1) 005246 032777 100000 174074 BIT #BIT15,@IBD ;/DID EOI SET?
(1) 005254 001011 BNE 1$ ;/YES - LETS TRY CLEARING IT.
(1) 005256 052777 000002 174062 BIS #BIT1,@IBS ;/SET EOP, MEMORY
(1) ;/REFRESH COULD HAVE
(1) ;/INTERRUPTED US.
(1) 005264 032777 100000 174056 BIT #BIT15,@IBD ;/DID EOI SET THIS TIME?
(1) 005272 001002 BNE 1$
(2)

```

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

```

(2) 005274 104002 ERROR 2 ;/MODULE FAULT DETECTED:
(1) ;/EOI FILED TO SET WHEN EOP SET.

```

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

```

(3) 005276 000410 BR TST33 ;;
(1) 005300 042777 000002 174040 1$: BIC #BIT1,@IBS ;/CLEAR EOP, SHOULD CLEAR EOI.
(1) 005306 032777 100000 174034 BIT #BIT15,@IBD ;/DID EOI CLEAR?
(3) 005314 001401 BEQ TST33 ;;
(2)

```

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

```

(2) 005316 104002 ERROR 2 ;/MODULE FAULT DETECTED:
(1) ;/EOI FAILED TO CLEAR.

```

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

(1)

```

669
670      ;*****
671      ;*TEST 33      *TEST THAT RFD SET WHEN CSR CLEAR,CLEAR WHEN ACC SET
672      ;*****
673      TST33: SCOPE
674      CLR      @IBS      ;CLEAR CSR.
675      BIT      #BIT10,@IBD ;DID RFD SET?
676      BNE      1$        ;YES CONTINUE.

```

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

```

(1)
(1) 005336 104002      ERROR 2      ;/MODULE FAULT DETECTED:
676      ;RFD FAILED TO SET.

```

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

```

678 005340 000410      BR      TST34      ;;
679
680 005342 052777 000200 173776 1$: BIS      #BIT7,@IBS      ;NOW SET ACC,RFD SHOULD CLEAR.
681 005350 032777 002000 173772      BIT      #BIT10,@IBD    ;DID IT CLEAR?
682 005356 001401      BEQ      TST34      ;;
683

```

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

```

(1)
(1) 005360 104002      ERROR 2      ;/MODULE FAULT DETECTED:
684      ;RFD FAILED TO CLEAR.

```

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

```

686
687
688      ;*****
689      ;*TEST 34      *TEST THAT WE CAN GENERATE AN ER2
690      ;*****
691      TST34: SCOPE
692      CLR      @IBS      ;CLEAR THE STATUS REG.
693      BIS      #BIT5!BIT0,@IBS ;SET TON; THIS SHOULD CAUSE AN
694      CLR      @IBD      ;ERROR SENCE NO LISTENERS ARE ON
695      BIT      #BIT14,@IBS ;AND WE SENT DATA TO THE BUS.
696      BNE      TST35      ;BUS.
697      ;DID ER2 SET?

```

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

```

(1)
(1) 005412 104001      ERROR 1      ;/MODULE FAULT DETECTED:
698      ;ER2 FAILED TO SET.

```

```
700  
701  
(3) :*****  
(3) :*TEST 35 *TEST THAT BUS INIT CLEARS ACC,TON,LON,REM,EIP,TCS  
(2) 005414 000004 :*****  
(1) 005416 012737 000005 001160 TST35: SCOPE  
702 MOV #5,$TIMES ;;DO 5 ITERATIONS  
703 005424 012777 000367 173714 MOV #367,@IBS ;SET ACC,TON,LON,REM,EOP, AND TCS.  
704 005432 000005 RESET ;ISSUE SYS INIT.  
705 005434 105777 173706 TSTB @IBS ;DID THEY ALL CLEAR?  
706 005440 001401 BEQ TST36 ;  
707
```

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

```
(1)  
(1) 005442 104001 ERROR 1 ;/MODULE FAULT DETECTED:  
708 ;BUS INIT FAILED TO CLEAR CSR.  
709
```

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

```
711  
712 :*****  
(3) :*TEST 36 *TEST IBC CLEARS ACC,TON,LON,REM AND EOP  
(3) :*****  
(2) 005444 000004 TST36: SCOPE  
(1) 005446 012737 000005 001160 MOV #5,$TIMES ;;DO 5 ITERATIONS  
713  
714 005454 012777 000266 173664 MOV #266,@IBS ;SET ACC,TON,LON,REM, AND EOP.  
715 005462 052777 000010 173656 BIS #BIT3,@IBS ;SET IBC, THIS SHOULD CLEAR ABOVE BITS.  
716 005470 032777 000010 173650 1$: BIT #BIT3,@IBS ;WAIT TILL IBC CLEARS  
717 005476 001374 BNE 1$ ;  
718 005500 032777 000266 173640 BIT #266,@IBS ;DID THEY CLEAR?  
719 005506 001401 BEQ TST37 ;  
720
```

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

```
(1)  
(1) 005510 104001 ERROR 1 ;/MODULE FAULT DETECTED:  
721 ;ACC,TON,LON,REM, AND/OR EOP  
722 ;FAILED TO CLEAR ON IBC
```

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

```
724  
725 :*****  
(3) :*TEST 37 *TEST THAT BUS INIT INDIRECTLY CLEARS IBD  
(3) :*****  
(2) 005512 000004 TST37: SCOPE  
(1) 005514 012737 000005 001160 MOV #5,$TIMES ;;DO 5 ITERATIONS  
726  
727 005522 012777 000260 173616 MOV #BIT7!BIT5!BIT4,@IBS ;SET ACC,TON, AND LON.  
728 005530 012777 000377 173612 MOV #377,@IBD ;LOAD IBD
```



```

729 005536 000005          RESET          ;ISSUE SYS INIT.
730 005540 105777 173604  TSTB @IBD      ;DID IT CLEAR?
731 005544 001401          BEQ TST40       ;;
732

```

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

```

(1)
(1) 005546 104002          ERROR 2          ;/MODULE FAULT DETECTED:
733                                         ;FAILED TO CLEAR LOW BYTE OF IBD ON
734                                         ;SYSTEM INIT.

```

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

```

736
737 .SBTTL
738 .SBTTL          INTERRUPT TESTS
739 .SBTTL
740

```

```

741 :::*****
(3) *TEST 40          *TEST THAT CMD CAN GENERATE AN INTERRUPT B
(3) :::*****
(2) 005550 000004          TST40: SCOPE

```

```

742
743 005552 005077 173570          CLR @IBS          ;CLEAR THE CSR.
744 005556 012777 000200 173622  MOV #200,@PRC
745 005564 012777 005622 173570  MOV #1$,@VECTC   ;SET UP INTERRUPT VECTOR
746 005572 052777 000101 173546  BIS #BIT0!BIT6,@IBS ;SET TCS, SHOULD CAUSE
747                                         ;/PR
(1) 005600 012746 000000          MOV #0,-(SP)      ;/SET CPU PRIORITY ON RETURN
(1) 005604 012746 005612          MOV #64$,-(SP)   ;/SHOW RETURN ADDRESS
(1) 005610 000002          RTI              ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 005612          64$:          ;/IN STATUS REG.)
748 005612 000240          NOP              ;CMD TO SET AND GIVE US AN
749 005614 000240          NOP              ;INTERRUPT.
750

```

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

```

(1)
(1) 005616 104001          ERROR 1          ;/MODULE FAULT DETECTED:
751                                         ;CMD FAILED TO GENERATE AN INTERRUPT.

```

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

```

753 005620 000402          BR 2$
754 005622          1$:          ADD #4,SP          ;/ADD #4 TO STACK POINTER.
(1) 005622 062706 000004          CLR @IBS          ;CLEAR INTERRUPT
755 005626 005077 173514          ;/-RESV-
756                                         ;/RESTORE VECTOR FOR
(1) 005632 013777 001406 173522  MOV PRC,@VECTC   ;/ILLEGAL INTRO.
(1) 005640 012777 004700 173540  MOV #4700,@PRC
757

```

```

758 :::*****
(3) *TEST 41          *TEST THAT TCR AND LNR CAN GENERATE INTERRUPTS
(3) :::*****

```

```

(2) 005646 000004          TST41: SCOPE
759 005650 012777 000200 173530  MOV      #200,@PRC
760 005656 012777 005742 173476  MOV      #1$,@VECTC ;SET UP INTERRUPT VECTOR FOR TKR INTERRUPT
761 005664 012777 000060 173454  MOV      #BIT4!BIT5,@IBS ;SET TON AND LON
762 005672 052777 000100 173446  BIS      #BIT6,@IBS ;ALLOW INTERRUPT
763                                     ;/PR
(1) 005700 012746 000000          MOV      #0,-(SP) ;/SET CPU PRIORITY ON RETURN
(1) 005704 012746 005712          MOV      #64$,-(SP) ;/SHOW RETURN ADDRESS
(1) 005710 000002          RTI      ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 005712                                     ;/IN STATUS REG.)
764 005712 000240          NOP
765 005714 000240          NOP
766

```

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

