

DRV11-J

DIAG TST PRT 2  
CVDRDBO

AH-F761B-MC  
FICHE 1 OF 1

JUL 1982  
COPYRIGHT © 79-82  
MADE IN USA



IDENTIFICATION

PRODUCT CODE: AC-F759B-MC  
PRODUCT NAME: CVDRDB0 DRV11J DIAG TST PRT2  
DATE CREATED: 17-FEB-82  
MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1979, 1982 DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
4.1	PROGRAM START
5.0	SOFTWARE SWITCH REGISTER
5.1	OPTIONS
5.2	CONTROL
6.0	ERROR REPORTING
6.1	ERROR COMMENT
6.2	ERROR DATA
7.0	MISCELLANEOUS
7.1	DRV11J BUS ADDRESS MODIFICATION
7.2	XXDP/APT NOTES
7.3	POWER FAIL
7.4	MULTIPLE DRV11J INTERFACE TESTING
7.5	RESTRICTIONS
8.0	EXECUTION TIME
9.0	PROGRAM TEST DESCRIPTIONS
10.0	LISTING

1.0 ABSTRACT

THE DRV11-J IS A GENERAL PURPOSE PARALLEL INTERFACE FOR THE LSI-11 BUS. IT HAS A BASIC CONFIGURATION OF 64 TRI-STATE IN/OUT LINES DIVIDED INTO FOUR GROUPS OF 16 BIT WORDS.

THERE ARE TWO 4K DIAGNOSTICS FOR THE DRV11-J OPTION. THE DRV11-J DIAGNOSTIC TEST PART 1 OF 2 CONTAINS A SERIES OF TESTS WITHOUT DRV11J INTERRUPTS DESIGNED TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS MADE ACCESSIBLE WITH THE LOOPBACK CABLE INSERTED INTO THE DRV11-J I/O CONNECTORS.

THE DRV11-J DIAGNOSTIC PART 2 OF 2 IS A SERIES OF TESTS WITH DRV11J INTERRUPTS DESIGNED TO TEST ALL LOGIC AND DATA PATHS MADE ACCESSIBLE WITH THE LOOPBACK CABLE INSERTED INTO THE I/O CONNECTORS.

THE DRV11-J IS CONTAINED ON A DOUBLE HEIGHT MODULE. THE MODULE CONTAINS TWO 50 PIN CONNECTORS FOR INTERFACING TO EXTERNAL USER DEVICES.

FOR DIAGNOSTIC TESTING, THE DRV11-J CABLE (BC05W-02) MUST BE INSTALLED WITH 1/2 TWIST BETWEEN THE 50 PIN CONNECTORS.

```

*****
*                                     ::GPA *
* NOTE: THIS DIAGNOSTIC HAS BEEN MODIFIED TO RUN IN KXT11 (SBC 11/21) *
* BASED SYSTEMS. THE PROGRAM WILL AUTOMATICALLY ADJUST ITSELF TO RUN *
* IN THE APPROPRIATE ENVIRONMENT AS FOLLOWS: *
* *
*          LSI-11, 11/2, AND 11/23          SBC 11/21          *
*          -----          -----          *
* CSR RANGE:          160010 TO 177760          174000 TO 177760 *
* PROGRAMMABLE... *
* ...VECTOR RANGE:   0000 TO 1774          000 TO 374          *
* *                                     ::GPA *
*****

```

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11/03, 11/23 COMPUTER OR LSI-11 PROCESSOR WITH A MINIMUM OF 4K MEMORY.
2. SERIAL LINE INTERFACE AND CONSOLE TERMINAL
3. DRV11-J OPTION WITH A BC05W-02 CABLE

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE

4.0 STARTING PROCEDURE

1. MAKE SURE THE DRV11-J CABLE IS INSERTED WITH 1/2 A TWIST ON THE I/O CONNECTORS OF THE DRV11-J OPTION. THIS WILL CONNECT PORT A TO PORT C AND CONNECT PORT B TO PORT D.
2. MAKE SURE THE DEVICE BUS ADDRESSES AGREE WITH THE DEFAULT VALUES DEFINED IN SECTION 7.1. IF NOT, CHANGE LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' ODT COMMAND.
3. THE PROGRAM SHOULD ALWAYS BE STARTED AT 200. STARTING AT 200(200G OR .R CVDRDB UNDER XXDP+), THE PROGRAM INITIALIZES ITSELF, PRINTS ITS ID(FIRST TIME ONLY) AND THEN PRINTS THAT THE DRV11-J CABLE IS REQUIRED(FIRST TIME ONLY) AND THEN PRINTS: SWR=XXXXXX NEW=

WHERE XXXXXX REPRESENTS THE CURRENT VALUE OF THE SOFTWARE SWITCH REGISTER. IF NO CHANGES ARE REQUIRED IN THE SWITCH REGISTER THEN JUST HIT CARRIAGE RETURN. IF CHANGES ARE REQUIRED, THEN A NEW VALUE MAY BE TYPED FOLLOWED BY A CARRIAGE RETURN. REFER TO SECTION 5.0 FOR SWITCH REGISTER OPTIONS.

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

THE PROGRAM SWITCH DEFAULT MODE IS SWR = 000000  
IF USING A VIDEO TERMINAL, BIT 15 = 1 (HALT ON ERROR),  
MAY BE HELPFUL IN KEEPING THE ERROR ON THE SCREEN.

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW12=1	010000	INHIBIT ILLEGAL VECTOR RETURN ROUTINES
SW11=1	004000	INHIBIT ITERATIONS
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <7-0>

\* SW12 EXPLANATION

SW12 = 1 WILL STORE THE REGULAR TRAP (PC+2, HALT) IN LOCATIONS 204-1774 FOR THE VECTOR TESTING. ON ILLEGAL VECTOR ERRORS, THIS WILL AID THE USER TO DETERMINE THE LOCATION OF THE ILLEGAL VECTOR. THE LOOPING CAPABILITIES THAT ARE NORMALLY PROVIDED BY ILLEGAL VECTOR SERVICE ROUTINES WHEN SW12=0 WILL NO LONGER BE AVAILABLE WHEN SW12 = 1.

IF USING SW12=1 TO TRACK DOWN THE ILLEGAL VECTOR, THE  
USER SHOULD RESTART AT ADDRESS 200 WITH SW12=0 TO  
OBTAIN LOOPING CAPABILITIES AFTER ERROR OCCURS.  
VECTOR AREA 0-200 WILL ALWAYS HAVE ILLEGAL VECTOR ROUTINES  
WHEN VECTOR TESTING (0-200) IN ORDER TO PRESERVE PROGRAM.

**\* WARNING:**

PLEASE USE "CTRL AND G" KEYS AND WAIT FOR THE "CTRL G" TO BE ECHOED BEFORE ATTEMPTING TO HALT THE CPU OR TO STOP TESTING OR THE PROGRAM MAY NOT BE RESTARTABLE.  
PROGRAM RELOCATION FOR LOCATIONS 0-200 TAKES PLACE DURING THE VECTOR TESTING OF 0-200. THE NORMAL CONTENTS OF 0-200 MAY NOT GET RESTORED IF THE CPU IS HALTED AND RELOAD OF THE PROGRAM MAY BE NECESSARY.

1. THE SOFTWARE SWITCH REGISTER 'SWREG' (LOC. 176) CAN BE CHANGED BY USING THE ODT FACILITIES.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CONTROL & G' KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.
3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFTWARE SWITCH REGISTER IF DESIRED BEFORE TYPING 'P' (CONTINUE).

**6.0 ERROR REPORTING**  
-----**6.1 ERROR COMMENT**

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF.

**6.2 ERROR DATA****6.2.1 ERROR TITLE TYPE A**

1. REG READ/WRITE ER
2. IRR REG ER
3. ACR REG ER
4. IMR REG ER
5. ISR REG ER

## 6. CHIP STAT ER

## 7. MULTIPLE INTERRUPTS RECEIVED

## 8. INTERRUPT TEST ERROR

ERRPC	TSTNUM	BUSADR	VAM	ADDR	EXPCT	RCVD
XXXXXX	XXXXXX	XXXXXX	XXXXXX	XXXXXX	XXXXXX	XXXXXX

ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
TSTNUM	TEST NUMBER WHERE THE ERROR OCCURRED
BUSADR	DRV11J BUS REG ADDRESS OF CONCERNED OPERATION
VAM	VALUE STORED INTO VECTOR ADDRESS MEMORY INDICATING BYTE COUNT AND LEVEL.
ADDR	VECTOR ADDRESS
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

## 6.2.2 ERROR TITLE TYPE B

## 1. ILLEGAL INTERRUPT RECEIVED

ERRPC	TSTNUM	BUSADR	VAM	ADDR
XXXXXX	XXXXXX	XXXXXX	XXXXXX	XXXXXX

ERRPC	ADDRESS WHERE THE ERROR WAS DETECTED
TSTNUM	TEST NUMBER WHERE THE ERROR OCCURRED
BUSADR	DRV11J BUS ADDRESS
VAM	VALUE OF VECTOR ADDRESS MEMORY AT TIME OF ILLEGAL INTERRUPT.
ADDR	VECTOR ADDRESS

## 6.2.3 ERROR TITLE TYPE C

## 1. ILLEGAL VECTOR MEM ADDR ER

ERRPC	TSTNUM	TESTPC	BUSADR	VAM	ADDR
XXXXXX	XXXXXX	XXXXXX	XXXXXX	XXXXXX	XXXXXX

ERRPC	ADDRESS WHERE ERROR WAS DETECTED
TSTNUM	TEST NUMBER WHERE THE ERROR OCCURED
TESTPC	ADDRESS OF TEST THAT WAS RUNNING WHEN DRV11J VECTORED TO THE WRONG ADDRESS.
BUSADR	BUS ADDRESS OF CONCERNED OPERATION
VAM	VALUE OF VECTOR ADDRESS MEMORY BYTE COUNT AND LEVEL.
ADDR	CORRECT VECTOR ADDRESS

## 6.2.4 ERROR TYPE D

THIS ERROR IS ONLY FOUND WHEN RUNNING THE VECTOR ADDRESS UNIQUENESS TESTS. THE VECTOR MEMORY IS PRESET TO PERFORM 64 INTERRUPTS. INTERRUPT SERVICE ROUTINES ARE STORED FROM VECTORS 300 TO 674. THE ROUTINES WILL STORE THE VALUE OF THE VECTOR ADDRESS WHERE THEY CAME FROM IN A BUFFER TO COMPARE AFTER ALL INTERRUPTS TAKE PLACE. THE BUFFER SHOULD THEN HAVE ADDRESS VALUES IN CONSECUTIVE ORDER STARTING WITH VECTOR VALUE 300 AND ENDING WITH VECTOR VALUE 674.



1. VECT ADDR MEM ER  
 ERRPC TSTNUM BUSADR EXPCT RCVD  
 XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX

ERRPC ADDRESS WHERE ERROR WAS DETECTED  
 TSTNUM TEST NUMBER WHERE ERROR OCCURRED  
 BUSADR STARTUP BUS ADDRESS BEFORE VECTOR UNIQUENESS  
 EXPCT THIS IS THE EXPECTED VECTOR ADDRESS  
 RCVD VECTOR ADDRESS VALUE STORED IN BUFFER

## 7.0 MISCELLANEOUS

### 7.1 DRV11-J BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE'(ADDR:2244) IF BASE BUS ADDRESS IS NOT 164160.

### 7.2 XXDP/APT NOTES

THIS DIAGNOSTIC DOES SUPPORT ACT AND APT ENVIRONMENTS.

### 7.3 POWER FAIL

A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).

### 7.4 MULTIPLE DRV11J INTERFACE TESTING

THIS PROGRAM DOES NOT "AUTO-SIZE" THE NUMBER OF DRV11J'S CONNECTED. THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 4 DRV11J INTERFACES WITH CONTIGUOUS BUS ADDRESSES. THIS IS ACCOMPLISHED BY THE USER SETTING UP LOCATION '\$DEVN'(ADDR:2246) WITH A BIT MAP INDICATING WHAT INTERFACES ARE TO BE TESTED.

I.E. BIT0= 1 SAYS TEST 1ST DRV11J,  
 BIT1= 1 SAYS TEST 1ST DRV11J  
 BIT2= 1 SAYS TEST 2ND DRV11J  
 BIT3= 1 SAYS TEST 3RD DRV11J

1ST UNIT = STARTING CSRA	164160	\$DEVN = 1
2ND UNIT = STARTING CSRA	164140	\$DEVN = 3
3RD UNIT = STARTING CSRA	164120	\$DEVN = 7
4TH UNIT = STARTING CSRA	164100	\$DEVN = 17

### 7.5 RESTRICTIONS

ALWAYS TYPE "CTRL G" AND WAIT FOR ECHO OF CHARACTERS BEFORE HALTING THE CPU OTHERWISE PORTIONS OF THE PROGRAM MAY NOT GET RESTORED. FOR MORE INFORMATION REFER TO SECTION 5.2.

### 8.0 EXECUTION TIME

EXECUTION TIME WILL BE LESS THAN 30 SECONDS PER DRV11J UNIT CONNECTED. AN "END PASS" MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS.

**GENERAL**

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE DRV11J OPTION. TESTING IS ACCOMPLISHED WITH THE AID OF THE DRV11J LOOP BACK CABLE PROVIDED FOR DIAGNOSTIC TESTING. A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING.

REFERENCES ARE MADE IN THE LISTING TO THE TWO INTERRUPT CONTROL CHIPS USED ON THE DRV11J AS GROUP1 AND GROUP2.

GROUP1 CHIP            CSRA = CONTROL FOR GROUP 1  
                         CSRB = DATA FOR GROUP1  
GROUP2 CHIP            CSRC = CONTROL FOR GROUP2  
                         CSRD = DATA FOR GROUP2

**9.1 TEST 1 - TEST CHIP INTERRUPT, GROUP1, GROUP2**

THIS TEST WILL CHECK THE ABILITY OF EACH INTERRUPT CONTROL CHIP TO INTERRUPT TO COMMON VECTOR 300. THIS TEST WILL ALSO CLEAR THE HIGHEST PRIORITY ISR BITS WITH TEST VARIATIONS OF IRR AND IMR BITS.

**9.2 TEST 2 - TEST ACR/ISR INTERRUPT, GROUP 1, GROUP 2**

THIS TEST WILL TEST THE ABILITY OF BOTH GROUPS TO INTERRUPT TO COMMON VECTOR 300 AND THE ABILITY OF THE ACR(AUTOCLEAR REGISTER TO CLEAR THE ISR AFTER BEING SET AND MASKED BY COMBINATIONS OF IRR BITS AND IMR BITS.

**9.3 TEST 3 - TEST VECTOR ADDR MEM, LEV 0-7, BY0-3, (204-1774), GRPS 1,2**

THIS TEST WILL CHECK BOTH GROUPS, ONE AT A TIME, TO VECTOR FROM ADDRESS 204 TO ADDRESS 1774 FOR LEVELS 0-7 AND FOR BYTE COUNTS 1-4. THE BYTE COUNT NUMBER WILL INDICATE THE NUMBER OF INTERRUPTS AT A GIVEN ADDRESS. THE IRR, IMR AND ACR REGISTERS WILL ALL BE EXERCISED AND CHECKED TO INSURE PROPER VECTORING.

**9.4 TEST 4 - TEST VECTOR ADDR MEM, LEV 0-7, BY0-3, (0-200), GRPS 1,2**

THIS TEST WILL CHECK BOTH GROUPS, ONE AT A TIME, TO VECTOR FROM ADDRESS 0-200 FOR LEVELS 0-7 AND FOR BYTE COUNTS 1-4. TESTS ARE DONE TO INSURE THAT INTERRUPTS ONLY TAKE PLACE WHEN EVERYTHING IS READY. THE CHIP BEING ARMED, INTERRUPT ENABLE IS SET AND THE PROPER IMR AND IRR BITS ARE SET. AFTER THE INTERRUPTS CHECK TO INSURE THAT THE INTERRUPTS GENERATED MATCH THE BYTE COUNT AND THAT THE PROPER ISR BITS WERE SET AND THE PROPER IRR BITS WERE CLEARED.

**9.5 TEST 5 - TEST VECTOR ADDR UNIQUENESS IN FIXED MODE FOR GROUPS 1,2**

THIS TEST WILL CHECK THE FIXED PRIORITY OF THE TWO INTERRUPT CONTROL CHIPS AS WELL AS THE ADDRESSING CAPABILITY OF THE VECTOR ADDRESS MEMORY. THE AUTOCLEAR REGISTER(ACR) IS USED TO CLEAR OUT THE ISR BITS AFTER 4 INTERRUPTS PER LEVEL HAS TAKEN PLACE. CLEARING THE ISR REGISTER WILL THEN ALLOW THE NEXT LEVEL OF PRIORITY TO INITIATE 4 INTERRUPTS. THE VECTOR MEMORY IS PRESET TO PERFORM 64 UNIQUE INTERRUPTS. INTERRUPT SERVICE ROUTINES ARE STORED FROM VECTORS 300-674. EACH INTERRUPT SERVICE ROUTINE UPON INTERRUPT ENTRY WILL STORE THE ADDRESS OF ITS VECTOR IN A BUFFER FOR A COMPARE AFTER 64 INTERRUPTS TAKE PLACE. THE BUFFER SHOULD THEN HAVE ADDRESS VALUES IN CONSECUTIVE ORDER STARTING WITH VECTOR VALUE 300 AND ENDING WITH VECTOR VALUE 674.

**9.6 TEST 6 - TEST VECTOR ADDRESS UNIQUENESS,ROTATING PRIORITY FOR BOTH GROUPS 1,2**

THIS TEST WILL CHECK THE ROTATING PRIORITY OF EACH OF THE TWO INTERRUPT CONTROL CHIPS AS WELL AS THE ADDRESSING CAPABILITY OF THE VECTOR ADDRESS MEMORY. THE VECTOR ADDRESS MEMORY IS PRESET FOR 64 UNIQUE INTERRUPTS. INTERRUPT SERVICE ROUTINES ARE STORED FROM VECTORS 300-674. ALL IRR BITS ARE SET AND ALL IMR BITS ARE CLEARED. AFTER THE FIRST FOUR INTERRUPTS FROM IRRO TAKES PLACE, THE ISRO BIT WILL SET AND IRRO BIT WILL CLEAR. TO PROVE ROTATION OF PRIORITY,THE ISRO BIT IS CLEARED AND THE IRRO BIT IS SET AGAIN IN ORDER TO PROVE THAT IRR1 WILL NOW BE CONSIDERED THE HIGHEST PRIORITY AND IRRO THE LOWEST PRIORITY. THIS FLOW IS REPEATED FOR ALL LEVELS OF GROUP 1(0-7). THEN GROUP 2 IS TESTED IN THE SAME MANNER FOR LEVELS 0-7. EACH INTERRUPT SERVICE ROUTINE UPON ACCESS FROM AN INTERRUPT WILL STORE THE ADDRESS OF THE VECTOR WHERE THE ROUTINE RESIDES IN A BUFFER. AFTER ALL 64 INTERRUPTS TAKE PLACE(4 AT A TIME FOR EACH LEVEL)THE BUFFER WILL BE CHECKED FOR CONSECUTIVE VECTOR VALUES OF 300-674. THIS WILL INSURE THAT PROPER VECTOR ADDRESSING AND ROTATION OF PRIORITIES WAS COMPLETED.

**10.0 LISTING**  
-----



```

(1)          000006      SP=      %6          ::STACK POINTER
(1)          000007      PC=      %7          ::PROGRAM COUNTER
(1)
(1)          000000     ::*PRIORITY LEVEL DEFINITIONS
(1)          000040      PR0=      0          ::PRIORITY LEVEL 0
(1)          000100      PR1=      40         ::PRIORITY LEVEL 1
(1)          000140      PR2=      100        ::PRIORITY LEVEL 2
(1)          000200      PR3=      140        ::PRIORITY LEVEL 3
(1)          000240      PR4=      200        ::PRIORITY LEVEL 4
(1)          000300      PR5=      240        ::PRIORITY LEVEL 5
(1)          000340      PR6=      300        ::PRIORITY LEVEL 6
(1)          000340      PR7=      340        ::PRIORITY LEVEL 7
(1)
(1)          100000     ::*"SWITCH REGISTER" SWITCH DEFINITIONS
(1)          040000      SW15=     100000
(1)          020000      SW14=     40000
(1)          010000      SW13=     20000
(1)          004000      SW12=     10000
(1)          002000      SW11=     4000
(1)          001000      SW10=     2000
(1)          000400      SW09=     1000
(1)          000200      SW08=     400
(1)          000100      SW07=     200
(1)          000040      SW06=     100
(1)          000020      SW05=     40
(1)          000010      SW04=     20
(1)          000004      SW03=     10
(1)          000002      SW02=     4
(1)          000001      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)          100000     ::*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)          040000      BIT15=    100000
(1)          020000      BIT14=    40000
(1)          010000      BIT13=    20000
(1)          004000      BIT12=    10000
(1)          002000      BIT11=    4000
(1)          001000      BIT10=    2000
(1)          000400      BIT09=    1000
(1)          000200      BIT08=    400
(1)          000100      BIT07=    200
(1)          000040      BIT06=    100
(1)          000020      BIT05=    40
(1)          000010      BIT04=    20
(1)          000004      BIT03=    10
(1)          000001      BIT02=    4

```

(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49

000002  
000001  
  
000004  
000010  
000014  
000014  
000014  
000020  
000024  
000030  
000034  
000060  
000064  
000240  
164160  
000001  
100000  
000400  
001000  
  
000020  
000030  
  
000040  
000050  
000060  
000070  
  
000100  
000110  
000120  
000130  
  
000140  
000160  
000170  
  
000200  
000240  
  
000240  
000244

BIT01= 2  
BIT00= 1  
.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0

;\*BASIC "CPU" TRAP VECTOR ADDRESSES  
ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS  
RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS  
TBITVEC=14 ;: "T" BIT  
TRTVEC= 14 ;:TRACE TRAP  
BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)  
IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
PWRVEC= 24 ;:POWER FAIL  
EMTVEC= 30 ;:EMULATOR TRAP (EMT) \*\*ERROR\*\*  
TRAPVEC=34 ;: "TRAP" TRAP  
TKVEC= 60 ;:TTY KEYBOARD VECTOR  
TPVEC= 64 ;:TTY PRINTER VECTOR  
PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR  
ABASE= 164160 ;:BASE ADDRESS  
ADEVM= 1 ;:DEFAULT TO ONE DRV11J  
RDY= BIT15  
DIR= BIT8  
IE= BIT9

;CHIP COMMAND SUMMARY  
CIRR= 20 ;:CLEAR IRR AND IMR  
CSIRR= 30 ;:CLEAR SINGLE IRR AND IMR BIT  
  
CIMR= 40 ;:CLEAR IMR  
CSIMR= 50 ;:CLEAR SINGLE IMR BIT  
SISR= 60 ;:SET ALL IMR BITS  
SSISR= 70 ;:SET SINGLE IMR BITS  
  
CIRR= 100 ;:CLEAR IRR  
CSIRR= 110 ;:CLEAR SINGLE IRR BITS  
SIRR= 120 ;:SET ALL IRR BITS  
SSIRR= 130 ;:SET SINGLE IRR BITS  
  
CHPISR= 140 ;:CLEAR HIGHEST PRIORITY ISR BIT  
CISR= 160 ;:CLEAR ISR  
CSISR= 170 ;:CLEAR SINGLE ISR BIT  
  
LMD04= 200 ;:LOAD MODE BITS M0-M4  
LMD57= 240 ;:LOAD MODE BITS M5-M7

;CHIP MODE BIT PRESELECTION  
MISR= 240  
MIMR= 244

```
50          000250          MIRR= 250
51          000254          MACR= 254
52
53          ;CHIP WRITE PRESELECTION
54          000300          PACR= 300          ;PRESELECT AUTO CLEAR REG. FOR WRITING
55          000260          PIMR= 260          ;PRESELECT IMR REG. FOR WRITING
56          000340          PVMA= 340          ;PRESELECT VECTOR MEMORY ADDRESS
57
58          .SBTTL TRAP CATCHER
(1)
(1)          000000          .=0
(1)          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)          000174          .=174
(1) 000174 000000          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000          SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER
(1)          .SBTTL STARTING ADDRESS(ES)
(1) 000200 000137 002476          JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
59          000100          .=100
60 000100 000104 000200 000002          .WORD 104,200,2          ;IF 'B EVENT' ON Q BUS IS CONNECTED
61          ;IGNORE IT'S INTERRUPT - JUST DO A RTI
```

```

63
(1)
(2)
(1)
(1)      000106
(1)      000046
(1) 000046 010760
(1)      000052
(1) 000052 000000
(1)      000106
(1)      002000
64
65
66
67
68
(1)
(2)
(1)
(2)
(1)      002000
(1)      000024
(1) 000024 000200
(1)      000044
(1) 000044 002000
(1)      002000
(2)
(1)
(1)
(1)
(1) 002000
(1) 002000 000000
(1) 002002 002170
(1) 002004 000014
(1) 002006 000036
(1) 002010 000036
(1) 002012 000031

```

```

.SBTTL ACT11 HOOKS
:*****
:HOOKS REQUIRED BY ACT11
      $SVPC=           ;SAVE PC
      .=46             ;:1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
      $ENDAD           ;:2)SET LOC.52 TO ZERO
      .=52
      .WORD 0          ;: RESTORE PC
      .=$SVPC
      .=2000
                               ;LONGEST TEST TIME
                               ;1ST PASS RUN TIME
                               ;ADDITIONAL RUN TIME

.SBTTL APT PARAMETER BLOCK
:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
      .$X=            ;;SAVE CURRENT LOCATION
      .=24            ;;SET POWER FAIL TO POINT TO START OF PROGRAM
      200            ;;FOR APT START UP
      .=44            ;;POINT TO APT INDIRECT ADDRESS PNTR.
      $APTHDR        ;;POINT TO APT HEADER BLOCK
      .=$X            ;;RESET LOCATION COUNTER
:*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.

$APTHD:
$SHBTS: .WORD 0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT: .WORD 12       ;;RUN TIME OF LONGEST TEST
$PASTM: .WORD 30      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 30      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```



69

(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1) 002100 002100  
(1) 002100 000000  
(1) 002102 000  
(1) 002103 000  
(1) 002104 000000  
(1) 002106 000000  
(1) 002110 000000  
(1) 002112 000000  
(1) 002114 000  
(1) 002115 001  
(1) 002116 000000  
(1) 002120 000000  
(1) 002122 000000  
(1) 002124 000000  
(1) 002126 000000  
(1) 002130 000000  
(1) 002132 000000  
(1) 002134 000  
(1) 002135 000  
(1) 002136 000000  
(1) 002140 177570  
(1) 002142 177570  
(1) 002144 177560  
(1) 002146 177562  
(1) 002150 177564  
(1) 002152 177566  
(1) 002154 000  
(1) 002155 002  
(1) 002156 012  
(1) 002157 000  
(1) 002160 000000  
(1) 002162 000000  
(1) 002164 077  
(1) 002165 015  
(1) 002166 000012  
(2)  
(2)  
(2)  
(3)  
(2)  
(2) 002170  
(2) 002170 000000  
(2) 002172 000000  
(2) 002174 000000  
(2) 002176 000000  
(2) 002200 000000  
(2) 002202 000000  
(2) 002204 000000  
(2) 002206 000000

.SBTTL COMMON TAGS

::\*\*\*\*\*  
:\*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
:\*USED IN THE PROGRAM.

SCMTAG: .=2100                    :::START OF COMMON TAGS  
\$STNM: .WORD 0                    :::CONTAINS THE TEST NUMBER  
\$ERFLG: .BYTE 0                  :::CONTAINS ERROR FLAG  
\$ICNT: .WORD 0                    :::CONTAINS SUBTEST ITERATION COUNT  
\$LPADR: .WORD 0                  :::CONTAINS SCOPE LOOP ADDRESS  
\$LPERR: .WORD 0                  :::CONTAINS SCOPE RETURN FOR ERRORS  
\$ERTTL: .WORD 0                  :::CONTAINS TOTAL ERRORS DETECTED  
\$ITEMB: .BYTE 0                  :::CONTAINS ITEM CONTROL BYTE  
\$ERMAX: .BYTE 1                  :::CONTAINS MAX. ERRORS PER TEST  
\$ERRPC: .WORD 0                  :::CONTAINS PC OF LAST ERROR INSTRUCTION  
\$GDADR: .WORD 0                  :::CONTAINS ADDRESS OF 'GOOD' DATA  
\$BDADR: .WORD 0                  :::CONTAINS ADDRESS OF 'BAD' DATA  
\$GDDAT: .WORD 0                  :::CONTAINS 'GOOD' DATA  
\$BDDAT: .WORD 0                  :::CONTAINS 'BAD' DATA  
          .WORD 0                  :::RESERVED--NOT TO BE USED  
          .WORD 0  
\$AUTOB: .BYTE 0                  :::AUTOMATIC MODE INDICATOR  
\$INTAG: .BYTE 0                  :::INTERRUPT MODE INDICATOR  
          .WORD 0  
\$SWR: .WORD DSWR                 :::ADDRESS OF SWITCH REGISTER  
\$DISPLAY: .WORD DDISP            :::ADDRESS OF DISPLAY REGISTER  
\$TKS: 177560                    :::TTY KBD STATUS  
\$TKB: 177562                    :::TTY KBD BUFFER  
\$TPS: 177564                    :::TTY PRINTER STATUS REG. ADDRESS  
\$TPB: 177566                    :::TTY PRINTER BUFFER REG. ADDRESS  
\$NULL: .BYTE 0                  :::CONTAINS NULL CHARACTER FOR FILLS  
\$FILLS: .BYTE 2                  :::CONTAINS # OF FILLER CHARACTERS REQUIRED  
\$FILLC: .BYTE 12                :::INSERT FILL CHARS. AFTER A 'LINE FEED'  
\$STPFLG: .BYTE 0                :::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)  
\$TIMES: 0                        :::MAX. NUMBER OF ITERATIONS  
\$ESCAPE:0                        :::ESCAPE ON ERROR ADDRESS  
\$QUES: .ASCII /?/                :::QUESTION MARK  
\$CR LF: .ASCII <15>            :::CARRIAGE RETURN  
\$LF: .ASCII <12>                :::LINE FEED

::\*\*\*\*\*  
.SBTTL APT MAILBOX-ETABLE

::\*\*\*\*\*  
.EVEN

\$MAIL:                            :::APT MAILBOX  
\$MSGTY: .WORD AMSGTY            :::MESSAGE TYPE CODE  
\$FATAL: .WORD AFATAL            :::FATAL ERROR NUMBER  
\$TESTN: .WORD ATESTN            :::TEST NUMBER  
\$PASS: .WORD APASS              :::PASS COUNT  
\$DEVCT: .WORD ADEVCT            :::DEVICE COUNT  
\$SUNIT: .WORD AUNIT             :::I/O UNIT NUMBER  
\$MSGAD: .WORD AMSGAD            :::MESSAGE ADDRESS  
\$MSGLG: .WORD AMSGLG            :::MESSAGE LENGTH