```

(1)
(1) 005716 104001          ERROR 1 ;/MODULE FAULT DETECTED:
767                                     ;FAILED TO GENERATE A TKR INTERRUPT.

```

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

```

769 005720 005077 173422          CLR      @IBS ;CLR CSR
770                                     ;/-RESV-
(1) 005724 013777 001406 173430  MOV      PRC,@VECTC ;/RESTORE VECTOR FOR
(1) 005732 012777 004700 173446  MOV      #4700,@PRC ;/ILLEGAL INTRO.
771 005740 000443          BR      TST42 ;:
772
773 005742                                     1$:
(1) 005742 062706 000004          ADD      #4,SP ;/ADD #4 TO STACK POINTER.
774                                     ;/-RESV-
(1) 005746 013777 001406 173406  MOV      PRC,@VECTC ;/RESTORE VECTOR FOR
(1) 005754 012777 004700 173424  MOV      #4700,@PRC ;/ILLEGAL INTRO.
775 005762 012777 000200 173420  MOV      #200,@PRD
776 005770 012777 006024 173366  MOV      #2$,@VECTD ;SET UP FOR LNR INTERRUPT.
777                                     ;/PR
(1) 005776 012746 000000          MOV      #0,-(SP) ;/SET CPU PRIORITY ON RETURN
(1) 006002 012746 006010          MOV      #65$,-(SP) ;/SHOW RETURN ADDRESS
(1) 006006 000002          RTI      ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006010                                     ;/IN STATUS REG.)
778 006010 105277 173334          INCB    @IBD ;SEND DATA - CLRS TKR SETS LNR
779                                     ;FOR INTERRUPT
780 NOP
781 NOP
782

```

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

```

(1)
(1) 006020 104001          ERROR 1 ;/MODULE FAULT DETECTED:
783                                     ;FAILED TO GENERATE LNR INTERRUPT

```

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

```

785 006022 000402          BR      3$
786

```

```

787 006024          2$:
(1) 006024 062706 000004          ADD    #4,SP          ;/ADD #4 TO STACK POINTER.
788 006030 005077 173312          CLR    @IBS          ;CLEAR THE STATUS REG.
789                                     ;/-RESV-
(1) 006034 013777 001410 173322          MOV    PRD,@VECTD ;/RESTORE VECTOR FOR
(1) 006042 012777 004700 173340          MOV    #4700,@PRD ;/ILLEGAL INTRO.
790
791                                     ;:*****
(3)                                     ;*TEST 42          *TEST THAT ER2 CAN GENERATE AN INTERRUPT
(3)                                     ;:*****
(2) 006050 000004          TST42: SCOPE
792
793 006052 005077 173270          CLR    @IBS          ;START WITH CSR CLEAR
794 006056 012777 000200 173316          MOV    #200,@PRA
795 006064 012777 006142 173264          MOV    #1$,@VECTA   ;SET UP INTERRUPT VECTOR
796                                     ;/PR
(1) 006072 012746 000200          MOV    #200,-(SP)   ;/SET CPU PRIORITY ON RETURN
(1) 006076 012746 006104          MOV    #64$,-(SP)  ;/SHOW RETURN ADDRESS
(1) 006102 000002          RTI          ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006104                                     ;/IN STATUS REG.)
797 006104 052777 000140 173234          BIS    #BIT5!BIT6,@IBS ;SET TON - NO LISTNERS ON
798 006112 105077 173232          CLR    @IBD          ;BUS BUT DATA PUT ON
799 006116 000240          NOP          ;BUS - THEREFORE AN INTERRUPT
800 006120 000240          NOP          ;SHOULD BE POSTED.
801                                     ;/PR
(1) 006122 012746 000000          MOV    #0,-(SP)    ;/SET CPU PRIORITY ON RETURN
(1) 006126 012746 006134          MOV    #65$,-(SP)  ;/SHOW RETURN ADDRESS
(1) 006132 000002          RTI          ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006134                                     ;/IN STATUS REG.)
802 006134 000240          NOP
803
804

```

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

```

(1)
(1) 006136 104001          ERROR 1          ;/MODULE FAULT DETECTED:
805                                     ;FAILED TO INTERRUPT ON ERROR2

```

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

```

807 006140 000402          BR     2$
808 006142
(1) 006142 062706 000004          1$: ADD    #4,SP          ;/ADD #4 TO STACK POINTER.
809 006146 005077 173174          2$: CLR    @IBS          ;CLEAR CSR
810                                     ;/-RESV-
(1) 006152 013777 001402 173176          MOV    PRA,@VECTA ;/RESTORE VECTOR FOR
(1) 006160 012777 004700 173214          MOV    #4700,@PRA ;/ILLEGAL INTRO.

```

```

811
812          .SBTTL
813          .SBTTL SECOND MODULE TESTS
814          .SBTTL
815

```

:::\*\*\*\*\*
\*TEST 43 \*TEST THAT MODULE PASSES 'BIAKI'

(3)



(3)  
(2) 006166 000004

::\*\*\*\*\*  
TST43: SCOPE

817  
827  
828

(1) ;\*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A  
(1) ;\*SECOUND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET.  
(1) ;\*ADDRESS OF THE SECOUND MODULE IS IN LOCATION 'IBS2' VECTOR  
(1) ;\*ADDRESS IS IN LOCATION 'VECTA2'. THE SECOUND IBV-11 SHOULD BE ELECTRICALLY SEC  
(1) ;\*TO INHIBIT THE USE OF TESTING WITH A SECOUND MODULE, MAKE  
(1) ;\*LOCATION '\$CDW1' ZERO.  
(1) ;\*

829  
830 006170 005737 001254  
831 006174 001002  
832 006176 000137 007000  
833 006202

3\$: TST \$CDW1 ;TESTING WITH  
BNE 3\$ ;SECOUND IBV11?  
JMP EOP ;NO-END PASS.

834  
835 006202 005077 173140  
836 006206 005077 173154  
837 006212 012777 000200 173176  
838 006220 012777 006256 173150  
839

CLR @IBS ;CLEAR CSR.  
CLR @IBS2 ;CLEAR SECOUND MODULE.  
MOV #200,@PRC2  
MOV #1\$,@VECTC2 ;SET UP VECTOR ADDR.

(1) 006226 012746 000000  
(1) 006232 012746 006240  
(1) 006236 000002  
(1) 006240

64\$: MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN  
MOV #64\$,-(SP) ;/SHOW RETURN ADDRESS  
RTI ;/CAUSE A RETURN (PUTS NEW STATUS  
;/IN STATUS REG.)

840 006240 012777 000140 173120  
841 006246 000240  
842 006250 000240  
843

MOV #BIT6:BIT5,@IBS2 ;SET INTR ENABLE AND TON ON SECOUND  
NOP ;IBV - SHOULD CAUSE A TKR INTERRUPT.  
NOP

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

(1)  
(1) 006252 104001  
844  
845  
846

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

848 006254 000402  
849 006256  
(1) 006256 062706 000004  
850 006262 005077 173100  
851  
(1) 006266 013777 001416 173102  
(1) 006274 012777 004700 173114  
852  
853

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

(3)  
(3)  
(2) 006302 000004  
854  
(1)

::\*\*\*\*\*  
;\*TEST 44 \*TEST THAT SRQ CAN GENERATE AN INTERRUPT  
::\*\*\*\*\*  
TST44: SCOPE  
;\*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A  
;\*SECOUND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET.