```
(2) 002210          SETABLE:          ::APT ENVIRONMENT TABLE
(2) 002210          $ENV: .BYTE      AENV          ::ENVIRONMENT BYTE
(2) 002211          $ENVN: .BYTE     AENVN         ::ENVIRONMENT MODE BITS
(2) 002212          $$WREG: .WORD    ASWREG        ::APT SWITCH REGISTER
(2) 002214          $USWR: .WORD    AUSWR         ::USER SWITCHES
(2) 002216          $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) 002220          $MAMS1: .BYTE    AMAMS1        ::HIGH ADDRESS,M.S. BYTE
(2) 002221          $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) 002222          $MADR1: .WORD    AMADR1        ::HIGH ADDRESS,BLK#1
(2)                : *          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2) 002224          $MAMS2: .BYTE    AMAMS2        ::HIGH ADDRESS,M.S. BYTE
(2) 002225          $MTYP2: .BYTE    AMTYP2        ::MEM. TYPE,BLK#2
(2) 002226          $MADR2: .WORD    AMADR2        ::MEM.LAST ADDRESS,BLK#2
(2) 002230          $MAMS3: .BYTE    AMAMS3        ::HIGH ADDRESS,M.S.BYTE
(2) 002231          $MTYP3: .BYTE    AMTYP3        ::MEM. TYPE,BLK#3
(2) 002232          $MADR3: .WORD    AMADR3        ::MEM.LAST ADDRESS,BLK#3
(2) 002234          $MAMS4: .BYTE    AMAMS4        ::HIGH ADDRESS,M.S.BYTE
(2) 002235          $MTYP4: .BYTE    AMTYP4        ::MEM. TYPE,BLK#4
(2) 002236          $MADR4: .WORD    AMADR4        ::MEM.LAST ADDRESS,BLK#4
(2) 002240          $VECT1: .WORD    AVECT1        ::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 002242          $VECT2: .WORD    AVECT2        ::INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 002244          $BASE: .WORD    ABASE         ::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 002246          $DEVN: .WORD    ADEVN         ::DEVICE MAP
(2) 002250          $CDW1: .WORD    ACDW1         ::CONTROLLER DESCRIPTION WORD#1
(2) 002252          $ETEND:
(2)                .MEXIT
```

(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
002252  
70  
71 002252 016101  
72 002254 016416  
73 002256 016666  
74 002260 000000  
75  
76  
77 002262 016120  
78 002264 016463  
79 002266 016702  
80 002270 000000  
81  
82  
83 002272 016142  
84 002274 016463  
85 002276 016702  
86 002300 000000  
87  
88  
89 002302 016155  
90 002304 016463  
91 002306 016702  
92 002310 000000  
93  
94  
95 002312 016170  
96 002314 016463  
97 002316 016702  
98 002320 000000  
99  
100  
101 002322 016203  
102 002324 016463  
103 002326 016702  
104 002330 000000  
105  
106  
107 002332 016375  
108 002334 016416  
109 002336 016666  
110 002340 000000

.SBTTL ERROR POINTER TABLE

:\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
:\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
:\*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
:\*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
:\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

::\* EM ::POINTS TO THE ERROR MESSAGE  
::\* DH ::POINTS TO THE DATA HEADER  
::\* DT ::POINTS TO THE DATA  
::\* DF ::POINTS TO THE DATA FORMAT

\$ERRTB:  
:ERROR 1 :REG TIMEOUT ER  
EM1 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DH1 :SERRPC TSTNUM SBDADR SGDDAT SBDDAT  
DT1  
0  
:ERROR 2 :REG READ/WRITE ER  
EM2 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD  
DH2 :SERRPC TSTNUM SBDADR VECVAL VECLOC SGDDAT SBDDAT  
DT2  
0  
:ERROR 3 :IRR REG ER  
EM3 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD  
DH2 :SERRPC TSTNUM SBDADR VECVAL VECLOC SGDDAT SBDDAT  
DT2  
0  
:ERROR 4 :ACR REG ER  
EM4 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD  
DH2 :SERRPC TSTNUM SBDADR VECVAL VECLOC SGDDAT SBDDAT  
DT2  
0  
:ERROR 5 :IMR REG ERROR  
EM5 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD  
DH2 :SERRPC TSTNUM SBDADR VECVAL VECLOC SGDDAT SBDDAT  
DT2  
0  
:ERROR 6 :ISR REG ERROR  
EM6 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD  
DH2 :SERRPC TSTNUM SBDADR VECVAL VECLOC SGDDAT SBDDAT  
DT2  
0  
:ERROR 7 :VECTOR ADDR MEM ER  
EM14 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DH1 :SERRPC TSTNUM SBDADR SGDDAT SBDDAT  
DT1  
0

```

111
112      :ERROR 10
113      002342 016251      EM10      :CHIP STAT ER
114      002344 016463      DH2       :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
115      002346 016702      DT2       :SERRPC TSTNUM $BDADR VECVAL VECLOC $GDDAT $BDDAT
116      002350 000000      0
117
118      :ERROR 11
119      002352 016266      EM11      :ILLEGAL INTERRUPT RECEIVED
120      002354 016550      DH3       :ERRPC TSTNUM BUSADR VAM ADDR
121      002356 016722      DT3       :SERRPC TSTNUM $BDADR VECVAL VECLOC
122      002360 000000      0
123
124      :ERROR 12
125      002362 016316      EM12      :INTERRUPT TEST ERROR
126      002364 016463      DH2       :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
127      002366 016702      DT2       :SERRPC TSTNUM $BDADR VECVAL VECLOC $GDDAT $BDDAT
128      002370 000000      0
129
130      :ERROR 13
131      002372 016343      EM13      :MULTIPLE INTERRUPTS RECEIVED
132      002374 016463      DH2       :MULTIPLE INTERRUPTS RECEIVED
133      002376 016702      DT2       :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
134      002400 000000      0       :SERRPC TSTNUM $BDADR VECVAL VECLOC $GDDAT $BDDAT
135
136      :ERROR 14
137      002402 016216      EM7       :ILLEGAL VECTOR ADDR MEM ERROR
138      002404 016615      DH4       :ERRPC TSTNUM TESTPC BUSADR VAM ADDR
139      002406 016736      DT4       :SERRPC TSTNUM $GDADR $BDADR VECVAL VECLOC
140      002410 000000      0
141
142
143      : BUS REGISTER ADDRESS POINTERS
144
145
146      DRCSA: ABASE
147      DRDBA: ABASE+2
148      DRCSB: ABASE+4
149      DRDBB: ABASE+6
150      DRCSC: ABASE+10
151      DRDBC: ABASE+12
152      DRCSD: ABASE+14
153      DRDBD: ABASE+16
154
155
156      :COMMON PROGRAM LOCATION(S)
157
158      TSTNUM: 0 ;CONTAINS TEST NUMBER ON ERROR
159      DMAP: 1
160      INTFLG: .WORD 0
161      XXDP: .WORD 0
162      IMRLOC: .WORD 0
163      ISRLOC: .WORD 0
164      IRRLOC: .WORD 0
165      ACRLOC: .WORD 0
166      VECLOC: .WORD 0

```

167	002454	000000	VECPAT: .WORD	0
168	002456	000000	VECVL: .WORD	0
169	002460	000000	SWRSV: .WORD	0
170	002462	000000	GRPCNT: .WORD	0
171	002464	000000	BDSAV: .WORD	0
172	002466	000000	LVLCNT: .WORD	0
173	002470	000000	BYCNT: .WORD	0
174	002472	000000	BYNUM: .WORD	0
175	002474	000000	LVLSAV: .WORD	0

```

178          .SBTTL PROGRAM START
179 002476   START:
(1)          .SBTTL INITIALIZE THE COMMON TAGS
(1)          ::CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 002476 012706 002100   MOV    #SCMTAG,R6      ::FIRST LOCATION TO BE CLEARED
(1) 002502 005026          CLR    (R6)+           ::CLEAR MEMORY LOCATION
(1) 002504 022706 002140   CMP    #SWR,R6      ::DONE?
(1) 002510 001374          BNE   -6             ::LOOP BACK IF NO
(1) 002512 012706 002100   MOV    #2100,SP     ::SETUP THE STACK POINTER
(1)          ::INITIALIZE A FEW VECTORS
(1) 002516 012737 014462 000020   MOV    #$$SCOPE,@#IOTVEC ::IOT VECTOR FOR SCOPE ROUTINE
(1) 002524 012737 000340 000022   MOV    #340,@#IOTVEC+2 ::LEVEL 7
(1) 002532 012737 014134 000030   MOV    #ERROR,@#EMTVEC  ::EMT VECTOR FOR ERROR ROUTINE
(1) 002540 012737 000340 000032   MOV    #340,@#EMTVEC+2 ::LEVEL 7
(1) 002546 012737 015732 000034   MOV    #STRAP,@#TRAPVEC ::TRAP VECTOR FOR TRAP CALLS
(1) 002554 012737 000340 000036   MOV    #340,@#TRAPVEC+2::LEVEL 7
(1) 002562 012737 015526 000024   MOV    #SPWRDN,@#PWRVEC ::POWER FAILURE VECTOR
(1) 002570 012737 000340 000026   MOV    #340,@#PWRVEC+2 ::LEVEL 7
(1) 002576 005037 002160          CLR    $TIMES       ::INITIALIZE NUMBER OF ITERATIONS
(1) 002602 005037 002162          CLR    $ESCAPE     ::CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002606 112737 000001 002115   MOV    #1,$SERMAX   ::ALLOW ONE ERROR PER TEST
(1) 002614 012737 002614 002106   MOV    #.,$SLPADR   ::INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002622 012737 002622 002110   MOV    #.,$SLPERR   ::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) 002630 013746 000004          MOV    @#ERRVEC,-(SP) ::SAVE ERROR VECTOR
(2) 002634 012737 002670 000004   MOV    #64$,@#ERRVEC ::SET UP ERROR VECTOR
(2) 002642 012737 177570 002140   MOV    #DSWR,SWR    ::SETUP FOR A HARDWARE SWICH REGISTER
(2) 002650 012737 177570 002142   MOV    #DDISP,DISPLAY ::AND A HARDWARE DISPLAY REGISTER
(2) 002656 022777 177777 177254   CMP    #-1,@SWR    ::TRY TO REFERENCE HARDWARE SWR
(2) 002664 001012          BNE   66$          ::BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)          ::AND THE HARDWARE SWR IS NOT = -1
(2) 002666 000403          BR    65$          ::BRANCH IF NO TIMEOUT
(2) 002670 012716 002676 64$:   MOV    #65$, (SP)  ::SET UP FOR TRAP RETURN
(2) 002674 000002          RTI
(2) 002676 012737 000176 002140 65$:   MOV    #SWREG,SWR  ::POINT TO SOFTWARE SWR
(2) 002704 012737 000174 002142   MOV    #DISPREG,DISPLAY
(2) 002712 012637 000004 66$:   MOV    (SP)+,@#ERRVEC ::RESTORE ERROR VECTOR
(1)          CLR    $PASS      ::CLEAR PASS COUNT
(2) 002716 005037 002176          BITB  #APTSIZE,$ENVM ::TEST USER SIZE UNDER APT
(2) 002722 132737 000200 002211   BEQ   67$          ::YES,USE NON-APT SWITCH
(2) 002730 001403          MOV    #$$SWREG,SWR ::NO,USE APT SWITCH REGISTER
(2) 002732 012737 002212 002140 67$:
(2) 002740          CLR    $FATAL     ::CLEAR ERROR NUMBER
180 002740 005037 002172          CLR    $MSGTYP     ::CLEAR MESSAGE TYPE
181 002744 005037 002170          CLR    $TESTN     ::CLEAR TEST NUMBER
182 002750 005037 002174          CALL  FALCON      :: CHECK FOR FALCON (KXT11)      ::GPA
183 002754 004737 017360          BEQ   1000$       :: BR IF NOT KXT11 SYSTEM      ::GPA
184 002760 001420          BIC   #40,@#22    :: FALCON, LOWER IOT...      ::GPA
185 002762 042737 000040 000022   BIC   #40,@#32    ::...EMT...                  ::GPA
186 002770 042737 000040 000032   BIC   #40,@#36    ::...AND TRAP TO LEVEL 6.    ::GPA
187 002776 042737 000040 000036   CMP   $BASE,#ABASE :: IS $BASE VIRGIN ??      ::GPA
188 003004 023727 002244 164160   BNE   1000$       :: BR IF NOT.                ::GPA
189 003012 001003          MOV   #174160,$BASE :: YES, USE ENGINEERING DEFAULT ::GPA
190 003014 012737 174160 002244 1000$:
191 003022

```

```

192                                     :CHECK OPERATING ENVIRONMENT
193 003022 005737 000042                TST    @#42                :ARE WE IN ACT/XXDP AUTO MODE?
194 003026 001410                        BEQ    1$                  :BRANCH IF NO
195 003030 023737 000042 000046        CMP    @#42,@#46          :IS IT ACT AUTO MODE?
196 003036 001410                        BEQ    2$                  :BRANCH IF YES
197 003040 012737 177777 002440        MOV    #-1,XXDP          :SET XXDP CHAIN MODE INDICATOR
198 003046 000404                        BR     2$
199 003050 123727 002210 000001 1$:    CMPB  $ENV,#1            :ARE WE IN APT AUTO MODE?
200 003056 001003                        BNE   3$                  :BRANCH IF NO
201 003060 112737 000001 002134 2$:    MOVB  #1,$AUTOB         :SET AUTO MODE INDICATOR
202
203                                     :PRINT TITLE IF NOT IN ACT OR APT AUTO MODE
204 003066 005227 177777                3$:    INC    #-1                :FIRST TIME?
205 003072 001014                        BNE   5$                  :SKIP TITLE IF NO
206 003074 004537 011354                JSR   R5,TOBUF           :STORE 0-202 INTO BUFFER AREA
207 003100 005737 002134                TST   $AUTOB            :ARE WE IN AUTO MODE?
208 003104 001403                        BEQ   4$                  :BRANCH TO TITLE TYPEOUT IF NOT
209 003106 005737 002440                TST   XXDP              :IS THE AUTO MODE UNDER XXDP?
210 003112 001404                        BEQ   5$                  :SKIP TITLE IF NOT
211 003114 104401 016012                4$:    TYPE  ,TITLED         :PRINT OUT THE TITLE
212 003120 104401 016054                TYPE  ,TLCABL           :PRINT DRV11J CABLE REQ'D
213
214                                     :GET THE VALUE IN THE SOFTWARE SWITCH REGISTER
215 003124 005737 002134                5$:    TST   $AUTOB         :ARE WE IN AUTOMATIC MODE?
216 003130 001001                        BNE   START1            :BRANCH IF YES
217 003132 104406                        GTSWR
218 003134 005037 002202                START1: CLR  $UNIT         :CLEAR UNIT NUMBER
219 003140 013737 002246 002434        MOV   $DEVN,DMAP        :UP TO 4 DRV11J'S
220 003146 042737 177760 002434        BIC   #177760,DMAP      :POSITION OF DRV11J'S
221 003154 013701 002244                MOV   $BASE,R1          :GET BASE ADDRESS
222 003160 010137 002412                MOV   R1,DRCSA         :MAKE BUS REG.POINTER = BASE
223 003164 032737 000001 002434        BIT   #1,DMAP           :IS FIRST DRV11-J SELECTED
224 003172 001002                        BNE   NEXPAS            :YES
225 003174 000137 010636                JMP   NXDEV1            :NO,ADVANCE BASE DRV11J ADDRESS
226 003200 012700 002412                NEXPAS: MOV  #DRCSA,R0   :SET UP REGISTER ADDRESS POINTERS
227 003204 010120                NEXPA1: MOV  R1,(R0)+    :LOAD EM
228 003206 062701 000002                ADD   #2,R1
229 003212 022700 002432                CMP   #DRDBD+2,R0      :ALL DONE?
230 003216 001372                        BNE   NEXPA1            :BR IF NOT
231 003220 004537 011524                JSR   R5,TRPCAT        :EXTEND TRAP CATCHER 204-1774
232 003224 012706 002100                MOV   #2100,SP         :ALWAYS RESET STACK
233 003230 013737 002202 002200        MOV   $UNIT,$DEVCT     :LOAD APT COUNTER WITH UNIT #
234 003236 123727 002210 000001        CMPB  $ENV,#1          :ARE WE IN APT MODE?
235 003244 001401                        BEQ   NEXPA2            :BR IF YES,SKIP RESET
236 003246 000005                        RESET
237 003250 106427 000340                NEXPA2: MTPS  #PR7
238

```

```

240
241
(3)
(3)
(2) 003254 000004
242 003256 013700 002412
243 003262 013701 002416
244 003266 013702 002412
245 003272 012737 003344 002110
246 003300 012737 000002 002462
247 003306 004537 011014
248 003312 012703 012636
249 003316 012737 000050 002442
250 003324 004537 011074
251 003330 012737 011644 000300
252 003336 012737 000340 000302
253 003344 005037 002436
254 003350 012706 002100
255 003354 112710 000340
256 003360 112711 000060
257 003364 010037 002122
258 003370 112710 000060
259 003374 113710 002442
260 003400 112710 000202
261 003404 106427 000000
262 003410 112712 000241
263
264
265 003414 112710 000241
266 003420 005737 002436
267 003424 001401
268 003426 104011
269 003430 005037 002436
270 003434 112710 000120
271 003440 106427 000340
272 003444 005737 002436
273 003450 001401
274 003452 104011
275 003454 005037 002436
276 003460 106427 000000
277 003464 152762 000002 000001
278 003472 012737 000001 002124
279 003500 013737 002436 002126
280 003506 023737 002124 002126
281 003514 001401
282 003516 104012
283 003520 106427 000340
284 003524 012737 101710 002124
285 003532 010237 002122
286 003536 011237 002126
287 003542 042737 000007 002126
288 003550 023737 002124 002126
289 003556 001401
290 003560 104002
291 003562 010137 002122
292 003566 005037 002126

*****
*TEST 1 TEST CHIP INTERRUPT ,GROUP 1,GROUP 2
*****
TST1: SCOPE
MOV DRCSA,R0 ;START WITH GROUP 1
MOV DRCSB,R1
MOV DRCSA,R2 ;REGISTER IS COMMON FOR BOTH GROUPS
MOV #111$,SLPERR ;SET UP LOOP RETURN
MOV #2,GRPCNT ;COUNTER FOR BOTH GROUPS
120$: JSR R5,CLRCSR ;CLEAR ALL CSRS
MOV #BGCHP3,R3 ;EXPECTED GDDAT FOR ISR,IMR
MOV #CSIMR,IMRLOC ;STORE CLEAR SINGLE IMR CODE
JSR R5,CLRIRR ;CLEAR IRR REGS
MOV #INTSR1,300 ;STORE VECTOR ROUTINE
MOV #340,302 ;STORE PSW
111$: CLR INTFLG ;CLEAR INT. FLAG
MOV #2100,SP ;INIT STACK IN CASE OF ILLEGAL INT. LOOP
MOVB #PVMA,(R0) ;PRESELECT VECTOR MEM ADDR
MOVB #60,(R1) ;WRITE VECTOR 300,BY0,LV0
MOV R0,$BDADR ;SAVE BUS ADDRESS
MOVB #SIMR,(R0) ;SET ALL IMR BITS
MOVB IMRLOC,(R0) ;CLEAR MASK ON SINGLE BIT
MOVB #202,(R0) ;LMD04 - COMMON VECTOR
MTPS #PRO
MOVB #241,(R2) ;USED IN TEST OF GROUP 2
;ARM GROUP 1 TO PASS
;ENABLE TO GROUP 2.
;LMD57 - ARM THE CHIP
;CHECK FOR INTERRUPTS
MOV #241,(R0)
TST INTFLG
BEQ 1$
ERROR 11 ;ERROR,ILLEGAL INTERRUPT
1$: CLR INTFLG ;RESET INT. COUNTER
MOVB #SIRR,(R0) ;SET ALL IRR BITS
MTPS #PR7
TST INTFLG ;CHECK FOR NO INTERRUPTS
BEQ 2$
ERROR 11 ;ILLEGAL INTERRUPTS
2$: CLR INTFLG ;CLEAR INT. COUNTER
MTPS #PRO
BISB #BIT1,1(R2) ;SET I/E,EXPECT INTERRUPT
MOV #1,$GDDAT ;EXPECT ONE INTERRUPT
MOV INTFLG,$BDDAT ;SHOULD HAVE ONE INTERRUPT
CMP $GDDAT,$BDDAT
BEQ 3$
ERROR 12 ;INTERRUPT TEST ERROR
3$: MTPS #PR7
MOV #101710,$GDDAT ;RDY,DIR,I/E, CHIP - 31X
MOV R2,$BDADR ;CSRA ADDRESS
MOV (R2),$BDDAT
BIC #7,$BDDAT ;CLEAR UNDEFINED BITS
CMP $GDDAT,$BDDAT
BEQ 4$
ERROR 2 ;CSRA REG ERROR
4$: MOV R1,$BDADR ;READY TO READ ISR REG.
CLR $BDDAT ;SET UP FOR BYTE READS

```



293	003572	005037	002124		CLR	\$GDDAT	:SET UP FOR BYTE EXPECTED	
294	003576	111337	002124		MOVB	(R3), \$GDDAT	:EXPECTED DATA	
295	003602	112710	000240		MOVB	#MISR, (R0)	:LOAD MODE TO READ ISR	
296	003606	111137	002126		MOVB	(R1), \$BDDAT		
297	003612	023737	002124	002126	CMP	\$GDDAT, \$BDDAT		
298	003620	001401			BEQ	5\$		
299	003622	104006			ERROR	6	:ISR ERROR	
300	003624	116337	000001	002124	5\$:	MOVB	1(R3), \$GDDAT	:STORE EXPECTED
301	003632	112710	000244		MOVB	#MIMR, (R0)	:LOAD MODE FOR IMR	
302	003636	111137	002126		MOVB	(R1), \$BDDAT		
303	003642	023737	002124	002126	CMP	\$GDDAT, \$BDDAT		
304	003650	001401			BEQ	6\$		
305	003652	104005			ERROR	5	:IMR ERROR	
306	003654	116337	000001	002124	6\$:	MOVB	1(R3), \$GDDAT	:EXPECTED IRR
307	003662	112710	000250		MOVB	#MIRR, (R0)	:LOAD MODE BITS FOR IRR	
308	003666	111137	002126		MOVB	(R1), \$BDDAT		
309	003672	023737	002124	002126	CMP	\$GDDAT, \$BDDAT		
310	003700	001401			BEQ	7\$		
311	003702	104003			ERROR	3	:IRR ERROR	
312	003704	005037	002124	7\$:	CLR	\$GDDAT		
313	003710	112710	000254		MOVB	#MACR, (R0)		
314	003714	111137	002126		MOVB	(R1), \$BDDAT		
315	003720	023737	002124	002126	CMP	\$GDDAT, \$BDDAT		
316	003726	001401			BEQ	10\$		
317	003730	104004			ERROR	4	:ACR ERROR	
318	003732	005037	002124	10\$:	CLR	\$GDDAT		
319	003736	112710	000140		MOVB	#CHPISR, (R0)	:CLEAR HIGHEST PRIOR ISR	
320	003742	112710	000240		MOVB	#MISR, (R0)	:LOAD MODE TO READ ISR	
321	003746	111137	002126		MOVB	(R1), \$BDDAT		
322	003752	023737	002124	002126	CMP	\$GDDAT, \$BDDAT		
323	003760	001401			BEQ	11\$		
324	003762	104006			ERROR	6	:ISR REG ERROR	
325	003764	012737	101710	002124	11\$:	MOV	#101710, \$GDDAT	:RDY, DIR, I/E, CHIP - 31X
326	003772	010237	002122		MOV	R2, \$BDADR	:CSRA ADDRESS	
327	003776	011237	002126		MOV	(R2), \$BDDAT		
328	004002	042737	000007	002126	BIC	#7, \$BDDAT	:CLEAR UNDEFINED BITS	
329	004010	023737	002124	002126	CMP	\$GDDAT, \$BDDAT		
330	004016	001401			BEQ	12\$		
331	004020	104002			ERROR	2	:CSRA REG ERROR	
332	004022	012737	100710	002124	12\$:	MOV	#100710, \$GDDAT	:RDY, DIR, CHIP - 31X
333	004030	042712	001000		BIC	#BIT9, (R2)	:CLEAR I/E	
334	004034	011237	002126		MOV	(R2), \$BDDAT	:READ CSRA	
335	004040	042737	000007	002126	BIC	#7, \$BDDAT	:CLEAR UNDEFINED BITS	
336	004046	023737	002124	002126	CMP	\$GDDAT, \$BDDAT		
337	004054	001401			BEQ	13\$		
338	004056	104002			ERROR	2	:CSRA REG ERROR	
339	004060	005723		13\$:	TST	(R3)+	:GET NEXT ISR, IMR EXPECTED RESULTS	
340	004062	020327	012656		CMP	R3, #EDCHP3	:CHECK FOR END	
341	004066	001404			BEQ	14\$		
342	004070	005237	002442		INC	IMRLOC	:INCREMENT MASK BIT TO CLEAR	
343	004074	000137	003344		JMP	111\$	:TEST NEXT ISR BIT	
344	004100	005337	002462	14\$:	DEC	GRPCNT	:TESTED BOTH GROUPS?	
345	004104	001406			BEQ	TST2	:BR IF DONE	
346	004106	013700	002422		MOV	DRCSC, R0	:SETUP FOR GROUP 2	
347	004112	013701	002426		MOV	DRCSD, R1	:CSRC = CONTROL GROUP 2	
348							:CSRD = DATA GROUP 2	

```

349 004116 000137 003306      JMP      120$      ;DO GROUP 2
350
351
352      ;*****
353      ;*TEST 2      TEST ACR/ISR INTERRUPT ,GROUP 1,GROUP 2
354      ;*****
355      (3)
356      (3)
357      (2) 004122 000004      TST2:  SCOPE
358      004124 013700 002412      MOV      DRCSA,R0
359      004130 013701 002416      MOV      DRCSB,R1
360      004134 013702 002412      MOV      DRCSA,R2
361      004140 012737 004212 002110  MOV      #111$, $LPERR      ;SET UP LOOP RETURN
362      004146 012737 000002 002462  MOV      #2,GRPCNT      ;DO TWO GROUPS
363      004154 004537 011014      JSR      R5,CLRCR      ;CLEAR ALL CSRS
364      004160 012703 012636      MOV      #BGCHP3,R3      ;EXPECTED GDDAT FOR ISR
365      004164 012737 000050 002442  MOV      #CSIMR,IMRLOC      ;STORE CLEAR SINGLE IMR CODE
366      004172 004537 011074      JSR      R5,CLRIRR      ;CLEAR IRR REGS
367      004176 012737 011644 000300  MOV      #INTSR1,300      ;STORE VECTOR ROUTINE
368      004204 012737 000340 000302  MOV      #340,302      ;STORE PSW
369      004212 005037 002436      111$:  CLR      INTFLG      ;CLEAR INT. FLAG
370      004216 012706 002100      MOV      #2100,SP      ;INIT STACK IN CASE OF ILLEGAL INT. LOOP
371      004222 042712 001000      BIC      #BIT9,(R2)      ;CLEAR INTERRUPT ENABLE
372      004226 112712 000242      MOVB    #242,(R2)      ;CLEAR MASTER MASK GP1
373      004232 112710 000242      MOVB    #242,(R0)      ;CLEAR MASTER MASK GP1,GP2
374      004236 112710 000300      MOVB    #PACR,(R0)      ;PRESELECT ACR FOR WRITING
375      004242 111311      MOVB    (R3),(R1)      ;NEXT BIT INTO ACR
376      004244 112710 000340      MOVB    #PVMA,(R0)      ;PRESELECT VECTOR MEM ADDR
377      004250 112711 000060      MOVB    #60,(R1)      ;WRITE VECTOR 300,BY0,LV0
378      004254 010037 002122      MOV      R0,$BDADR      ;STORE BUS ADDRESS
379      004260 112710 000060      MOVB    #SIMR,(R0)      ;SET ALL IMR BITS
380      004264 113710 002442      MOVB    IMRLOC,(R0)      ;CLEAR MASK ON SINGLE BIT
381      004270 112710 000202      MOVB    #202,(R0)      ;LMD04 - COMMON VECTOR
382      004274 106427 000000      MTPS    #PRO
383      004300 152762 000002 000001  BISB    #BIT1,1(R2)      ;SET I/E,NO INTERRUPT
384      004306 005737 002436      TST     INTFLG      ;CHECK FOR INTERRUPTS
385      004312 001401      BEQ     1$
386      004314 104011      ERROR   11      ;ERROR,ILLEGAL INTERRUPT
387      004316 005037 002436      1$:    CLR     INTFLG      ;RESET INT. COUNTER
388      004322 112710 000120      MOVB    #SIRR,(R0)      ;SET ALL IRR BITS
389      004326 106427 000340      MTPS    #PR7
390      004332 005737 002436      TST     INTFLG      ;CHECK FOR NO INTERRUPTS
391      004336 001401      BEQ     2$
392      004340 104011      ERROR   11      ;ILLEGAL INTERRUPTS
393      004342 005037 002436      2$:    CLR     INTFLG      ;CLEAR INT. COUNTER
394      004346 106427 000000      MTPS    #PRO
395      004352 112712 000241      MOVB    #241,(R2)      ;USED IN TEST OF GROUP 2
396      004356 112710 000241      MOVB    #241,(R0)      ;ARM GROUP 1 CHIP TO PASS
397      004362 012737 000001 002124  MOV      #1,$GDDAT      ;ENABLE TO GROUP 2 CHIP.
398      004370 013737 002436 002126  MOV      INTFLG,$BDDAT      ;ARM THE CHIP
399      004376 023737 002124 002126  CMP      $GDDAT,$BDDAT      ;EXPECT ONE INTERRUPT
400      004404 001401      BEQ     3$      ;SHOULD HAVE ONE INTERRUPT
401      004406 104012      ERROR   12      ;INTERRUPT TEST ERROR
402      004410 106427 000340      3$:    MTPS    #PR7
403      004414 010137 002122      4$:    MOV      R1,$BDADR      ;READY TO READ ISR REG.
404      004420 005037 002126      CLR     $BDDAT      ;SET UP FOR BYTE READS

```

```

402 004424 005037 002124 CLR $GDDAT ;SET UP FOR BYTE EXPECTED
403 004430 112710 000240 MOV #MISR, (R0) ;LOAD MODE TO READ ISR
404 004434 111137 002126 MOV (R1), $BDDAT
405 004440 023737 002124 002126 CMP $GDDAT, $BDDAT ;ACR SHOULD CLEAR ISR
406 004446 001401 BEQ 5$
407 004450 104006 ERROR 6 ;ISR ERROR
408 004452 116337 000001 002124 5$: MOV 1(R3), $GDDAT ;STORE EXPECTED
409 004460 112710 000244 MOV #MIMR, (R0) ;LOAD MODE FOR IMR
410 004464 111137 002126 MOV (R1), $BDDAT
411 004470 023737 002124 002126 CMP $GDDAT, $BDDAT
412 004476 001401 BEQ 6$
413 004500 104005 ERROR 5 ;IMR ERROR
414 004502 116337 000001 002124 6$: MOV 1(R3), $GDDAT ;EXPECTED IRR
415 004510 112710 000250 MOV #MIRR, (R0) ;LOAD MODE BITS FOR IRR
416 004514 111137 002126 MOV (R1), $BDDAT
417 004520 023737 002124 002126 CMP $GDDAT, $BDDAT
418 004526 001401 BEQ 7$
419 004530 104003 ERROR 3 ;IRR ERROR
420 004532 111337 002124 7$: MOV (R3), $GDDAT ;EXPECTED ACR
421 004536 112710 000254 MOV #MACR, (R0)
422 004542 111137 002126 MOV (R1), $BDDAT
423 004546 023737 002124 002126 CMP $GDDAT, $BDDAT
424 004554 001401 BEQ 10$
425 004556 104004 ERROR 4 ;ACR ERROR
426 004560 005723 10$: TST (R3)+ ;GET NEXT ISR, IMR, IRR EXPECTED RESULTS
427 004562 020327 012656 CMP R3, #EDCHP3 ;CHECK FOR END
428 004566 001404 BEQ 11$ ;DONE WITH GROUP TEST
429 004570 005237 002442 INC IMRLOC ;INCREMENT MASK BIT TO CLEAR
430 004574 000137 004212 JMP 111$ ;TEST NEXT ISR BIT
431 004600 005337 002462 11$: DEC GRPCNT ;TESTED BOTH GROUPS?
432 004604 001406 BEQ TST3 ;:BR IF DONE
433 004606 013700 002422 MOV DRCSA, R0 ;SETUP FOR GROUP 2
434 004612 013701 002426 MOV DRCSA, R1 ;CSRC = CONTROL GROUP 2
435 ;CSR = DATA GROUP 2
436 004616 000137 004154 JMP 120$ ;DO GROUP 2
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454

```

\*\*\*\*\*  
 : \*TEST 3 TEST VECTOR ADDR MEM, LEV 0-7, BY0-3, (204-1774), GRPS 1,2  
 \*\*\*\*\*