```
(1) ;*ADDRESS OF THE SECONUD MODULE IS IN LOCATION 'IBS2' VECTOR
(1) ;*ADDRESS IS IN LOCATION 'VECTA2'. THE SECONUD IBV-11 SHOULD BE ELECTRICALLY SEC
(1) ;*TO INHIBIT THE USE OF TESTING WITH A SECONUD MODULE, MAKE
(1) ;*LOCATION '$CDW1' ZERO.
(1) ;*
855
856 006304 005077 173036 CLR @IBS ;CLEAR CSRS.
857 006310 005077 173052 CLR @IBS2
858
859 006314 012777 000200 173062 MOV #200,@PRB ;SET UP INTERRUPT VECTOR.
860 006322 012777 006366 173030 MOV #1$,@VECTB
861 ;/PR
(1) 006330 012746 000000 MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
(1) 006334 012746 006342 MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
(1) 006340 000002 RTI ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006342 ;/IN STATUS REG.)
862 006342 012777 000100 172776 64$: MOV #100,@IBS ;ENABLE INTERRUPTS
863 006350 052777 100000 173010 BIS #BIT15,@IBS2 ;SETTING SRQ IN THE 'CDW' MODULE
864 ;WILL PUT SRQ ON THE IB BUS
865 ;IS ER1 INH SW IS SET.
866 006356 000240 NOP
867 006360 000240 NOP
868
;:$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$
(1)
(1) 006362 104001 ERROR 1 ;/MODULE FAULT DETECTED:
869 ;SRQ FAILED TO GENERATE
870 ;AN INTERRUPT
;:$$$$$$$$$$$^^^ ERROR ^^$$$$$$$$$$$$
872 006364 000402 BR 2$
873
874 006366 1$: ADD #4,$P ;/ADD #4 TO STACK POINTER.
(1) 006366 062706 000004
875
876 006372 2$: ;/-RESV-
(1) 006372 013777 001404 172760 MOV PRB,@VECTB ;/RESTORE VECTOR FOR
(1) 006400 012777 004700 172776 MOV #4700,@PRB ;/ILLEGAL INTRO.
877 006406 005077 172734 CLR @IBS ;CLEAR CSRS
878 006412 005077 172750 CLR @IBS2
879
880 ;:*****
(3) ;*TEST 45 *TEST THAT ERROR1 IS GENERATED IF ATN IS ON THE IB BUS
(3) ;:*****
(2) 006416 000004 TST45: SCOPE
881
882 ;*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A
(1) ;*SECONUD MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET.
(1) ;*ADDRESS OF THE SECONUD MODULE IS IN LOCATION 'IBS2' VECTOR
(1) ;*ADDRESS IS IN LOCATION 'VECTA2'. THE SECONUD IBV-11 SHOULD BE ELECTRICALLY SEC
(1) ;*TO INHIBIT THE USE OF TESTING WITH A SECONUD MODULE, MAKE
```

```

(1) ;*LOCATION '$CDW1' ZERO.
(1) ;*
883
884 006420 005077 172742 CLR @IBS2 ;CLR CSR OF 2ND MODULE.
885 006424 005277 172736 INC @IBS2 ;ASSERT ATN ON IB BUS
886 ;ASSERTED ATN ON IBV UNDER TEST-
887 ;THIS SHOULD CAUSE AN ERROR 1
888 ;SENCE THE 2ND IBV HAS ATN SET.
889 006430 032777 020000 172710 BIT #BIT13,@IBS ;DID ERROR 1 SET?
890 006436 001001 BNE TST46 ;:
891

```

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

```

(1)
(1) 006440 104001 ERROR 1 ;/MODULE FAULT DETECTED:
892 ;FAILED TO GENERATE ERROR 1

```

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

```

894
895 ;*****
(3) ;*TEST 46 *TEST THAT ERROR 1 IS GENERATED IF IFC IS PUT ON IB BUS BY SECOUND MODUL
(3) ;*****
(2) 006442 000004 TST46: SCOPE
(1) 006444 012737 000005 001160 MOV #5,$TIMES ;:DO 5 ITERATIONS
896 ;*WARNING! THIS TEST IS DESIGNED TO BE EXERCISED WITH A
(1) ;*SECOUND MODULE (IBV-11) WITH SWITCH 'ER1 INH' SET.
(1) ;*ADDRESS OF THE SECOUND MODULE IS IN LOCATION 'IBS2' VECTOR
(1) ;*ADDRESS IS IN LOCATION 'VECTA2'. THE SECOUND IBV-11 SHOULD BE ELECTRICALLY SEC
(1) ;*TO INHIBIT THE USE OF TESTING WITH A SECOUND MODULE, MAKE
(1) ;*LOCATION '$CDW1' ZERO.
(1) ;*
897 006452 005077 172670 CLR @IBS ;CLEAR CSR
898 006456 012777 000010 172702 MOV #BIT3,@IBS2 ;ASSERT IFC FROM TESTOR
899 006464 032777 020000 172654 BIT #BIT13,@IBS ;DID ERROR 1 GET SET?
900 ;IF SO - NEXT TEST
901 006472 001010 BNE TST47 ;:
902 006474 012777 000010 172664 MOV #BIT3,@IBS2 ;IF NOT WE'LL TRY AGAIN SENCE MEMORY
903 006502 032777 020000 172636 BIT #BIT13,@IBS ;REFRESH COULD HAVE GO IN THE WAY.
904 006510 001001 BNE TST47 ;:
905

```

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

```

(1)
(1) 006512 104001 ERROR 1 ;/MODULE FAULT DETECTED:
906 ;ERROR 1 FAILED TO SET WHEN
907 ;IFC WAS ON IB-BUS AND MODULE
908 ;UNDER TEST DIDN'T PUT IT THERE.

```

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

```

910
911 ;*****
(3) ;*TEST 47 *TEST THAT ERROR 1 IS GENERATED IF REN IS ON IB BUS

```



```

(3)
(2) 006514 000004
912
913
(1)
(1)
(1)
(1)
(1)
(1)
(1)
914
915 006516 005077 172624 CLR @IBS ;CLEAR CSRS.
916 006522 005077 172640 CLR @IBS2
917 006526 052777 000004 172632 BIS #BIT2,@IBS2 ;ASSERT REN ON IB BUS FROM 2ND
918 ;MODULE. 1ST IBV-11 SHOULD
919 006534 032777 020000 172604 BIT #BIT13,@IBS ;GENERATE AN ERROR 1;DID IT??
920 006542 001001 BNE TST50 ;;
921

```

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

```

(1)
(1) 006544 104001 ERROR 1 ;/MODULE FAULT DETECTED:
922 ;FAILED TO GENERATE AN ERROR 1.

```

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