```

(2) 004622 000004 TST3: SCOPE
441 004624 013700 002412 MOV DRCSA, R0
442 004630 013701 002416 MOV DRCSB, R1
443 004634 013702 002412 MOV DRCSA, R2
444 004640 012737 000002 002462 MOV #2, GRPCNT ;DO TWO GROUPS
445 004646 012737 000001 002472 120$: MOV #1, BYNUM ;INIT FOR BYTE COUNT 0
446 004654 004537 011014 JSR R5, CLRCR ;CLEAR ALL CSRS
447 004660 012737 000340 002456 MOV #PVMA, VECVAL ;START VECTOR LEVEL 0, BYTE COUNT 0
448 004666 012704 012636 121$: MOV #BGCHP3, R4 ;EXPECTED ISR, IMR PATTERN
449 004672 012737 000010 002466 MOV #8, LVL CNT ;COUNTER FOR 0-7 LEVELS
450 004700 012737 000050 002442 MOV #CSIMR, IMRLOC ;STORE SINGLE IMR CODE
451 004706 012737 000130 002446 MOV #SSIRR, IRRLOC ;STORE SINGLE IRR CODE
452 004714 012737 000170 002444 MOV #CSISR, ISRLOC ;STORE CLEAR SINGLE ISR CODE
453 004722 004537 011524 JSR R5, TRPCAT ;RESTORE TRAP CATCHER
454 004726 012737 004742 002110 MOV #112$, $LPERR ;STORE SCOPE LOOP

```

```

455 004734 012737 000204 002452 111$: MOV #204,VECLOC ;START AT VECTOR 204
456 004742 106427 000340 112$: MTPS #340 ;
457 004746 012706 002100 MOV #2100,SP ;INIT STACK IN CASE OF ILLEGAL INT. LOOP
458 004752 042712 001000 BIC #BIT9,(R2) ;CLEAR INTERRUPT ENABLE
459 004756 004537 011074 JSR R5,CLRIRR ;CLEAR IRR REGS
460 004762 112712 000241 MOVB #241,(R2) ;USED IN TEST OF GROUP 2
461 ;ARM GROUP 1 CHIP TO PASS
462 ;ENABLE TO GROUP2
463 004766 013703 002452 MOV VECLOC,R3 ;GET VECTOR
464 004772 010337 002454 MOV R3,VECPAT
465 004776 006037 002454 ROR VECPAT ;ROTATE VECTOR ADDR TWO RIGHT
466 005002 006037 002454 ROR VECPAT ;TO WRITE VECTOR MEM ADDR.
467 005006 012713 011700 MOV #INTBY4,(R3) ;STORE VECTOR ROUTINE
468 005012 012763 000340 000002 MOV #340,2(R3) ;STORE PSW
469 005020 005037 002436 CLR INTFLG ;CLEAR INT. FLAG
470 005024 013737 002472 002470 MOV BYNUM,BYCNT ;BYTE COUNTS OF 0-3
471 005032 113710 002456 MOVB VECVAL,(R0) ;PRESELECT VECTOR MEM ADDR
472 005036 113711 002454 113$: MOVB VECPAT,(R1) ;WRITE VECTOR
473 005042 005337 002470 DEC BYCNT ;DONE WITH BYTE COUNT VALUE
474 005046 001373 BNE 113$ ;NO,LOAD VECTOR FOR NEXT BYTE COUNT
475 005050 010037 002122 MOV R0,$BDADR ;SAVE BUS ADDRESS
476 005054 112710 000060 MOVB #SIMR,(R0) ;SET ALL IMR BITS
477 005060 113710 002442 MOVB IMRLOC,(R0) ;CLEAR MASK ON SINGLE BIT
478 005064 106427 000000 MTPS #PRO
479 005070 112710 000241 MOVB #241,(R0) ;LMD57 - ARM THE CHIP
480 005074 005737 002436 TST INTFLG ;CHECK FOR INTERRUPTS
481 005100 001401 BEQ 1$
482 005102 104011 ERROR 11 ;ERROR,ILLEGAL INTERRUPT
483 005104 005037 002436 1$: CLR INTFLG ;RESET INT. COUNTER
484 005110 113710 002446 MOVB IRRLOC,(R0) ;SET SINGLE IRR BIT
485 005114 106427 000340 MTPS #PR7
486 005120 005737 002436 TST INTFLG ;CHECK FOR NO INTERRUPTS
487 005124 001401 BEQ 2$
488 005126 104011 ERROR 11 ;ILLEGAL INTERRUPTS
489 005130 005037 002436 2$: CLR INTFLG ;CLEAR INT. COUNTER
490 005134 106427 000000 MTPS #PRO
491 005140 152762 000002 000001 BISB #BIT1,1(R2) ;SET I/E,EXPECT INTERRUPT
492 005146 013737 002472 002124 MOV BYNUM,$GDDAT ;EXPECT 1 TO 4 INTERRUPTS
493 ;BASED ON BYTE COUNT VALUE
494 005154 013737 002436 002126 MOV INTFLG,$BDDAT ;SHOULD HAVE 1 TO 4 INTERRUPTS
495 ;BASED ON BYTE COUNT VALUE
496 005162 023737 002124 002126 CMP $GDDAT,$BDDAT
497 005170 001401 BEQ 3$
498 005172 104012 ERROR 12 ;INTERRUPT TEST ERROR
499 005174 106427 000340 3$: MTPS #PR7
500 005200 010137 002122 4$: MOV R1,$BDADR ;READY TO READ ISR REG.
501 005204 005037 002126 CLR $BDDAT ;SET UP FOR BYTE READS
502 005210 005037 002124 CLR $GDDAT ;SET UP FOR BYTE EXPECTED
503 005214 111437 002124 MOVB (R4),$GDDAT ;EXPECTED DATA
504 005220 112710 000240 MOVB #MISR,(R0) ;LOAD MODE TO READ ISR
505 005224 111137 002126 MOVB (R1),$BDDAT
506 005230 023737 002124 002126 CMP $GDDAT,$BDDAT
507 005236 001401 BEQ 5$
508 005240 104006 ERROR 6 ;ISR ERROR
509 005242 116437 000001 002124 5$: MOVB 1(R4),$GDDAT ;STORE EXPECTED
510 005250 112710 000244 MOVB #MIMR,(R0) ;LOAD MODE FOR IMR

```

```
511 005254 111137 002126      MOVB (R1),SBDDAT
512 005260 023737 002124 002126  CMP  $GDDAT,SBDDAT
513 005266 001401      BEQ  6$
514 005270 104005      ERROR 5          :IMR ERROR
515 005272 005037 002124      CLR  $GDDAT
516 005276 112710 000250      MOVB #MIRR,(R0)   :LOAD MODE BITS FOR IRR
517 005302 111137 002126      MOVB (R1),SBDDAT
518 005306 023737 002124 002126  CMP  $GDDAT,SBDDAT
519 005314 001401      BEQ  7$
520 005316 104003      ERROR 3          :IRR ERROR
521 005320 005037 002124      CLR  $GDDAT
522 005324 112710 000254      MOVB #MACR,(R0)
523 005330 111137 002126      MOVB (R1),SBDDAT
524 005334 023737 002124 002126  CMP  $GDDAT,SBDDAT
525 005342 001401      BEQ  10$
526 005344 104004      ERROR 4          :ACR ERROR
527 005346 005037 002124      CLR  $GDDAT
528 005352 113710 002444      MOVB ISRLOC,(R0)  :CLEAR SINGLE ISR BIT
529 005356 112710 000240      MOVB #MISR,(R0)  :LOAD MODE TO READ ISR
530 005362 111137 002126      MOVB (R1),SBDDAT
531 005366 023737 002124 002126  CMP  $GDDAT,SBDDAT
532 005374 001401      BEQ  11$
533 005376 104006      ERROR 6          :ISR REG ERROR
534 005400 004537 011432      JSR  R5,RESTRP   :RESTORE VECTOR JUST TESTED
535 005404 005737 017374      TST  KXTFLAG     : KXT11 ??
536 005410 001404      BEQ  100$        : BR IF NOT
537 005412 022737 000400 002452  CMP  #400,VECLOC : YES, FINISHED 204-374 ??
538 005420 000403      403             : SKIP NEXT TO FIND OUT
539 005422      100$:
540 005422 022737 002000 002452  CMP  #2000,VECLOC :FINISHED?
541 005430 001402      BEQ  12$        :FINISHED VECTORS 204-1774?
542 005432 000137 004742      JMP  112$       :TEST NEXT VECTOR ADDR
543 005436 005724      12$: TST (R4)+      :INDEX EXPECTED ISR AND IMR PATTERN
544 005440 005237 002442      INC  IMRLOC     :STORE CLEAR FOR NEXT MASK BIT
545 005444 005237 002446      INC  IRRLOC     :STORE SET FOR NEXT IRR BIT
546 005450 005237 002444      INC  ISRLOC     :STORE CLEAR FOR NEXT ISR BIT
547 005454 005237 002456      INC  VECVAL     :SETUP TO TEST NEXT VECTOR LEVEL
548 005460 005337 002466      DEC  LVLCNT     :FINISHED LEVELS 0-7
549 005464 001402      BEQ  13$        :YES,CHECK BYTE COUNT END
550 005466 000137 004734      JMP  111$       :NO,DO NEXT LEVEL
551 005472 005237 002472      13$: INC  BYNUM     :INDEX BYTE COUNT
552 005476 104407      CKSWR          :LOOK FOR SWITCH CHANGE REQUEST
553 005500 022737 000400 002456  CMP  #400,VECVAL :FINISHED ALL BYTE COUNTS 0-3
554 005506 001402      BEQ  14$        :FINISHED TWO GROUPS?
555 005510 000137 004666      JMP  121$       :DO NEXT BYTE COUNT FOR
556      14$:
557 005514 005337 002462      DEC  GRPCNT     :LEVELS 0-7
558 005520 001406      BEQ  TST4       :FINISHED TWO GROUPS?
559 005522 013700 002422      MOV  DRCSC,R0   :BR IF DONE
560 005526 013701 002426      MOV  DRCSD,R1   :CSRC = CONTROL GROUP 2
561 005532 000137 004646      JMP  120$       :CSRD = DATA GROUP 2
562      :DO GROUP 2 FOR BYTE COUNTS 0-3
563      :AND LEVELS 0-7 FOR EACH BYTE COUNT.
564
565
(3)
*****
:*TEST 4 TEST VECTOR ADDR MEM,LVLS 0-7,BY0-3,(0-200)GRPS 1,2
```

```
(3)
(2) 005536 000004
566 005540 013700 002412
567 005544 013701 002416
568 005550 013702 002412
569 005554 012737 000002 002462
570 005562 012737 000001 002472 120$:
571 005570 004537 011014
572 005574 012737 000340 002456
573 005602 012704 012636 121$:
574 005606 012737 000010 002466
575 005614 012737 000050 002442
576 005622 012737 000130 002446
577 005630 012737 005642 002110
578 005636 005037 002452 111$:
579 005642 106427 000340 112$:
580 005646 012706 002100
581 005652 042712 001000
582 005656 004537 011074
583 005662 112712 000241
584
585
586 005666 013703 002452
587 005672 004537 012024
588 005676 010337 002454
589 005702 006037 002454
590 005706 006037 002454
591 005712 012713 012124
592 005716 012763 000340 000002
593 005724 005037 002436
594 005730 013737 002472 002470
595
596 005736 113710 002456
597 005742 113711 002454 113$:
598 005746 005337 002470
599 005752 001373
600 005754 010037 002122
601 005760 112710 000060
602 005764 113710 002442
603 005770 106427 000000
604 005774 112710 000241
605 006000 005737 002436
606 006004 001405
607 006006 004537 011374
608 006012 104011
609 006014 004537 012024
610 006020 005037 002436 1$:
611 006024 113710 002446
612 006030 106427 000340
613 006034 005737 002436
614 006040 001405
615 006042 004537 011374
616 006046 104011
617 006050 004537 012024
618 006054 005037 002436 2$:
619 006060 106427 000000
```

\*\*\*\*\*  
TST4: SCOPE  
MOV DRCSA,R0  
MOV DRCSB,R1  
MOV DRCSA,R2  
MOV #2,GRPCNT :DO TWO GROUPS  
MOV #1,BYNUM :INIT FOR FIRST BYTE COUNT  
JSR R5,CLRCR :CLEAR ALL CSRS  
MOV #PVMA,VECVL :START VECTOR LEVEL 0,BYTE COUNT 0  
MOV #BGCHP3,R4 :ISR,IMR PATTERN  
MOV #8.,LVLcnt :START LEVEL COUNT 0-7  
MOV #CSIMR,IMRLOC :STORE CLEAR SINGLE IMR BIT CODE  
MOV #SSIRR,IRRLOC :STORE SET SINGLE IRR BIT CODE  
MOV #112\$, \$LPERR :LOOP RETURN  
CLR VECLOC :START AT VECTOR 0  
MTPS #340  
MOV #2100,SP :INIT STACK IN CASE OF ILLEGAL INT. LOOP  
BIC #BIT9,(R2) :CLEAR INTERRUPT ENABLE  
JSR R5,CLRIRR :CLEAR IRR REGS  
MOVB #241,(R2) :USED IN TEST OF GROUP 2  
:ARM GROUP 1 CHIP TO PASS  
:ENABLE TO GROUP 2.  
MOV VECLOC,R3 :GET VECTOR  
JSR R5,CAT200 :STORE TRAP CATCHER FOR 0-200  
MOV R3,VECPAT  
ROR VECPAT :ROTATE VECTOR ADDR TWO RIGHT  
ROR VECPAT :TO WRITE VECTOR MEM ADDR.  
MOV #INTSR3,(R3) :STORE VECTOR ROUTINE  
MOV #340,2(R3) :STORE PSW  
CLR INTFLG :CLEAR INT. FLAG  
MOV BYNUM,BYCNT :BYTE COUNT OF 0 TO 3  
:FOR 1 TO 4 VECTORS.  
MOVB VECVAL,(R0) :PRESELECT VECTOR MEM ADDR  
MOVB VECPAT,(R1) :WRITE VECTOR  
DEC BYCNT :DO 1 TO 4 VECTORS  
BNE 113\$ :WRITE VECTORS BASED ON BYTE COUNT  
MOV R0,\$BDADR :SAVE BUS ADDRESS  
MOVB #SIMR,(R0) :SET ALL IMR BITS  
MOVB IMRLOC,(R0) :CLEAR MASK ON SINGLE BIT  
MTPS #PRO  
MOVB #241,(R0) :LMD57 - ARM THE CHIP  
TST INTFLG :CHECK FOR INTERRUPTS  
BEQ 1\$  
JSR R5,FRMBUF :RESTORE 0-202  
ERROR 11 :ERROR,ILLEGAL INTERRUPT  
JSR R5,CAT200 :RESTORE TRAP CATCHER  
CLR INTFLG :RESET INT. COUNTER  
MOVB IRRLOC,(R0) :SET SINGLE IRR BIT  
MTPS #PR7  
TST INTFLG :CHECK FOR NO INTERRUPTS  
BEQ 2\$  
JSR R5,FRMBUF :RESTORE 0-202  
ERROR 11 :ILLEGAL INTERRUPTS  
JSR R5,CAT200 :RESTORE TRAP CATCHER  
CLR INTFLG :CLEAR INT. COUNTER  
MTPS #PRO

620	006064	152762	000002	000001		BISB	#BIT1,1(R2)	:SET I/E,EXPECT INTERRUPT
621	006072	013737	002472	002124		MOV	BYNUM,\$GDDAT	:EXPECT NUMBER OF INTERRUPTS
622								:EQUAL TO BYTE COUNT.
623	006100	013737	002436	002126		MOV	INTFLG,\$BDDAT	:SHOULD HAVE NUMBER OF INTERRUPTS
624								:EQUAL TO BYTE COUNT.
625	006106	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
626	006114	001405				BEQ	3\$	
627	006116	004537	011374			JSR	R5,FRMBUF	:RESTORE 0-202
628	006122	104012				ERROR	12	:INTERRUPT TEST ERROR
629	006124	004537	012024			JSR	R5,CAT200	:RESTORE TRAP CATCHER SUB
630	006130	106427	000340		3\$:	MTPS	#PR7	
631	006134	010137	002122		4\$:	MOV	R1,\$BDADR	:READY TO READ ISR REG.
632	006140	005037	002126			CLR	\$BDDAT	:SET UP FOR BYTE READS
633	006144	005037	002124			CLR	\$GDDAT	:SET UP FOR BYTE EXPECTED
634	006150	111437	002124			MOVB	(R4),\$GDDAT	:EXPECTED DATA
635	006154	112710	000240			MOVB	#MISR,(R0)	:LOAD MODE TO READ ISR
636	006160	111137	002126			MOVB	(R1),\$BDDAT	
637	006164	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
638	006172	001405				BEQ	5\$	
639	006174	004537	011374			JSR	R5,FRMBUF	:RESTORE 0-200
640	006200	104006				ERROR	6	:ISR ERROR
641	006202	004537	012024			JSR	R5,CAT200	:RESTORE TRAP CATCHER
642	006206	116437	000001	002124	5\$:	MOVB	1(R4),\$GDDAT	:STORE EXPECTED
643	006214	112710	000244			MOVB	#MIMR,(R0)	:LOAD MODE FOR IMR
644	006220	111137	002126			MOVB	(R1),\$BDDAT	
645	006224	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
646	006232	001405				BEQ	6\$	
647	006234	004537	011374			JSR	R5,FRMBUF	:RESTORE 0-202
648	006240	104005				ERROR	5	:IMR ERROR
649	006242	004537	012024			JSR	R5,CAT200	:RESTORE CATCHER
650	006246	005037	002124		6\$:	CLR	\$GDDAT	
651	006252	112710	000250			MOVB	#MIRR,(R0)	:LOAD MODE BITS FOR IRR
652	006256	111137	002126			MOVB	(R1),\$BDDAT	
653	006262	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
654	006270	001405				BEQ	7\$	
655	006272	004537	011374			JSR	R5,FRMBUF	:RESTORE 0-202
656	006276	104003				ERROR	3	:IRR ERROR
657	006300	004537	012024			JSR	R5,CAT200	:RESTORE TRAP CATCHER
658	006304	005037	002124		7\$:	CLR	\$GDDAT	
659	006310	112710	000254			MOVB	#MACR,(R0)	
660	006314	111137	002126			MOVB	(R1),\$BDDAT	
661	006320	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
662	006326	001405				BEQ	10\$	
663	006330	004537	011374			JSR	R5,FRMBUF	:RESTORE 0-202
664	006334	104004				ERROR	4	:ACR ERROR
665	006336	004537	012024			JSR	R5,CAT200	:RESTORE TRAP CATCHER
666	006342	005037	002124		10\$:	CLR	\$GDDAT	
667	006346	112710	000160			MOVB	#CISR,(R0)	:CLEAR ISR
668	006352	112710	000240			MOVB	#MISR,(R0)	:LOAD MODE TO READ ISR
669	006356	111137	002126			MOVB	(R1),\$BDDAT	
670	006362	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
671	006370	001405				BEQ	11\$	
672	006372	004537	011374			JSR	R5,FRMBUF	:RESTORE 0-202
673	006376	104006				ERROR	6	:ISR REG ERROR
674	006400	004537	012024			JSR	R5,CAT200	:RESTORE TRAP CATCHER
675	006404	004537	011762		11\$:	JSR	R5,RES200	:RESTORE VECTOR JUST TESTED