```

924
925
(3)
(3)
(2) 006546 000004
926
927
(1)
(1)
(1)
(1)
(1)
(1)
(1)
928
929 006550 005077 172612 CLR @IBS2 ;CLEAR CSRS.
930 006554 005077 172566 CLR @IBS
931
932 006560 012777 000200 172614 MOV #200,@PRA
933 006566 012777 006632 172562 MOV #1$,@VECTA ;SET UP VECTOR ADDR.
934
935 006574 052777 000100 172544 BIS #BIT06,@IBS ;SET INTERRUPT ENABLE
936 ;/PR
(1) 006602 012746 000000 MOV #0,-(SP) ;/SET CPU PRIORITY ON RETURN
(1) 006606 012746 006614 MOV #64$,-(SP) ;/SHOW RETURN ADDRESS
(1) 006612 000002 RTI ;/CAUSE A RETURN (PUTS NEW STATUS
(1) 006614 ;/IN STATUS REG.)
937 006614 052777 000004 172544 BIS #BIT2,@IBS2 ;GENERATE AN ERROR 1 AS PER LAST TEST.
938 006622 000240 NOP

```

64\$:

939 006624 000240  
940

NOP

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

(1)  
(1) 006626 104001  
941

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

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

943 006630 000402  
944  
945 006632  
(1) 006632 062706 000004  
946 006636

1\$:

ADD #4,SP ;/ADD #4 TO STACK POINTER.

2\$:

(1) 006636 013777 001402 172512  
(1) 006644 012777 004700 172530  
947 006652 005077 172510  
948 006656 005077 172464  
949  
960  
961

;/-RESV-  
MOV PRA,@VECTA ;/RESTORE VECTOR FOR  
MOV #4700,@PRA ;/ILLEGAL INTRO:  
CLR @IBS2 ;CLEAR CSRS.  
CLR @IBS

:::\*\*\*\*\*  
;\*TEST 51 \*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.  
;\*

(3)  
(2) 006662 000004  
962  
963 006664 012737 006704 001110  
964 006672 012737 000000 001124  
965 006700 005037 001126  
966  
967 006704 005077 172436  
968 006710 005077 172452  
969 006714 052777 000041 172424  
970 006722 052777 000020 172436  
971 006730 013777 001124 172412  
972 006736 117737 172426 001126  
973 006744 123737 001124 001126  
974 006752 001402  
975

:::\*\*\*\*\*  
TST51: SCOPE

MOV #1\$,\$LPERR ;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.  
MOVB @IBD2,\$BDDA' ;READ ATA FROM KGM.  
CMPB \$GDDAT,\$BDDAT ;DATA SENT = DATA RECEIVED?  
BEQ 2\$ ;YES, CONTINUE

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

(1)  
(1) 006754 104004  
976

ERROR 4 ;/MODULE FAULT DETECTED:  
;ERROR - BAD DATA PASSED BETWEEN

977 ;MODULE UNDER TEST AND KGM.

979 006756 000407  
980 BR TST52  
::\$\$\$\$\$\$\$\$\$^^^ ERROR ^^\$\$\$\$\$\$\$\$\$\$  
::

981 006760 105237 001124 2\$: INCB \$GDDAT ;CHANGE PATTERN.  
982 006764 001347 BNE 1\$ ;IF NOT DONE, CONTINUE.

983  
984 006766 005077 172354 CLR @IBS ;CLEAR CSR'S  
985 006772 005077 172370 CLR @IBS2

986  
987  
(3) :\*\*\*\*\*  
(3) :\*TEST 52 \*TEMP END OF TESTS  
(2) :\*\*\*\*\*

988 (2) 006776 000004 TST52: SCOPE

989 007000 EOP:

990  
991 .SBTTL SYSMAC ROUTINES:

992  
993 .SBTTL END OF PASS ROUTINE

(1) :\*\*\*\*\*  
(1) :\*INCREMENT THE PASS NUMBER (\$PASS)  
(1) :\*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)  
(1) :\*IF THERES A MONITOR GO TO IT  
(1) :\*IF THERE ISN'T JUMP TO RSTART

(1) 007000 \$EOP:

(1) 007000 000004 SCOPE  
(1) 007002 005037 001102 CLR \$TSTNM ;:ZERO THE TEST NUMBER  
(1) 007006 005037 001160 CLR \$TIMES ;:ZERO THE NUMBER OF ITERATIONS  
(1) 007012 005237 001202 INC \$PASS ;:INCREMENT THE PASS NUMBER  
(1) 007016 042737 100000 001202 BIC #100000,\$PASS ;:DON'T ALLOW A NEG. NUMBER  
(1) 007024 005327 DEC (PC)+ ;:LOOP?

(1) 007026 000001 \$EOPCT: .WORD 1  
(1) 007030 003022 BGT \$DOAGN ;:YES  
(1) 007032 012737 MOV (PC)+,@(PC)+ ;:RESTORE COUNTER

(1) 007034 000001 \$ENDCT: .WORD 1  
(1) 007036 007026 \$EOPCT  
(1) 007040 104401 007105 TYPE ,SENDMG ;:TYPE 'END PASS #'  
(2) 007044 013746 001202 MOV \$PASS,-(SP) ;:SAVE \$PASS FOR TYPEOUT  
(2) 007050 104405 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN

(1) 007052 104401 007102 TYPE ,SENULL ;:TYPE A NULL CHARACTER  
(1) 007056 013700 000042 \$GET42: MOV @#42,R0 ;:GET MONITOR ADDRESS  
(1) 007062 001405 BEQ \$DOAGN ;:BRANCH IF NO MONITOR  
(1) 007064 000005 RESET ;:CLEAR THE WORLD  
(1) 007066 004710 \$ENDAD: JSR PC,(R0) ;:GO TO MONITOR  
(1) 007070 000240 NOP ;:SAVE ROOM  
(1) 007072 000240 NOP ;:FOR  
(1) 007074 000240 NOP ;:ACT11

(1) 007076 \$DOAGN: JMP @(PC)+ ;:RETURN  
(1) 007076 000137 \$RTNAD: .WORD RSTART

(1) 007100 002266



```
(1) 007102 377 377 000 $ENULL: .BYTE -1,-1,0 ::NULL CHARACTER STRING
(1) 007105 015 042412 042116 $ENDMG: .ASCIZ <15><12>/END PASS #/
(1) 007112 050040 051501 020123
(1) 007120 000043
994 ;/DELMA
(1) ;/ROUTINE TO PROVIDE DELAYS IN INCREMENTS OF 25 US
(1) ;/
(1) ;/ CALL= JSR R5,DEL50
(1) ;/ .WORD X (# OF 25 US TO DELAY)
(1) ;/ ;RETURNS HERE
(1) ;/
(1) 007122 012500 DEL50: MOV (5)+,R0 ;/GET # OF 25 US DELAYS
(1) 007124 012701 000002 1$: MOV #2.,R1 ;/# FOR LOOP TO DO 50 US.
(1) 007130 005301 2$: DEC R1 ;/DEC IT
(1) 007132 001376 BNE 2$ ;/WAITED 25. TIMES?
(1) 007134 005300 DEC R0 ;/DONE # OF 50 US DELAY DESIRED?
(1) 007136 001372 BNE 1$ ;/NO - NEXT ONE.
(1) 007140 000205 RTS R5 ;/YES - EXIT.
(1)
995 .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) ;*SW10=1 BELL ON ERROR
(1) ;*SW09=1 LOOP ON ERROR
(1) ;*CALL
(1) ;* ERROR N ;:ERROR=EMT AND N=ERROR ITEM NUMBER
(1)
(1) 007142 $ERROR:
(1) 007142 104407 CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
(1) 007144 105237 001103 7$: INCB $ERFLG ;:SET THE ERROR FLAG
(1) 007150 001775 BEQ 7$ ;:DON'T LET THE FLAG GO TO ZERO
(1) 007152 013777 001102 171762 MOV $TSTNM,@DISPLAY ;:DISPLAY TEST NUMBER AND ERROR FLAG
(1) 007160 032777 002000 171752 BIT #BIT10,@SWR ;:BELL ON ERROR?
(1) 007166 001402 BEQ 1$ ;:NO - SKIP
(1) 007170 104401 001164 TYPE ,SBELL ;:RING BELL
(1) 007174 005237 001112 1$: INC $ERTTL ;:COUNT THE NUMBER OF ERRORS
(1) 007200 011637 001116 MOV (SP),$ERRPC ;:GET ADDRESS OF ERROR INSTRUCTION
(1) 007204 162737 000002 001116 SUB #2,$ERRPC
(1) 007212 117737 171700 001114 MOVB @$ERRPC,$ITEMB ;:STRIP AND SAVE THE ERROR ITEM CODE
(1) 007220 032777 020000 171712 BIT #BIT13,@SWR ;:SKIP TYPEOUT IF SET
(1) 007226 001004 BNE 20$ ;:SKIP TYPEOUTS
(1) 007230 004737 007330 JSR PC,$ERRTYP ;:GO TO USER ERROR ROUTINE
(1) 007234 104401 001171 TYPE ,$CRLF
(1) 007240
(1) 007240 122737 000001 001214 20$: CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
(1) 007246 001007 BNE 2$ ;:NO,SKIP APT ERROR REPORT
```



```

(1) 007440 000207           RTS      PC           ;;RETURN
(2) 007442              7$:      MOV      @(R0)+,-(SP)  ;;SAVE @(R0)+ FOR TYPEOUT
(2) 007442 013046           TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(2) 007444 104402           TST      (R0)    ;;IS THERE ANOTHER NUMBER?
(1) 007446 005710           BEQ      6$      ;;BR IF NO
(1) 007450 001770           TYPE     ,8$    ;;TYPE TWO(2) SPACES
(1) 007452 104401 007460     BR       7$      ;;LOOP
(1) 007456 000771           .ASCIZ  / /     ;;TWO(2) SPACES
(1) 007460 020040 000       .EVEN
(1) 007464 007464           .SBTTL SCOPE HANDLER ROUTINE

(1)
(2)
(1) *****
(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) 007464           $SCOPE:
(1) 007464 104407           CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(2) 007466 104407           CKSWR
(1) 007470 032777 040000 171442 1$:  BIT      #BIT14,@SWR  ;;LOOP ON PRESENT TEST?
(1) 007476 001114           BNE      $OVER    ;;YES IF SW14=1
(1)           ;#####START OF CODE FOR THE XOR TESTER#####
(1) 007500 000416           $XTSTR: BR     6$  ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)           MOV      @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 007502 013746 000004           MOV      #5$,@#ERRVEC ;;SET FOR TIMEOUT
(1) 007506 012737 007526 000004           TST      @#177060    ;;TIME OUT ON XOR?
(1) 007514 005737 177060           MOV      (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
(1) 007520 012637 000004           BR       $$VLAD    ;;GO TO THE NEXT TEST
(1) 007524 000463           5$:      CMP      (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
(1) 007526 022626           MOV      (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
(1) 007530 012637 000004           BR       7$      ;;LOOP ON THE PRESENT TEST
(1) 007534 000423           6$:;#####END OF CODE FOR THE XOR TESTER#####
(1) 007536 032777 000400 171374           BIT      #BIT08,@SWR  ;;LOOP ON SPEC. TEST?
(1) 007544 001404           BEQ      2$      ;;BR IF NO
(1) 007546 127737 171366 001102           CMPB    @SWR,$TSTNM  ;;ON THE RIGHT TEST? SWR<7:0>
(1) 007554 001465           BEQ      $OVER    ;;BR IF YES
(1) 007556 105737 001103           2$:      TSTB    $ERFLG    ;;HAS AN ERROR OCCURRED?
(1) 007562 001421           BEQ      3$      ;;BR IF NO
(1) 007564 123737 001115 001103           CMPB    $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 007572 101015           BHI     3$      ;;BR IF NO
(1) 007574 032777 001000 171336           BIT      #BIT09,@SWR  ;;LOOP ON ERROR?
(1) 007602 001404           BEQ      4$      ;;BR IF NO
(1) 007604 013737 001110 001106 7$:  MOV      $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 007612 000446           BR       $OVER
(1) 007614 105037 001103           4$:      CLRB    $ERFLG    ;;ZERO THE ERROR FLAG
  
```



```

(1) 007620 005037 001160 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 007624 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
(1) 007626 032777 004000 171304 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1) 007634 001011 BNE 1$ ;;BR IF YES
(1) 007636 005737 001202 TST $PASS ;;IF FIRST PASS OF PROGRAM
(1) 007642 001406 BEQ 1$ ;; INHIBIT ITERATIONS
(1) 007644 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 007650 023737 001160 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 007656 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 007660 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1) 007666 013737 007744 001160 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 007674 105237 001102 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
(1) 007700 113737 001102 001200 MOVB $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(1) 007706 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(1) 007712 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(1) 007716 005037 001162 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 007722 112737 000001 001115 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 007730 013777 001102 171204 $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 007736 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(1) 007742 000002 RTI ;;FIXES PS
(1) 007744 003720 $MXCNT: 2000. ;;MAX. NUMBER OF ITERATIONS

```

998 .SBTTL TTY INPUT ROUTINE

```

(1)
(2) ;:*****
(1) .ENABL LSB

```

```

(1) ;:*****
(1) ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL.
(1) ;*WHEN OPERATING IN TTY FLAG MODE.

```

```

(1) 007746 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
(1) 007754 001074 BNE 15$ ;;BRANCH IF NO
(1) 007756 105777 171162 TSTB @$TKS ;;CHAR THERE?
(1) 007762 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
(1) 007764 117746 171156 MOVB @$TKB,-(SP) ;;SAVE THE CHAR
(1) 007770 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
(1) 007774 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
(1) 010000 001062 BNE 15$ ;;NO, RETURN TO USER
(1) 010002 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
(1) 010010 001456 BEQ 15$ ;;BRANCH IF YES
(1) 010012 104401 010473 TYPE ,SCNTLG ;;ECHO THE CONTROL-G (^G)
(1) 010016 104401 010500 $GTSWR: TYPE ,SMSWR ;;TYPE CURRENT CONTENTS
(2) 010022 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
(2) 010026 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 010030 104401 010511 TYPE ,SMNEW ;;PROMPT FOR NEW SWR
(1) 010034 005046 19$: CLR -(SP) ;;CLEAR COUNTER
(1) 010036 005046 CLR -(SP) ;;THE NEW SWR
(1) 010040 105777 171100 7$: TSTB @$TKS ;;CHAR THERE?
(1) 010044 100375 BPL 7$ ;;IF NOT TRY AGAIN
(1)
(1) 010046 117746 171074 MOVB @$TKB,-(SP) ;;PICK UP CHAR
(1) 010052 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII

```



```

(1) 010262 026627 000004 000023      CMP      4(SP),#23      ;; IS IT A CONTROL-S?
(1) 010270 001013                    BNE      3$            ;; BRANCH IF NO
(1) 010272 105777 170646              2$:    TSTB     @TKS      ;; WAIT FOR A CHARACTER
(1) 010276 100375                    BPL      2$            ;; LOOP UNTIL ITS THERE
(1) 010300 117746 170642              MOVB     @TKB,-(SP)    ;; GET CHARACTER
(1) 010304 042716 177600              BIC      #^C177,(SP)  ;; MAKE IT 7-BIT ASCII
(1) 010310 022627 000021              CMP      (SP)+,#21    ;; IS IT A CONTROL-Q?
(1) 010314 001366                    BNE      2$            ;; IF NOT DISCARD IT
(1) 010316 000750                    BR       1$            ;; YES, RESUME
(1) 010320 026627 000004 000140      3$:    CMP      4(SP),#140  ;; IS IT UPPER CASE?
(1) 010326 002407                    BLT      4$            ;; BRANCH IF YES
(1) 010330 026627 000004 000175      CMP      4(SP),#175  ;; IS IT A SPECIAL CHAR?
(1) 010336 003003                    BGT      4$            ;; BRANCH IF YES
(1) 010340 042766 000040 000004      BIC      #40,4(SP)   ;; MAKE IT UPPER CASE
(1) 010346 0000C2                    4$:    RTI              ;; GO BACK TO USER
(2)                                     ;; *****
(1)                                     ;; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;; *CALL:
(1)                                     ;; *
(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) 010350 010346                    $RDLIN: MOV      R3,-(SP)  ;; SAVE R3
(1) 010352 012703 010456              1$:    MOV      #$TTYIN,R3  ;; GET ADDRESS
(1) 010356 022703 010466              2$:    CMP      #$TTYIN+8.,R3  ;; BUFFER FULL?
(1) 010362 101405                    BLOS     4$            ;; BR IF YES
(1) 010364 104410                    RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
(1) 010366 112613                    MOVB     (SP)+,(R3)   ;; GET CHARACTER
(1) 010370 122713 000177              10$:   CMPB     #177,(R3)    ;; IS IT A RUBOUT
(1) 010374 001003                    BNE      3$            ;; SKIP IF NOT
(1) 010376 104401 001170              4$:    TYPE     ,SQUES   ;; TYPE A '?'
(1) 010402 000763                    BR       1$            ;; CLEAR THE BUFFER AND LOOP
(1) 010404 111337 010454              3$:    MOVB     (R3),9$    ;; ECHO THE CHARACTER
(1) 010410 104401 010454              TYPE     ,9$
(1) 010414 122723 000015              CMPB     #15,(R3)+   ;; CHECK FOR RETURN
(1) 010420 001356                    BNE      2$            ;; LOOP IF NOT RETURN
(1) 010422 105063 177777              CLRB    -1(R3)       ;; CLEAR RETURN (THE 15)
(1) 010426 104401 001172              TYPE     ,LF         ;; TYPE A LINE FEED
(1) 010432 012603                    MOV      (SP)+,R3    ;; RESTORE R3
(1) 010434 011646                    MOV      (SP),-(SP)  ;; ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 010436 016666 000004 000002      MOV      4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 010444 012766 010456 000004      MOV      #$TTYIN,4(SP)
(1) 010452 000002                    RTI              ;; RETURN
(1) 010454 000          9$:    .BYTE     0          ;; STORAGE FOR ASCII CHAR. TO TYPE
(1) 010455 000          .BYTE     0          ;; TERMINATOR
(1) 010456 000010      $TTYIN: .BLKB     8.   ;; RESERVE 8 BYTES FOR TTY INPUT
(1) 010466 052536 005015 000      $CNTLU: .ASCIZ   /^U/<15><12>  ;; CONTROL 'U'
(1) 010473 136 006507 000012      $CNTLG: .ASCIZ   /^G/<15><12>  ;; CONTROL 'G'
(1) 010500 005015 053523 020122      $MSWR:  .ASCIZ   <15><12>/SWR = /
(1) 010506 020075 000
(1) 010511 040 047040 053505      $MNEW:  .ASCIZ   / NEW = /
(1) 010516 036440 000040
999                                     .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)

```





(1)	010664	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
(1)	010666	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
(1)	010672	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
(1)	010676	110337	010742		MOVB	R3,8\$	::SAVE FOR TYPING
(1)	010702	104401	010742		TYPE	,8\$	::GO TYPE THIS DIGIT
(1)	010706	105337	010744	7\$:	DECB	\$OCNT	::COUNT BY 1
(1)	010712	003347			BGT	2\$	::BR IF MORE TO DO
(1)	010714	002402			BLT	6\$	::BR IF DONE
(1)	010716	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
(1)	010720	000744			BR	2\$	::GO DO THE LAST DIGIT
(1)	010722	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
(1)	010724	012604			MOV	(SP)+,R4	::RESTORE R4
(1)	010726	012603			MOV	(SP)+,R3	::RESTORE R3
(1)	010730	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
(1)	010736	012616			MOV	(SP)+,(SP)	
(1)	010740	000002			RTI		::RETURN
(1)	010742	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
(1)	010743	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
(1)	010744	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
(1)	010745	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
(1)	010746	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