```
676 006410 004537 011374 JSR R5,FRMBUF ;RESTORE 0-200 TO SYSMAC CONTROL
677 006414 022737 000204 002452 CMP #204,VECLOC ;FINISHED?
678 006422 001402 BEQ 12$
679 006424 000137 005642 JMP 112$ ;TEST NEXT VECTOR ADDR
680 006430 005724 12$: TST (R4)+ ;INDEX EXPECTED ISR AND IMR PATTERN
681 006432 005237 002442 INC IMRLOC ;STORE CLEAR NEXT IMR BIT
682 006436 005237 002446 INC IRRLOC ;STORE SET NEXT IRR BIT
683 006442 005237 002456 INC VECVAL ;SETUP TO TEST NEXT VECTOR LEVEL
684 006446 005337 002466 DEC LVL CNT ;FINISHED LEVELS 0-7?
685 006452 001402 BEQ 13$ ;YES,CHECK BYTE COUNT END
686 006454 000137 005636 JMP 111$ ;NO,DO NEXT LEVEL,SAME BYTE COUNT
687 006460 005237 002472 13$: INC BYNUM ;INDEX BYTE COUNT
688 006464 104407 CKSWR ;LOOK FOR SWITCH CHANGE
689 006466 022737 000400 002456 CMP #400,VECVAL ;FINISHED ALL FOUR BYTE COUNTS
690 006474 001402 BEQ 14$ ;YES,FINISHED GROUP
691 006476 000137 005602 JMP 121$ ;NO,DO NEXT BYTE COUNT WITH
692 ;LEVELS 0-7.
693 006502 005337 002462 14$: DEC GRPCNT ;FINISHED BOTH GROUP 1 AND GROUP 2?
694 006506 001406 BEQ 15$ ;YES,BOTH GROUPS TESTED
695 006510 013700 002422 MOV DRCSC,R0 ;NO,TEST GROUP 2
696 ;CSRC = CONTROL GROUP 2
697 006514 013701 002426 MOV DRCSD,R1 ;CSRD = DATA GROUP 2
698 006520 000137 005562 JMP 120$ ;RETURN AND TEST GROUP2
699 ;FOR BYTE COUNTS 0-3 AND
700 ;LEVELS 0-7 FOR EACH BYTE COUNT.
701 006524 004537 011374 15$: JSR R5,FRMBUF ;RESTORE 0-202,EXIT TEST
702
703
704
(3) ;*****
(3) ;*TEST 5 TEST VECTOR ADDR UNIQUENESS IN FIXED MODE FOR GRPS 1,2
(2) 006530 000004 ;*****
705 TST5: SCOPE ;BYTE COUNT = 4
706 ;LEVELS = 0-7
707 006532 004537 011014 JSR R5,CLRCSR ;CLEAR ALL CSRS
708 006536 004537 011334 JSR R5,CLRBF ;CLEAR VECTOR BUFFER AREA
709 006542 004537 011524 JSR R5,TRPCAT ;RESTORE TRAP CATCHER
710 006546 004537 011300 JSR R5,STRVEC ;STORE VECTOR AREA 300-674
711 ;WITH INT. SERV ROUTINES
712 006552 012737 000377 002456 MOV #377,VECVAL ;VECTOR MEM FINAL VALUE
713 006560 004537 011074 JSR R5,CLRIRR ;CLEAR IRR REGS
714 006564 012704 017160 MOV #VBUF,R4 ;VECTOR ADDRESS BUFFER
715 006570 004537 011132 JSR R5,VECFIL ;FILL VECTOR MEMORY FOR GROUPS 1,2
716 ;WITH VECTORS 300-674
717 006574 012705 012210 MOV #SETVEC,R5 ; R5 POINTS TO COMMON HANDLER ;;GPA
718 006600 013700 002412 MOV DRCSA,R0 ;GROUP 1 CSR
719 006604 013701 002422 MOV DRCSC,R1 ;GROUP 2 CSR
720 006610 013702 002416 MOV DRCSE,R2
721 006614 013703 002426 MOV DRCSD,R3
722 006620 010037 002122 MOV R0,$BDADR ;STORE BUS ADDRESS
723 006624 112710 000300 MOVB #PACR,(R0) ;PRESELECT GP1 ACR FOR WRITING
724 006630 112712 000377 MOVB #377,(R2) ;SET ALL ACR BITS GP1
725 006634 112710 000040 MOVB #CIMR,(R0) ;CLEAR ALL IMR BITS GP1
726 006640 005037 002436 CLR INTFLG ;CLEAR INT. FLAG
727 006644 106427 000000 MTPS #PRO
728 006650 112710 000120 MOVB #SIRR,(R0) ;SET ALL IRR BITS
```





785	007210	104003			ERROR	3		:IRR ERROR	
786	007212	012737	000377	002124	MOV	#377,\$GDDAT		:EXPECTED ACR	
787	007220	112710	000254		MOVB	#MACR,(R0)		:READ ACR	
788	007224	111237	002126		MOVB	(R2),\$BDDAT			
789	007230	023737	002124	002126	CMP	\$GDDAT,\$BDDAT			
790	007236	001401			BEQ	11\$			
791	007240	104004			ERROR	4		:ACR ERROR	
792	007242	005037	002124		CLR	\$GDDAT			
793	007246	005037	002126		CLR	\$BDDAT			
794	007252	010337	002122		MOV	R3,\$BDADR		:CSR ADDRESS	
795	007256	112711	000240		MOVB	#MISR,(R1)		:READ ISR ,GP2	
796	007262	111337	002126		MOVB	(R3),\$BDDAT			
797	007266	023737	002124	002126	CMP	\$GDDAT,\$BDDAT			
798	007274	001401			BEQ	12\$			
799	007276	104006			ERROR	6		:ISR ERROR,GP1	
800	007300	112711	000240		MOVB	#MIMR,(R1)		:READ IMR GP2	
801	007304	111337	002126		MOVB	(R3),\$BDDAT			
802	007310	023737	002124	002126	CMP	\$GDDAT,\$BDDAT			
803	007316	001401			BEQ	13\$			
804	007320	104005			ERROR	5		:IMR ERROR,GP2	
805									
806	007322	112711	000250		MOVB	#MIRR,(R1)		:READ IRR	
807	007326	111337	002126		MOVB	(R3),\$BDDAT			
808	007332	023737	002124	002126	CMP	\$GDDAT,\$BDDAT			
809	007340	001401			BEQ	14\$			
810	007342	104003			ERROR	3		:IRR ERROR,GP2	
811	007344	112711	000254		MOVB	#MACR,(R1)		:READ ACR	
812	007350	012737	000377	002124	MOV	#377,\$GDDAT			
813	007356	111337	002126		MOVB	(R3),\$BDDAT			
814	007362	023737	002124	002126	CMP	\$GDDAT,\$BDDAT			
815	007370	001401			BEQ	15\$			
816	007372	104004			ERROR	4		:ACR ERROR,GP2	
817	007374	012702	000100		MOV	#64,R2			
818	007400	012704	017160		MOV	#VBUF,R4		:VECTOR BUFFER	
819	007404	012737	000300	002124	MOV	#300,\$GDDAT		:START WITH FIRST VECTOR 300	
820	007412	010437	002122		MOV	R4,\$BDADR		:VECTOR BUFFER ADDRESS	
821	007416	011437	002126		MOV	(R4),\$BDDAT			
822	007422	023737	002124	002126	CMP	\$GDDAT,\$BDDAT			
823	007430	001401			BEQ	17\$		:VECTORED PROPERLY	
824	007432	104007			ERROR	7		:VECTOR ADDR MEM ERROR	
825	007434	062737	000004	002124	ADD	#4,\$GDDAT			
826	007442	005737	017374		TST	KXTFLAG		: KXT11 ??	::GPA
827	007446	001407			BEQ	1000\$		: BR IF NOT	::GPA
828	007450	023727	002124	000400	CMP	\$GDDAT,#400		: YES, REACHED VECTOR 400 ??	::GPA
829	007456	103403			BLO	1000\$		: NOT YET, CONTINUE	::GPA
830	007460	012737	000300	002124	MOV	#300,\$GDDAT		: YES, RESET EXPECTED TO 300	::GPA
831	007466								::GPA
832	007466	062704	000002		ADD	#2,R4		:NEXT BUFFER ADDRESS	
833	007472	005302			DEC	R2		:COMPLETED 64 BUFFER CHECKS?	
834	007474	001346			BNE	16\$		:CHECK NEXT VECTOR BUFFER LOCATION	
835								:THAT WAS STORED BY THE INTERRUPT	
836								:SERVICE ROUTINES.	
837									
838									
(3)									
(3)									

\*\*\*\*\*  
 :\*TEST 6 TEST VECTOR ADDR UNIQUENESS IN ROTATING MODE FOR GRPS 1,2  
 :\*\*\*\*\*

(2)	007476	000004		TST6:	SCOPE	
839						:BYTE COUNT = 4
841	007500	004537	011014		JSR R5,CLRCR	:LEVELS = 0-7
842	007504	004537	011334		JSR R5,CLRBF	:CLEAR ALL CSRS
843	007510	004537	011524		JSR R5,TRPCAT	:CLEAR VECTOR BUFFER AREA
844	007514	004537	011300		JSR R5,STRVEC	:RESTORE TRAP CATCHER
845						:STORE VECTOR AREA 300-674
846	007520	012737	000377	002456	MOV #377,VECVL	:WITH INT. SERV ROUTINES
847	007526	004537	011074		JSR R5,CLIRR	:VECTOR MEM FINAL VALUE
848	007532	012704	017160		MOV #VBUF,R4	:CLEAR IRR REGS
849	007536	004537	011132		JSR R5,VECFIL	:VECTOR ADDRESS BUFFER
850						:FILL VECTOR MEMORY FOR GROUPS 1,2
851	007542	012705	012210		MOV #SETVEC,R5	:WITH VECTORS 300-674
852	007546	013700	002412		MOV DRCSA,R0	: R5 POINTS TO COMMON HANDLER. ;;GPA
853	007552	013701	002422		MOV DRCSB,R1	:GROUP 1 CSR
854	007556	013702	002416		MOV DRCSB,R2	:GROUP 2 CSR
855	007562	013703	002426		MOV DRCSB,R3	
856	007566	010037	002122		MOV R0,\$BDADR	:STORE BUS ADDRESS
857	007572	012737	000010	002474	MOV #8,LVLSAV	:COUNTER FOR 7 LEVELS
858	007600	112710	000201		MOVB #201,(R0)	:ROTATING PRIORITY GROUP 1
859	007604	112710	000040		MOVB #CIMR,(R0)	:CLEAR ALL IMR BITS GP1
860	007610	005037	002436		CLR INTFLG	:CLEAR INT. FLAG
861	007614	106427	000000		MTPS #PRO	
862	007620	112710	000120		MOVB #SIRR,(R0)	:SET ALL IRR BITS
863	007624	112710	000241		MOVB #241,(R0)	:LMD57 - ARM THE CHIP
864	007630	106427	000340		MTPS #340	
865	007634	005737	002436		TST INTFLG	:CHECK FOR NO INTERRUPTS
866	007640	001401			BEQ 1\$	
867	007642	104011			ERROR 11	:ILLEGAL INTERRUPT
868	007644	005037	002436	1\$:	CLR INTFLG	
869	007650	112711	000201		MOVB #201,(R1)	:ROTATING PRIORITY FOR GROUP 2
870	007654	112711	000040		MOVB #CIMR,(R1)	:CLEAR ALL IMR BITS GP2
871	007660	106427	000000		MTPS #PRO	
872	007664	112711	000120		MOVB #SIRR,(R1)	:SET ALL IRR BITS
873	007670	112711	000241		MOVB #241,(R1)	:LMD57 - ARM THE CHIP
874	007674	106427	000340		MTPS #PR7	
875	007700	005737	002436		TST INTFLG	:CHECK FOR NO INTERRUPTS
876	007704	001401			BEQ 2\$	
877	007706	104011			ERROR 11	:ILLEGAL INTERRUPT
878	007710	005037	002436	2\$:	CLR INTFLG	
879	007714	106427	000000		MTPS #PRO	
880	007720	152760	000002	000001	BISB #BIT1,1(R0)	:SET INT. ENABLE, EXPECT INTERRUPTS
881	007726	012737	000004	002124	MOV #4,\$_JDAT	:EXPECT 4 INTERRUPTS
882	007734	013737	002436	002126	MOV INTFLG,\$BDDAT	:SHOULD HAVE 4 INTERRUPTS
883	007742	106427	000340		MTPS #PR7	
884	007746	023737	002124	002126	CMP \$GDDAT,\$BDDAT	
885	007754	001401			BEQ 100\$	
886	007756	104012			ERROR 12	:INTERRUPT TEST ERROR
887	007760	112710	000140	100\$:	MOVB #CHPISR,(R0)	:CLEAR HIGHEST PRIORITY ISR
888	007764	112710	000120		MOVB #SIRR,(R0)	:SET ALL IRR BITS
889	007770	005337	002474		DEC LVLSAV	:DONE 8 LEVELS, GROUP 1
890	007774	001345			BNE 2\$	:DO NEXT LEVEL IN ROTATION
891	007776	112710	000100		MOVB #CIRR,(R0)	:CLEAR IRR BITS GROUP 1
892	010002	012737	000010	002474	MOV #8,LVLSAV	:DO 8 LEVELS, GROUP 2
893	010010	005037	002436	101\$:	CLR INTFLG	:CLEAR INT FLAG

894	010014	106427	000000			MTPS	#PRO	
895	010020	012737	000004	002124		MOV	#4,\$GDDAT	
896	010026	013737	002436	002126		MOV	INTFLG,\$BDDAT	:SHOULD HAVE FOUR INTERRUPTS
897	010034	106427	000340			MTPS	#PR7	
898	010040	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
899	010046	001401				BEQ	102\$	
900	010050	104012				ERROR	12	:INTERRUPT TEST ERROR
901	010052	112711	000140		102\$:	MOVB	#CHPISR,(R1)	:CLEAR HIGHEST PRIORITY ISR
902	010056	112711	000120			MOVB	#SIRR,(R1)	:SET ALL IRR BITS
903	010062	005337	002474			DEC	LVL SAV	:DONE 8 LEVELS?
904	010066	001350				BNE	101\$	:DO NEXT LEVEL IN ROTATION,GP2
905	010070	112711	000100			MOVB	#CIRR,(R1)	:CLEAR IRR BITS,GROUP 2
906	010074	012737	101750	002124	3\$:	MOV	#101750,\$GDDAT	:RDY,DIR,I/E,CHIP - 35X
907	010102	010037	002122			MOV	R0,\$BDADR	:CSRA
908	010106	011037	002126			MOV	(R0),\$BDDAT	
909	010112	042737	000007	002126		BIC	#7,\$BDDAT	:CLEAR UNDEFINED BITS
910	010120	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
911	010126	001401				BEQ	4\$	
912	010130	104002				ERROR	2	:CSRA REG ERROR
913	010132	012737	000350	002124	4\$:	MOV	#350,\$GDDAT	:STORE EXPECTED
914	010140	010137	002122			MOV	R1,\$BDADR	:CSRC
915	010144	011137	002126			MOV	(R1),\$BDDAT	
916	010150	042737	000007	002126		BIC	#7,\$BDDAT	:CLEAR UNDEFINED BITS
917	010156	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
918	010164	001401				BEQ	5\$	
919	010166	104002				ERROR	2	:CSRC ERROR
920	010170	010237	002122		5\$:	MOV	R2,\$BDADR	:CSRB ADDRESS
921	010174	005037	002126			CLR	\$BDDAT	
922	010200	005037	002124			CLR	\$GDDAT	
923	010204	112710	000240			MOVB	#MISR,(R0)	:READ ISR
924	010210	111237	002126			MOVB	(R2),\$BDDAT	
925	010214	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
926	010222	001401				BEQ	6\$	
927	010224	104006				ERROR	6	:ISR ERROR,GP1
928	010226	112710	000244		6\$:	MOVB	#MIMR,(R0)	:MODE BITS FOR IMR
929	010232	111237	002126			MOVB	(R2),\$BDDAT	
930	010236	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
931	010244	001401				BEQ	7\$	
932	010246	104005				ERROR	5	:IMR ERROR,GP1
933	010250	112710	000250		7\$:	MOVB	#MIRR,(R0)	:MODE BITS FOR IRR
934	010254	111237	002126			MOVB	(R2),\$BDDAT	
935	010260	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
936	010266	001401				BEQ	10\$	
937	010270	104003				ERROR	3	:IRR ERROR
938	010272	112710	000254		10\$:	MOVB	#MACR,(R0)	:READ ACR
939	010276	111237	002126			MOVB	(R2),\$BDDAT	
940	010302	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
941	010310	001401				BEQ	11\$	
942	010312	104004				ERROR	4	:ACR ERROR
943	010314	010337	002122		11\$:	MOV	R3,\$BDADR	:CSRD ADDRESS
944	010320	112711	000240			MOVB	#MISR,(R1)	:READ ISR,GP2
945	010324	111337	002126			MOVB	(R3),\$BDDAT	
946	010330	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
947	010336	001401				BEQ	12\$	
948	010340	104006				ERROR	6	:ISR ERROR,GP1
949	010342	112711	000244		12\$:	MOVB	#MIMR,(R1)	:READ IMR GP2