```

1001      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)      ::*****
(1)      ::*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)      ::*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT.  DEPENDING ON WHETHER THE
(1)      ::*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)      ::*BEFORE THE FIRST DIGIT OF THE NUMBER.  LEADING ZEROS WILL ALWAYS BE
(1)      ::*REPLACED WITH SPACES.
(1)      ::*CALL:
(1)      ::*      MOV      NUM,-(SP)      ::PUT THE BINARY NUMBER ON THE STACK
(1)      ::*      TYPDS      ::GO TO THE ROUTINE
(2)      $TYPDS:
(3)      010750      MOV      R0,-(SP)      ::PUSH R0 ON STACK
(3)      010752      MCV      R1,-(SP)      ::PUSH R1 ON STACK
(3)      010754      MOV      R2,-(SP)      ::PUSH R2 ON STACK
(3)      010756      MOV      R3,-(SP)      ::PUSH R3 ON STACK
(3)      010760      MOV      R5,-(SP)      ::PUSH R5 ON STACK
(1)      010762      MOV      #20200,-(SP)  ::SET BLANK SWITCH AND SIGN
(1)      010766      MOV      20(SP),R5    ::GET THE INPUT NUMBER
(1)      010772      BPL      1$          ::BR IF INPUT IS POS.
(1)      010774      NEG      R5          ::MAKE THE BINARY NUMBER POS.
(1)      010776      MOV      #'-,1(SP)    ::MAKE THE ASCII NUMBER NEG.
(1)      011004      005000      1$:      CLR      R0          ::ZERO THE CONSTANTS INDEX
(1)      011006      012703      011164    MOV      #$DBLK,R3    ::SETUP THE OUTPUT POINTER
(1)      011012      112723      000040    MOV      #' ,(R3)+    ::SET THE FIRST CHARACTER TO A BLANK
(1)      011016      005002      2$:      CLR      R2          ::CLEAR THE BCD NUMBER
(1)      011020      016001      011154    MOV      $DTBL(R0),R1 ::GET THE CONSTANT
(1)      011024      160105      3$:      SUB      R1,R5        ::FORM THIS BCD DIGIT
(1)      011026      002402      BLT      4$          ::BR IF DONE
(1)      011030      005202      INC      R2          ::INCREASE THE BCD DIGIT BY 1
(1)      011032      000774      BR      3$
(1)      011034      060105      4$:      ADD      R1,R5        ::ADD BACK THE CONSTANT
(1)      011036      005702      TST      R2          ::CHECK IF BCD DIGIT=0
(1)      011040      001002      BNE      5$          ::FALL THROUGH IF 0
(1)      011042      105716      TSTB    (SP)         ::STILL DOING LEADING 0'S?
(1)      011044      100407      BMI      7$          ::BR IF YES
(1)      011046      106316      5$:      ASLB    (SP)         ::MSD?
(1)      011050      103003      BCC      6$          ::BR IF NO
(1)      011052      116663      000001  177777    MOV      1(SP),-1(R3) ::YES--SET THE SIGN
(1)      011060      052702      000060  6$:      BIS      #'0,R2      ::MAKE THE BCD DIGIT ASCII
(1)      011064      052702      000040  7$:      BIS      #' ,R2      ::MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)      011070      110223      MOV      R2,(R3)+    ::PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)      011072      005720      TST      (R0)+      ::JUST INCREMENTING
(1)      011074      020027      000010    CMP      R0,#10      ::CHECK THE TABLE INDEX
(1)      011100      002746      BLT      2$          ::GO DO THE NEXT DIGIT
(1)      011102      003002      BGT      8$          ::GO TO EXIT
(1)      011104      010502      MOV      R5,R2      ::GET THE LSD
(1)      011106      000764      BR      6$          ::GO CHANGE TO ASCII
(1)      011110      105726      8$:      TSTB    (SP)+      ::WAS THE LSD THE FIRST NON-ZERO?
(1)      011112      100003      9$:      BPL      9$          ::BR IF NO
(1)      011114      116663      177777  177776    MOV      -1(SP),-2(R3) ::YES--SET THE SIGN FOR TYPING
(1)      011122      105013      CLRB    (R3)         ::SET THE TERMINATOR
(3)      011124      012605      MOV      (SP)+,R5    ::POP STACK INTO R5

```





```

(1) 011306 001006 BNE 5$
(1) 011310 005726 TST (SP)+ ::POP <CR><LF> EQUIV
(1) 011312 104401 TYPE ::TYPE A CR AND LF
(1) 011314 001171 $CRLF
(1) 011316 105037 011452 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
(1) 011322 000755 BR 2$ ::GET NEXT CHARACTER
(1) 011324 004737 011406 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
(1) 011330 123726 001156 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
(1) 011334 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
(1) 011336 013746 001154 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
(1) ::AND THE NULL CHAR.
(1) 011342 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
(1) 011346 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
(1) 011350 004737 011406 JSR PC,$TYPEC ::GO TYPE A NULL
(1) 011354 105337 011452 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
(1) 011360 000770 BR 7$ ::LOOP
  
```

;HORIZONTAL TAB PROCESSOR

```

(1) 011362 112716 000040 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
(1) 011366 004737 011406 9$: JSR PC,$TYPEC ::TYPE A SPACE
(1) 011372 132737 000007 011452 BITB #7,$CHARCNT ::BRANCH IF NOT AT
(1) 011400 001372 BNE 9$ ::TAB STOP
(1) 011402 005726 TST (SP)+ ::POP SPACE OFF STACK
(1) 011404 000724 BR 2$ ::GET NEXT CHARACTER
(1) 011406 105777 167536 $TYPEC: TSTB @STPS ::WAIT UNTIL PRINTER IS READY
(1) 011412 100375 BPL $TYPEC
(1) 011414 116677 000002 167530 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 011422 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
(1) 011430 001003 BNE 1$ ::BRANCH IF NO
(1) 011432 105037 011452 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT
(1) 011436 000406 BR $TYPEX ::EXIT
(1) 011440 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
(1) 011446 001402 BEQ $TYPEX ::BRANCH IF YES
(1) 011450 105227 INCB (PC)+ ::COUNT THE CHARACTER
(1) 011452 000000 $CHARCNT: .WORD 0 ::CHARACTER COUNT STORAGE
(1) 011454 000207 $TYPEX: RTS PC
  
```

.SBTTL APT COMMUNICATIONS ROUTINE

```

(1)
(2)
(1) 011456 112737 000001 011722 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
(1) 011464 112737 000001 011720 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
(1) 011472 000403 BR $ATYC
(1) 011474 112737 000001 011722 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
(2) 011502 $ATYC:
(3) 011502 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
(3) 011504 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
(1) 011506 105737 011720 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
(1) 011512 001450 BEQ 5$ ::IF NOT: BR
(1) 011514 122737 000001 001214 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
(1) 011522 001031 BNE 3$ ::IF NOT: BR
(1) 011524 132737 000100 001215 BITB #APTPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
(1) 011532 001425 BEQ 3$ ::IF NOT: BR
  
```



```
(1) 011534 017600 000004      MOV @4(SP),R0      ;;GET MESSAGE ADDR.
(1) 011540 062766 000002 000004  ADD #2,4(SP)      ;;BUMP RETURN ADDR.
(1) 011546 005737 001174      TST $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
(1) 011552 001375      BNE 1$           ;;IF NOT: WAIT
(1) 011554 010037 001210      MOV R0,$MSGAD    ;;PUT ADDR IN MAILBOX
(1) 011560 105720      TSTB (R0)+      ;;FIND END OF MESSAGE
(1) 011562 001376      BNE 2$
(1) 011564 163700 001210      SUB $MSGAD,R0   ;;SUB START OF MESSAGE
(1) 011570 006200      ASR R0          ;;GET MESSAGE LNGTH IN WORDS
(1) 011572 010037 001212      MOV R0,$MSGGLT  ;;PUT LENGTH IN MAILBOX
(1) 011576 012737 000004 001174  MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 011604 000413      BR 5$
(1) 011606 017637 000004 011632 3$: MOV @4(SP),4$   ;;PUT MSG ADDR IN JSR LINKAGE
(1) 011614 062766 000002 000004  ADD #2,4(SP)     ;;BUMP RETURN ADDRESS
(3) 011622 013746 177776      MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 011626 004737 011174      JSR PC,$TYPE    ;;CALL TYPE MACRO
(1) 011632 000000      4$: .WORD 0
(1) 011634      5$:
(1) 011634 105737 011722      10$: TSTB $FFLG      ;;SHOULD REPORT FATAL ERROR?
(1) 011640 001416      BEQ 12$         ;;IF NOT: BR
(1) 011642 005737 001214      TST $ENV        ;;RUNNING UNDER APT?
(1) 011646 001413      BEQ 12$         ;;IF NOT: BR
(1) 011650 005737 001174      11$: TST $MSGTYPE    ;;FINISHED LAST MESSAGE?
(1) 011654 001375      BNE 11$        ;;IF NOT: WAIT
(1) 011656 017637 000004 001176  MOV @4(SP),$FATAL ;;GET ERROR #
(1) 011664 062766 000002 000004  ADD #2,4(SP)     ;;BUMP RETURN ADDR.
(1) 011672 005237 001174      INC $MSGTYPE    ;;TELL APT TO TAKE ERROR
(1) 011676 105037 011722      12$: CLRB $FFLG    ;;CLEAR FATAL FLAG
(1) 011702 105037 011721      CLRB $LFLG     ;;CLEAR LOG FLAG
(1) 011706 105037 011720      CLRB $MFLG     ;;CLEAR MESSAGE FLAG
(3) 011712 012601      MOV (SP)+,R1   ;;POP STACK INTO R1
(3) 011714 012600      MOV (SP)+,R0   ;;POP STACK INTO R0
(1) 011716 000207      RTS PC        ;;RETURN
(1) 011720      000           $MFLG: .BYTE 0  ;;MESSG. FLAG
(1) 011721      000           $LFLG: .BYTE 0  ;;LOG FLAG
(1) 011722      000           $FFLG: .BYTE 0  ;;FATAL FLAG
(1) 011724      .EVEN
(1) 000200      APTSIZE=200
(1) 000001      APTENV=001
(1) 000100      APTSPool=100
(1) 000040      APTCSUP=040
1004 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2) ;;*****
(1) :POWER DOWN ROUTINE
(1) $PWRDN: MOV # $ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 011732 012737 000340 000026  MOV #340,@#PWRVEC+2 ;;PRIO:7
(3) 011740 010046      MOV R0,-(SP)   ;;PUSH R0 ON STACK
(3) 011742 010146      MOV R1,-(SP)   ;;PUSH R1 ON STACK
(3) 011744 010246      MOV R2,-(SP)   ;;PUSH R2 ON STACK
(3) 011746 010346      MOV R3,-(SP)   ;;PUSH R3 ON STACK
(3) 011750 010446      MOV R4,-(SP)   ;;PUSH R4 ON STACK
(3) 011752 010546      MOV R5,-(SP)   ;;PUSH R5 ON STACK
(3) 011754 017746 167160      MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
```