```

950 010346 111337 002126      MOVB      (R3),SBDDAT
951 010352 023737 002124 002126      CMP      $GDDAT,SBDDAT
952 010360 001401                BEQ      13$
953 010362 104005                ERROR    5          :IMR ERROR,GP2
954
955 010364 112711 000250      13$:     MOVB      #MIRR,(R1)      :READ IRR
956 010370 111337 002126      MOVB      (R3),SBDDAT
957 010374 023737 002124 002126      CMP      $GDDAT,SBDDAT
958 010402 001401                BEQ      14$
959 010404 104003                ERROR    3          :IRR ERROR,GP2
960 010406 112711 000254      14$:     MOVB      #MACR,(R1)      :READ ACR
961 010412 111337 002126      MOVB      (R3),SBDDAT
962 010416 023737 002124 002126      CMP      $GDDAT,SBDDAT
963 010424 001401                BEQ      200$
964 010426 104004                ERROR    4          :ACR ERROR,GP2
965 010430 105010      200$:    CLRB      (R0)          :INIT GROUP 1 MODE BITS
966 010432 105011                CLRB      (R1)          :INIT GROUP 2 MODE BITS
967 010434 012737 101700 002124      MOV      #101700,$GDDAT :EXPECTED CSRA
968 010442 010037 002122      MOV      R0,$BDADR      :CSRA
969 010446 011037 002126      MOV      (R0),SBDDAT
970 010452 042737 000007 002126      BIC      #7,SBDDAT      :CLEAR UNDEFINED BITS
971 010460 023737 002124 002126      CMP      $GDDAT,SBDDAT
972 010466 001401                BEQ      201$
973 010470 104002                ERROR    2          :CSRA REG ERROR
974 010472 012737 000200 002124 201$:     MOV      #200,$GDDAT    :STORE EXPECTED
975 010500 010137 002122      MOV      R1,$BDADR      :CSRC
976 010504 011137 002126      MOV      (R1),SBDDAT
977 010510 042737 000007 002126      BIC      #7,SBDDAT      :CLEAR UNDEFINED BITS
978 010516 023737 002124 002126      CMP      $GDDAT,SBDDAT
979 010524 001401                BEQ      15$
980 010526 104002                ERROR    2          :CSRC ERROR
981 010530 012702 000100      15$:     MOV      #64,R2
982 010534 012704 017160      MOV      #VBUF,R4      :VECTOR BUFFER
983 010540 012737 000300 002124      MOV      #300,$GDDAT    :START WITH FIRST VECTOR 300
984 010546 010437 002122      16$:     MOV      R4,$BDADR      :VECTOR BUFFER ADDRESS
985 010552 011437 002126      MOV      (R4),SBDDAT
986 010556 023737 002124 002126      CMP      $GDDAT,SBDDAT
987 010564 001401                BEQ      17$
988 010566 104007                ERROR    7          :VECTORED PROPERLY
989 010570 062737 000004 002124 17$:     ADD      #4,$GDDAT      :VECTOR ADDR MEM ERROR
990 010576 005737 017374      TST      KXTFLAG        : KXT11 ??
991 010602 001407                BEQ      1000$         : BR IF NOT
992 010604 023727 002124 000400      CMP      $GDDAT,#400    : YES, REACHED VECTOR 400 ??
993 010612 103403                BLO      1000$         : NOT YET, CONTINUE
994 010614 012737 000300 002124      MOV      #300,$GDDAT    : YES, RESET EXPECTED TO 300
995 010622                1000$:
996 010622 062704 000002      ADD      #2,R4          :NEXT BUFFER ADDRESS
997 010626 005302                DEC      R2             :COMPLETED 64 BUFFER CHECKS?
998 010630 001346                BNE      16$           :CHECK NEXT VECTOR BUFFER LOCATION
999
1000
1001
1002
1003                :DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11J'S HAVE BEEN TESTED.
1004 010632 013701 002412      NXDEV:  MOV      DRCSA,R1 :INIT TO SETUP DRV11J ADDRESS
1005 010636 000241                NXDEV1: CLC           :CLEAR CARRY FOR DEVICE MAP
    
```

```

1006 010640 006037 002434 ROR DMAP ;LOOK FOR NEXT DRV11J
1007 010644 001412 BEQ SEOP ;BR IF ALL TESTED
1008 010646 162701 000020 SUB #20,R1 ;NEXT DRV11J STARTS -20 FROM PAST CSR
1009 010652 005237 002202 INC SUNIT ;UPDAT UNIT #
1010 010656 032737 000001 002434 BIT #1,DMAP ;IS UNIT SELECTED?
1011 010664 001764 BEQ NXDEV1 ;BR IF NOT
1012 010666 000137 003200 JMP NEXPAS ;TEST NEXT DRV11J
1013
1014 .SBTTL END OF PASS ROUTINE
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 010672 SEOP:
(2) 010672 000240 NOP
(1) 010674 005037 002102 CLR $TSTNM ;:ZERO THE TEST NUMBER
(1) 010700 005037 002160 CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
(1) 010704 005237 002176 INC $PASS ;:INCREMENT THE PASS NUMBER
(1) 010710 042737 100000 002176 BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
(1) 010716 005327 DEC (PC)+ ;:LOOP?
(1) 010720 000001 SEOPCT: .WORD 1
(1) 010722 003022 BGT $DOAGN ;:YES
(1) 010724 012737 MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
(1) 010726 000001 SENDCT: .WORD 1
(1) 010730 010720 SEOPCT
(1) 010732 104401 010777 TYPE $SENDMG ;:TYPE 'END PASS #'
(2) 010736 013746 002176 MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
(2) 010742 104405 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
(1) 010744 104401 010774 TYPE $ENULL ;:TYPE A NULL CHARACTER
(1) 010750 013700 000042 SGET42: MOV @#42,R0 ;:GET MONITOR ADDRESS
(1) 010754 001405 BEQ $DOAGN ;:BRANCH IF NO MONITOR
(1) 010756 000005 RESET ;:CLEAR THE WORLD
(1) 010760 004710 SENDAD: JSR PC,(R0) ;:GO TO MONITOR
(1) 010762 000240 NOP ;:SAVE ROOM
(1) 010764 000240 NOP ;:FOR
(1) 010766 000240 NOP ;:ACT11
(1) 010770 SDOAGN:
(1) 010770 000137 JMP @(PC)+ ;:RETURN
(1) 010772 003134 SRTNAD: .WORD START1
(1) 010774 377 377 000 SENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
(1) 010777 015 042412 042116 SENDMG: .ASCIZ <15><12>/END PASS #/

```

.SBTTL PROGRAM SUBROUTINES

1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071

```
*****  
:CLEAR ALL CONTROL/STATUS REGISTERS  
*****  
CLRCSR: MOV R0,-(SP) :SAVE R0  
MTPS #PR7 :PSW = 340  
MOV #PVMA,VECVAL :INIT VECTOR ADDR MEM = 340  
MOV #300,VECLOC :INIT VECTOR ADDR TO 300  
MOV DRCSA,R0 :START OF CSR ADDRESS  
CLR $GDDAT :CLEAR EXPECTED  
CLR $BDDAT :CLEAR REC'D  
CLR (R0) :CLEAR CSRA;CHIP RESET GROUP 1  
CLRB 5(R0) :CLEAR HIGH BYTE CSRB  
CLR 10(R0) :CLEAR CSRC;CHIP RESET GROUP 2  
CLRB 15(R0) :CLEAR HIGH BYTE CSRD  
MOV (SP)+,R0 :RETURN R0  
RTS R5  
  
:CLEAR IRR REGISTERS, GROUP 1, GROUP 2 WITH CHIP RESET  
CLRIRR: MOV R0,-(SP)  
MOV DRCSA,R0 :START OF CSR ADDRESS  
CLRB 11(R0) :CSRC TO INPUT MODE  
MOVB #BIT0,1(R0) :CSRA TO OUTPUT MODE  
CLR 2(R0) :CLEAR DBRA  
CLRB (R0) :CHIP RESET OF GROUP1  
CLRB 10(R0) :CHIP RESET OF GROUP 2  
MOV (SP)+,R0 :RESTORE REGISTER  
RTS R5 :EXIT  
  
:ROUTINE TO FILL VECTOR MEMORY FOR VECTOR UNIQUENESS TEST  
:300-474 GROUP 1  
:500-674 GROUP 2  
VECFIL: MOV DRCSA,R0 :START GROUP 1  
MOV DRCSB,R1  
MOV #2,R2 :TWO GROUPS  
MOV #60,VECPAT :VECTOR START 300  
1$: MOV #370,VECVAL :START VECTOR LEVEL 0,BY4  
MOV #8,LVLCNT :LEVELS 0-7  
2$: MOVB VECVAL,(R0)  
MOV #4,BYCNT :DO FOUR BYTE COUNTS  
3$: MOVB VECPAT,(R1) :WRITE VECTOR  
INC VECPAT :SETUP FOR NEXT VECTOR  
TST KXTFLAG : KXT11 ?? :GPA  
BEQ 1000$ : BR IF NOT :GPA  
CMP VECPAT,#100 : YES, REACED VECTOR 400 ?? :GPA  
BLO 1000$ : NOT YET, CONTINUE :GPA  
MOV #60,VECPAT : YES, WRAP-AROUND TO 300 AGAIN :GPA  
: WE'LL END UP WITH 300-374... :GPA  
: ...REPLICATED 4 TIMES. :GPA  
1000$:  
4$: DEC BYCNT :DONE FOUR BYTE COUNTS?  
BNE 3$ :DO NEXT VECTOR  
DEC LVLCNT :DONE ALL LEVELS IN GROUP?
```