```

(1) 011760 010637 012070      MOV      SP,$SAVR6      ;;SAVE SP
(1) 011764 012737 011776 000024  MOV      #$PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 011772 000000      HALT
(1) 011774 000776      BR        -2           ;;HANG UP
(1)
(2)
(1)
:*****
:POWER UP ROUTINE
(1) 011776 012737 012064 000024 $PWRUP: MOV      #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1) 012004 013706 012070      MOV      $SAVR6,SP      ;;GET SP
(1) 012010 005037 012070      CLR      $SAVR6        ;;WAIT LOOP FOR THE TTY
(1) 012014 005237 012070      1$: INC      $SAVR6      ;;WAIT FOR THE INC
(1) 012020 001375      BNE      1$           ;;OF WORD
(3) 012022 012677 167112      MOV      (SP)+,@SWR     ;;POP STACK INTO @SWR
(3) 012026 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
(3) 012030 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
(3) 012032 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
(3) 012034 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
(3) 012036 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
(3) 012040 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
(1) 012042 012737 011724 000024  MOV      #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 012050 012737 000340 000026  MOV      #340,@#PWRVEC+2 ;;PRIO:7
(1) 012056 104401      TYPE     $POWER        ;REPORT THE POWER FAILURE
(1) 012060 012072      $PWRMG: .WORD $POWER    ;;POWER FAIL MESSAGE POINTER
(1) 012062 000002      RTI
(1) 012064 000000      $ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
(1) 012066 000776      BR        -2           ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 012070 000000      $SAVR6: 0             ;;PUT THE SP HERE
(1) 012072 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'
(1) 012100 000122
(1)
1005      .EVEN
1006
1007      ;*
1008      ;*THIS ROUTINE WILL PROTECT THE PROGRAM
1009      ;*FROM INTERRUPTS.
1010      ;*
1011      ;*THE TRAP CATCHER IS SET UP FOR
1012      ;*      .WORD      +2
1013      ;*      .WORD      JSR      PC,R0
1014      ;*
1015      ;*ILLEGAL INTERRUPTS OR INTERRUPTS TO THE WRONG VECTOR
1016      ;*GOTO THE VECTOR AND PICK UP THE '.+2' AS AN ADDRESS
1017      ;*AND '4700' AS NEW STATUS.
1018      ;*THE .+2 AS A PC WILL CAUSE EXECUTION OF THE 'JSR PC,R0' (AN ILLEGAL INSTR).
1019      ;*AND TRAP TO LOCATION '4'. IN LOCATION 10 WE HAVE A
1020      ;*POINTER HERE. IF THIS CONDITION CAUSES A TRAP TO LOC. 4
1021      ;*WE WILL REPORT IT IN THE SAME MANNER THAT WE WOULD
1022      ;*REPORT ANY OTHER ERROR.
1023
1024      ;*IF A BUSS ERROR TRAP DID OCCUR AND CAUSE A TRAP TO 4,
1025      ;*WE WILL HALT.
1026 012102 011637 012146      IOTRD: MOV      (6),TRTO    ;GET WHERE WE CAME TO.
1027 012106 162737 000004 012146  SUB      #4,TRTO      ;FORM REAL ADDR.
1028

```

```

1029 012114 023727 012146 001000      CMP      TRTO,#1000      ;DID TRAP COME FROM LESS THAN ADDR. 1000?
1030 012122 003402                      BLE      2$
1031
1032 012124 000000                      1$:     HALT           ;NO!  MUST BE A BUSS ILLEGAL ADDR. TIME OUT.
1033                                     ; ADDRESS CONTAINED IN TRTO.
1034
1035 012126 000776                      BR       1$           ;DON'T ALLOW A CONTINUE.
1036 012130                                     2$:
1037
1038 012130 016637 000004 012150      MOV      4(6),TRFRO    ;GET TRAPPED FROM ADDR.
1039
1040 012136 062706 000004              ADD      #4,SP        ;/ADD #4 TO STACK POINTER.
1041
1042

```

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

```

(1)
(1) 012142 104007      ERROR 7           ;/MODULE FAULT DETECTED:
1043                                     ;ERROR! ILLEGAL INTERRUPT
1044                                     ;OR INTERRUPT TO WRONG
1045                                     ;VECTOR - IF TEST NUMBER
1046                                     ;IS LESS THAN 10, ITS LIKELY
1047                                     ;(BUT NOT EXCLUSIVELY) TO BE A
1048                                     ;DEVICE OTHER THAN THE IBV-11
1049                                     ;TO BLAME.
1050                                     ;IF THE INTERRUPT OCCURRED
1051                                     ;DURING AN INTERRUPT TEST, I'D
1052                                     ;SUSPECT A PROBLEM WITH THE
1053                                     ;IBV-11.
1054                                     ;IF THE ADDRESS THE INTERRUPT
1055                                     ;VECTOR TO IS WITHIN THE RANGE
1056                                     ;OF VECTORS ASSIGNED TO THE IBV-11,
1057                                     ;THEN I'D SUSPECT THE IBV-11
1058                                     ;INTERRUPTED ILLEGALLY.
1059                                     ;IF THE ADDRESS THE INTERRUPT
1060                                     ;VECTORED TO IS OUTSIDE OF THE
1061                                     ;RANGE ASSIGNED TO THE IBV-11,
1062                                     ;I'D SUSPECT THAT THE
1063                                     ;IBV-11 PUT THE WRONG VECTOR ON
1064                                     ;THE BUSS DURING THE INTERRUPT
1065                                     ;PROCESS.
1066                                     ;FOR THIS ERROR - DON'T
1067                                     ;USE 'LOOP ON ERROR' OPTION.
1068                                     ;ALSO EXPECT THE INTERRUPT TEST TO
1069                                     ;REPORT THAT THE IBV-11 DIDN'T
1070                                     ;INTERRUPT.
1071                                     ;FOLLOW RECOMMENDED PROCEDURE
1072                                     ;IN THE DOCUMENT (ON THIS DIAGNOSTIC)
1073                                     ;FOR LOOPING ON ERROR

```

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

1075 012144 000002  
1076

1077 012146 000000 TRTO: .WORD 0 ;ADDR THAT WE INTERRUPTED TO  
1078 012150 000000 TRFRO: .WORD 0 ;ADDR THAT WE INTERRUPTED FROM.

1079  
1080 .SBTTL TRAP DECODER

(1)  
(2) ;\*\*\*\*\*  
(1) ;\*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION  
(1) ;\*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS  
(1) ;\*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL  
(1) ;\*GO TO THAT ROUTINE.

(1) 012152 010046 \$TRAP: MOV R0,-(SP) ;:SAVE R0  
(1) 012154 016600 000002 MOV 2(SP),R0 ;:GET TRAP ADDRESS  
(1) 012160 005740 TST -(R0) ;:BACKUP BY 2  
(1) 012162 111000 MOV B(R0),R0 ;:GET RIGHT BYTE OF TRAP  
(1) 012164 006300 ASL R0 ;:POSITION FOR INDEXING  
(1) 012166 016000 012206 MOV \$TRPAD(R0),R0 ;:INDEX TO TABLE  
(1) 012172 000200 RTS R0 ;:GO TO ROUTINE

(1)  
(1) ;:THIS IS USE TO HANDLE THE 'GETPRI' MACRO

(1) 012174 011646 \$TRAP2: MOV (SP),-(SP) ;:MOVE THE PC DOWN  
(1) 012176 016666 000004 000002 MOV 4(SP),2(SP) ;:MOVE THE PSW DOWN  
(1) 012204 000002 RTI ;:RESTORE THE PSW

(1)  
(3) .SBTTL TRAP TABLE  
(3) ;\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
(3) ;\*BY THE 'TRAP' INSTRUCTION.

(3) : ROUTINE  
(3) :-----  
(3) \$TRPAD: .WORD \$TRAP2  
(3) \$TYPE ;:CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE  
(3) \$TYPOC ;:CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)  
(3) \$TYPOS ;:CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)  
(3) \$TYPON ;:CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)  
(3) \$TYPDS ;:CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)  
(1)  
(3) 012206 012174 \$GTSWR ;:CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING  
(1)  
(3) 012224 007746 \$CKSWR ;:CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR  
(3) 012226 010230 \$RDCHR ;:CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE  
(3) 012230 010350 \$RDLIN ;:CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE

1081  
1082 .SBTTL MESSAGES AND TABLES

1083  
1084 012232 005007 044415 051502 EM1: .ASCIZ<7><12><15>#IBS FUNCTION ERROR#  
012240 043040 047125 052103  
012246 047511 020116 051105  
012254 047522 000122

1085  
1086 012260 005007 044415 042102 EM2: .ASCIZ<7><12><15>#IBD FUNCTION ERROR#





1105		012654				.EVEN	
1106							
1107	012654	001200	001116	001346	DT1:	.WORD	\$TESTN,\$ERRPC,IBS
1108							
1109	012662	000000			IBSA:	.WORD	0
1110							
1111	012664	000000	000000		IBDA:	.WORD	0,0
1112							
1113	012670	001200	001116	001124	DT3:	.WORD	\$TESTN,\$ERRPC,\$GDDAT,\$BDDAT,0
	012676	001126	000000				
1114							
1115	012702	001200	001116	001346	DT5:	.WORD	\$TESTN,\$ERRPC,IBS,0
	012710	000000					
1116							
1117	012712	001200	001116	012146	DT7:	.WORD	\$TESTN,\$ERRPC,TRTO,TRFRO,0
	012720	012150	000000				
1118							
1119	012724	000000	000000		DF0:	.WORD	0,0
1120							
1121		000001				.END	





























ADD	217 252 998 996	219 254 999 998	221 256 1001 1080	227 283 1002	229 288 1003	231 317 1040	233 324	235 754	237 773	239 787	242 808	244 849	246 874	248 945	250 996	
ASL																
ASLB	1001															
ASR	1003															
BCC	1001															
BEQ	211 486 616 999 997	262 501 662 1002 998	338 510 664 1003 999	349 522 666 1001	365 531 668	373 563 682	395 564 706	407 565 719	416 566 731	428 567 974	437 568 993	449 569 995	453 570 996	465 578 997	476 610 998	
BGE																
BGT	993	998	999	1001												
BHI	997															
BIC	223 662	392 666	413 668	434 993	483 998	508 999	528	563	564	565	566	567	568	569	570	
BIS	382 715	383 746	403 762	424 797	445 863	472 917	497 935	518 937	597 969	662 970	664 998	666 999	668 1001	680	691	
BISB	996															
BIT	590 903	600 919	609 995	615 997	662	664	666	668	673	681	695	716	718	889	899	
BITB	211	1002	1003													
BLE	1030															
BLOS	998															
BLT	998	999	1001	1002												
BMI	1001															
BNE	211 920 995	262 982 998	591 994 999	601 995 1001	662 996 1002	664 997	666 998	668 999	674 1001	696 1002	717 1003	831 1004	890	901	904	
BPL																
BR	211 564 753 1003	262 565 771 1004	282 566 785 1035	315 567 807	322 568 848	370 569 872	386 570 943	390 595 979	411 605 995	432 614 996	457 662 997	481 664 998	505 666 999	526 668 1001	563 678 1002	
CLR	211 517 664 857 985 693	336 529 666 877 993 798	347 563 668 878 996 997	363 564 672 884 897 997	381 565 690 897 998 1001	393 566 743 915 999 1002	402 567 755 916 1001 1003	414 568 769 929 1004	423 569 788 930	435 570 793 947	444 574 809 948	471 576 835 965	484 588 836 967	495 589 850 968	507 662 856 984	
CLRB																
CMP	211	262	385	406	427	448	452	464	475	500	521	997	998	1001	1029	
CMPB	262	563	564	565	566	567	568	569	570	973	995	997	998	1002	1003	
DEC	993	994	996													
DECB	999	1002														
EMT	44															
HALT	995	1002	1004	1032												
INC	262	885	993	995	997	998	999	1001	1003	1004						
INCB	778	981	995	997	1002											
IOT	44															
JMP	42	832	993													
JSR	459	993	995	998	1002	1003										
MOV	211 236 267 343	212 238 268 361	213 241 269 364	215 243 271 372	216 245 272 384	218 247 278 394	220 249 292 404	222 251 293 405	224 253 303 415	225 255 305 425	226 258 306 426	228 259 312 436	230 260 330 446	232 261 332 447	234 265 337 450	



.IIF	12	14	57	211	262	993	995	996	997	998	1002	1080			
.IRP	209	265	303	332	343	361	379	400	421	442	469	491	515	563	554
	565	566	567	568	569	570	572	586	662	664	666	668	670	688	701
	712	725	741	758	791	816	853	880	895	911	925	961	987	997	1001
.LIST	1003	1004													
	2	14	32	44	57	209	211	262	265	284	287	289	291	303	318
	321	325	328	332	339	341	343	350	352	361	366	369	374	377	379
	387	389	396	398	400	408	410	417	419	421	429	431	438	440	442
	454	456	466	468	469	478	480	487	489	491	502	504	511	513	515
	523	525	532	534	563	564	565	566	567	568	569	570	572	579	584
	586	592	594	602	604	611	613	617	619	662	664	666	668	670	675
	677	683	685	688	697	699	701	707	710	712	720	723	725	732	735
	741	750	752	758	766	768	782	784	791	804	806	816	843	847	853
	868	871	880	891	893	895	905	909	911	921	923	925	940	942	961
.MACRO	975	978	987	993	995	997	998	1042	1074	1080					
	14	57	59	111	126	137	141	149	155	211	295	354	536	620	818
	950	1080													
.MCALL	5	6	7	8	9	44	57	211	262						
.MEXIT	57														
.NLIST	1	14	24	44	57	209	211	262	265	284	287	289	291	303	318
	321	325	328	332	339	341	343	350	352	361	366	369	374	377	379
	387	389	396	398	400	408	410	417	419	421	429	431	438	440	442
	454	456	466	468	469	478	480	487	489	491	502	504	511	513	515
	523	525	532	534	563	564	565	566	567	568	569	570	572	579	584
	586	592	594	602	604	611	613	617	619	662	664	666	668	670	675
	677	683	685	688	697	699	701	707	710	712	720	723	725	732	735
	741	750	752	758	766	768	782	784	791	804	806	816	843	847	853
	868	871	880	891	893	895	905	909	911	921	923	925	940	942	961
	975	978	987	993	995	997	998	1042	1074	1080					
.PAGE	57														
.REPT	26														
.SBTTL	14	16	44	51	55	57	175	207	211	262	265	303	332	343	361
	379	400	421	442	469	491	515	563	564	565	566	567	568	569	570
	572	586	662	664	666	668	670	688	701	712	725	737	738	739	741
	758	791	812	813	814	816	853	880	895	911	925	961	987	991	993
	995	996	997	998	999	1001	1002	1003	1004	1080	1082				
.TITLE	12														
.WORD	25	31	34	36	37	39	51	55	57	179	180	181	182	183	184
	185	186	187	188	189	190	191	192	196	197	198	199	201	202	203
	204	460	993	996	999	1002	1003	1004	1077	1078	1080	1107	1109	1111	1113
	1115	1117	1119												

ERRORS DETECTED: 0



\*CVIBBA, CVIBBA/CRF=CVIBBA  
RUN-TIME: 30 13 2 SECONDS  
CORE USED: 25K