```

1072 011250 001403          BEQ      5$
1073 011252 005237 002456  INC      VECVAL      ;INC NEXT LEVEL
1074 011256 000744          BR       2$          ;DO NEXT LEVEL
1075 011260 005302          5$: DEC      R2          ;DONE BOTH GROUPS?
1076 011262 001405          BEQ      6$          ;DONE,EXIT
1077 011264 013700 002422  MOV      DRCSC,R0    ;DO GROUP 2
1078 011270 013701 002426  MOV      DRCSD,R1
1079 011274 000727          BR       1$
1080 011276 000205          6$: RTS      R5          ;EXIT
1081
1082          ;ROUTINE TO STORE VECTOR AREA 300-674 WITH INTERRUPT SERVICE
1083 011300 012700 000300  STRVEC: MOV      #300,R0      ;START AT VECTOR 300
1084 011304 012701 012236  MOV      #INT0,R1      ;FIRST SERVICE ROUTINE
1085 011310 012702 000100  MOV      #64,R2        ;STORE 64 SERVICE ROUTINES
1086 011314 010120          1$: MOV      R1,(R0)+
1087 011316 012720 000340  MOV      #340,(R0)+
1088          ;;GPA ADD      #6,R1      ;SETUP FOR NEXT VECTOR STORAGE
1089 011322 062701 000004  ADD      #4,R1      ;SETUP NEXT VETOR.      ;;GPA
1090 011326 005302          DEC      R2          ;DONE WITH 64 VECTOR ROUTINES?
1091 011330 001371          BNE      1$
1092 011332 000205          RTS      R5          ;EXIT
1093
1094          ;ROUTINE TO CLEAR VECTOR BUFFER
1095 011334 012700 017160  CLRBF: MOV      #VBUF,R0    ;VECTOR BUFFER START
1096 011340 012701 000100  MOV      #64,R1
1097 011344 005020          1$: CLR      (R0)+
1098 011346 005301          DEC      R1
1099 011350 001375          BNE      1$
1100 011352 000205          RTS      R5
1101
1102          ;ROUTINE TO STORE 0-202 INTO BUFFER AREA TO ALLOW
1103          ;VECTOR TESTS TO 0-200.
1104 011354 005000          TOBUF: CLR      R0
1105 011356 012701 016754  MOV      #DBUF,R1      ;BUFFER START
1106 011362 012021          1$: MOV      (R0)+,(R1)+    ;STORE 0-202 INTO BUFFER
1107 011364 022700 000204  CMP      #204,R0      ;FINISHED?
1108 011370 001374          BNE      1$          ;NO
1109 011372 000205          RTS      R5          ;RETURN
1110
1111          ;STORE BUFFER AREA INTO LOCATIONS 0-202
1112          ;AFTER VECTOR TESTS
1113 011374 010046          FRMBUF: MOV      R0,-(SP)    ;SAVE R0
1114 011376 010146          MOV      R1,-(SP)
1115 011400 005000          CLR      R0
1116 011402 012701 016754  1$: MOV      #DBUF,R1      ;BUFFER START
1117 011406 012120          MOV      (R1)+,(R0)+    ;STORE BUFFER INTO LOCS 0-202
1118 011410 022700 000204  CMP      #204,R0      ;FINISHED?
1119 011414 001374          BNE      1$          ;NO
1120 011416 013777 002460 170514  MOV      SWRSV,SWR     ;STORE LOCATION 176,SOFT SWR
1121 011424 012601          MOV      (SP)+,R1      ;RESTORE R1
1122 011426 012600          MOV      (SP)+,R0      ;RESTORE R0
1123 011430 000205          RTS      R5          ;RETURN
1124
1125          ;RESTORE VECTOR JUST TESTED TO ORIGINAL TRAP FOR 204-1774
1126 011432 022737 000001 002210  RESTRP: CMP      #1,SENV    ;CHECK FOR APT
1127 011440 001421          BEQ      1$          ;YES,BRANCH

```



```

1128 011442 032777 010000 170470      BIT      #SW12,@SWR      ;CHECK TYPE OF TRAP RETURN
1129 011450 001412                BEQ      10$           ;RESTORE ILLEGAL VECTOR ROUTINE
1130 011452 062737 000002 002452      ADD      #2,VECLOC    ;RESTORE TRAP
1131 011460 013723 002452                MOV      VECLOC,(R3)+ ;TO VECTOR
1132 011464 005013                CLR      (R3)         ;RESTORE HALT
1133 011466 062737 000002 002452      ADD      #2,VECLOC    ;SETUP FOR NEXT
1134 011474 000412                BR       2$           ;RETURN
1135 011476 012723 011752      10$:    MOV      #TRPALL,(R3)+ ;RESTORE ILLEGAL TRAP ROUTINE
1136 011502 000402                BR       11$         ;RESTORE PSW
1137 011504 012723 011734      1$:    MOV      #TRPOUT,(R3)+ ;RESTORE APT SUB TRAP
1138                                ;IF ON APT
1139 011510 012713 000340      11$:    MOV      #340,(R3)    ;RESTORE PSW SAVE
1140 011514 062737 000004 002452      ADD      #4,VECLOC    ;UPDATE FOR NEXT VECTOR
1141 011522 000205                RTS      R5           ;RETURN
1142
1143                                ;ROUTINE TO SET UP TRAP CATCHER 204-1774 IN STANDALONE MODE OR
1144                                ;A COMMON VECTOR ROUTINE SETUP UNDER APT.
1145                                ;VECTOR ROUTINE IS USED ONLY IN AN APT ENVIRONMENT AND WILL STORE
1146                                ;A COMMON SUBROUTINE IN VECTOR AREA 204-1774.
1147
1148 011524 010046      TRPCAT: MOV      R0,-(SP)    ;SAVE R0
1149 011526 010146                MOV      R1,-(SP)
1150 011530 022737 000001 002210      CMP      #1,$ENV     ;CHECK FOR APT
1151 011536 001426                BEQ      APTVEC      ;YES,SET UP TRAPS FOR APT
1152 011540 012700 000204                MOV      #204,R0     ;START CATCHER AT 204
1153 011544 032777 010000 170366      BIT      #SW12,@SWR  ;TEST IF SW12 IS SET
1154 011552 001010                BNE      1$         ;YES,BR AND STORE REGULAR TRAP
1155                                ;CATCHER(+2,HALT) IN LOCATIONS
1156                                ;204-1774
1157 011554 012720 011752      11$:    MOV      #TRPALL,(R0)+ ;NO,PLACE ILLEGAL VECTOR RETURN
1158 011560 012720 000340                MOV      #340,(R0)+  ;ROUTINES IN LOCATIONS 204-1774
1159 011564 022700 002000                CMP      #2000,R0   ;STORED ALL VECTORS?
1160 011570 001371                BNE      11$
1161 011572 000421                BR       ENDTRP     ;RETURN AFTER VECTOR STORAGE
1162 011574 010001      1$:    MOV      R0,R1     ;DRV11J CAN VECTOR 0-1774
1163 011576 005721                TST      (R1)+
1164 011600 010120                MOV      R1,(R0)+
1165 011602 005020                CLR      (R0)+
1166 011604 022701 001776      CMP      #1776,R1   ;CHECK FOR VECTOR END
1167 011610 001371                BNE      1$
1168 011612 000411                BR       ENDTRP     ;RETURN IN STAND ALONE MODE
1169 011614 012701 000204      APTVEC: MOV      #204,R1 ;STARTING VECTOR ADDRESS
1170 011620 012721 011734      1$:    MOV      #TRPOUT,(R1)+ ;ILL INTERRUPT ROUTINE
1171 011624 012721 000340                MOV      #340,(R1)+ ;PSW NEXT
1172 011630 022701 002000                CMP      #2000,R1   ;DONE?
1173 011634 001371                BNE      1$         ;DO NEXT VECTOR
1174 011636 012601      ENDTRP: MOV      (SP)+,R1
1175 011640 012600                MOV      (SP)+,R0
1176 011642 000205                RTS      R5         ;EXIT
1177
1178                                ;INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 204-1774
1179 011644 005237 002436      INTSR1: INC      INTFLG ;COUNT INTERRUPT
1180 011650 012737 000001 002124      MOV      #1,$GDDAT  ;STORE EXPECTED
1181 011656 013737 002436 002126      MOV      INTFLG,$BDDAT ;SAVE INT. COUNT
1182 011664 023737 002124 002126      CMP      $GDDAT,$BDDAT ;SHOULD BE ONE INTERRUPT
1183 011672 001401                BEQ      1$

```

```

1184 011674 104013          ERROR 13          :MULTIPLE INTERRUPTS RECEIVED
1185 011676 000002          RTI              :RETURN FROM INT.
1186
1187 ;INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 204-1774
1188 ;FOR A BYTE COUNT OF 1 TO 4 VECTORS IN VECTOR ADDR MEMORY.
1189 011700 005237 002436          INTBY4: INC      INTFLG          :COUNT INTERRUPT
1190 011704 013737 002472 002124      MOV      BYNUM,$GDDAT        :STORE EXPECTED
1191 011712 013737 002436 002126      MOV      INTFLG,$BDDAT       :SAVE INT. COUNT
1192 011720 023737 002124 002126      CMP      $GDDAT,$BDDAT      :INTERRUPTS SHOULD NOT EXCEED BYTE COUNT
1193 011726 002001          BGE      1$
1194 011730 104013          ERROR 13          :MULTIPLE INTERRUPTS RECEIVED
1195 ;MORE INTERRUPTS THAN BYTE COUNT
1196 011732 000002          RTI              :RETURN FROM INT.
1197
1198 ;COMMON ILLEGAL VECTOR RETURN ROUTINE FOR APT MODE.
1199 011734 011637 002120      TRPOUT: MOV      (SP),$GDADR    :SAVE ADDRESS OF TEST
1200 011740 004537 011374      JSR      R5,FRMBUF          :RETURN LOCATIONS 0-202
1201 011744 104014          ERROR 14          :ILLEGAL VECTOR ADDR MEM ERROR
1202 011746 000000          HALT             :CANNOT CONTINUE
1203 011750 000000          HALT             :CANNOT CONTINUE
1204
1205 ;COMMON ILLEGAL VECTOR RETURN ROUTINE FOR VECTOR AREA
1206 ;204-1774
1207 011752 011637 002120      TRPALL: MOV      (SP),$GDADR    :SAVE PC OF TEST
1208 011756 104014          ERROR 14          :ILLEGAL VECTOR ADDR MEM ERROR
1209 011760 000002          RTI              :RETURN
1210
1211 ;RESTORE VECTOR JUST TESTED TO ERROR SUBROUTINE FOR 0-202
1212 011762 022737 000001 002210      RES200: CMP      #1,$ENV        :CHECK FOR APT
1213 011770 001405          BEQ      1$          :YES,BRANCH
1214 011772 012723 012170      MOV      #TRP200,(R3)+      :ERROR TRAP SUB. FOR 0-200
1215 011776 012713 000340      MOV      #340,(R3)          :PSW
1216 012002 000404          BR       2$          :RETURN
1217 012004 012723 011734          1$: MOV      #TRPOUT,(R3)+      :RESTORE APT SUB TRAP
1218 ;IF ON APT
1219 012010 012713 000340          MOV      #340,(R3)          :RESTORE PSW SAVE
1220 012014 062737 000004 002452      2$: ADD      #4,VECLOC        :UPDATE FOR NEXT VECTOR
1221 012022 000205          RTS      R5            :RETURN
1222
1223 ;ROUTINE TO SET UP TRAP SUBROUTINE 0-200 IN STANDALONE MODE OR
1224 ;A COMMON ERROR VECTOR ROUTINE SETUP UNDER APT.
1225
1226 012024 010046          CAT200: MOV      R0,-(SP)          :SAVE R0
1227 012026 017737 170106 002460      MOV      @SWR,SWRSV          :SAVE LOCATION 176,SOFT SWR
1228 012034 022737 000001 002210      CMP      #1,$ENV          :CHECK FOR APT
1229 012042 001411          BEQ      APT200          :YES,SET UP TRAPS FOR APT
1230 012044 005000          CLR      R0              :START CATCHER SUB AT 0
1231 012046 012720 012170          1$: MOV      #TRP200,(R0)+      :STORE VECTOR TRAP SUBROUTINE
1232 012052 012720 000340      MOV      #340,(R0)+        :PSW
1233 012056 022700 000204      CMP      #204,R0          :FINISHED 0-202
1234 012062 001371          BNE      1$
1235 012064 000410          BR       END200          :RETURN IN STAND ALONE MODE
1236 012066 005000          APT200: CLR      R0              :START AT VECTOR 0
1237 012070 012720 011734          1$: MOV      #TRPOUT,(R0)+      :VECTOR ERROR ROUTINE
1238 012074 012720 000340      MOV      #340,(R0)+        :PSW NEXT
1239 012100 022700 000204      CMP      #204,R0          :DONE?

```

```

1240 012104 001371          BNE      1$          :DO NEXT VECTOR
1241 012106 012600          END200: MOV      (SP)+,R0
1242 012110 012713 012124  MOV      #INTSR3,(R3)  :STORE VECTOR ROUTINE
1243 012114 012763 000340  MOV      #340,2(R3)   :STORE PSW
1244 012122 000205          RTS       R5          :EXIT
1245
1246
1247          :INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 0-200,BYTE COUNT 0-3
1248          :FOR BYTE COUNTS OF 1 TO 4 VECTORS IN VECTOR ADDRESS MEMORY.
1249 012124 005237 002436  INTSR3: INC      INTFLG      :COUNT INTERRUPT
1250 012130 013737 002472 002124  MOV      BYNUM,$GDDAT  :STORE EXPECTED
1251 012136 013737 002436 002126  MOV      INTFLG,$BDDAT :SAVE INT. COUNT
1252 012144 023737 002124 002126  CMP      $GDDAT,$BDDAT :INTERRUPTS SHOULD NOT EXCEED BYTE COUNT
1253 012152 002005          BGE      1$
1254 012154 004537 011374          JSR      R5,FRMBUF    :RESTORE 0-202 BEFORE ERROR
1255 012160 104013          ERROR   13          :MULTIPLE INTERRUPTS RECEIVED
1256          :MORE INTERRUPTS THAN BYTE COUNT
1257 012162 004537 012024          JSR      R5,CAT200   :RESTORE VECTOR SUBROUTINES
1258 012166 000002          1$:      RTI       :RETURN FROM INT.
1259
1260          :COMMON ILLEGAL VECTOR ERROR RETURN ROUTINE FOR 0-200
1261 012170 011637 002120  TRP200: MOV      (SP),$GADR  :SAVE PC OF TEST
1262 012174 004537 011374          JSR      R5,FRMBUF    :RETURN LOC 0-200
1263 012200 104014          ERROR   14          :ILLEGAL VECTOR ADDR MEM ERROR
1264 012202 004537 012024          JSR      R5,CAT200   :RESTORE VECTOR SUBROUTINES FOR PROCEED
1265 012206 000002          RTI       :RETURN
1266
1267          :VECTOR SERVICE ROUTINE FOR VECTOR UNIQUENESS TEST
1268          :SETVEC: MOV      (R5)+,(R4)+  :STORE VALUE IN BUFFER  ::GPA
1269 012210 017624 000000  SETVEC: MOV      @ (SP),(R4)+  :STORE VECTOR NUMBER IN BUFFER ::GPA
1270 012214 005726          TST      (SP)+        :RESTORE STACK FOR RTI
1271 012216 005237 002436  INC      INTFLG      :COUNT INTERRUPT
1272 012222 022737 000100 002436  CMP      #64.,INTFLG
1273 012230 002001          BGE      1$          :64 EXPECTED INTERRUPTS?
1274 012232 104013          ERROR   13          :ERROR ,MORE THAN 64 INTERRUPTS
1275 012234 000002          1$:      RTI       :RETURN FOR NEXT INTERRUPT
1276
1277          :INTERRUPT SERVICE ROUTINES THAT ARE STORED FROM 300-674
1278          :IN THE VECTOR UNIQUENESS TEST.
1279          : R5 POINTS TO "SETVEC:" DURING TESTS 5 AND 6, AND CALLS          ::GPA
1280          : CHANGED FROM "JSR R5,SETVEC" TO "JSR PC,(R5)" TO CONSERVE      ::GPA
1281          : SPACE AND REMAIN UNDER 4KW TOTAL PROGRAM SIZE.                ::GPA
1282 012236 004715          INTO:   JSR      PC,(R5) ;STORE 300 INTO VBUF,TO BE          ::GPA
1283 012240 000300          300          :CHECKED AFTER ALL INTERRUPTS.
1284 012242 004715          INT1:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ::GPA
1285 012244 000304          304
1286 012246 004715          INT2:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ::GPA
1287 012250 000310          310
1288 012252 004715          INT3:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ::GPA
1289 012254 000314          314
1290 012256 004715          INT4:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ::GPA
1291 012260 000320          320
1292 012262 004715          INT5:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ::GPA
1293 012264 000324          324
1294 012266 004715          INT6:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ::GPA
1295 012270 000330          330

```

1296	012272	004715	INT7:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1297	012274	000334		334		
1298	012276	004715	INT8:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1299	012300	000340		340		
1300	012302	004715	INT9:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1301	012304	000344		344		
1302	012306	004715	INT10:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1303	012310	000350		350		
1304	012312	004715	INT11:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1305	012314	000354		354		
1306	012316	004715	INT12:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1307	012320	000360		360		
1308	012322	004715	INT13:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1309	012324	000364		364		
1310	012326	004715	INT14:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1311	012330	000370		370		
1312	012332	004715	INT15:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1313	012334	000374		374		
1314	012336	004715	INT16:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1315	012340	000400		400		
1316	012342	004715	INT17:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1317	012344	000404		404		
1318	012346	004715	INT18:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1319	012350	000410		410		
1320	012352	004715	INT19:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1321	012354	000414		414		
1322	012356	004715	INT20:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1323	012360	000420		420		
1324	012362	004715	INT21:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1325	012364	000424		424		
1326	012366	004715	INT22:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1327	012370	000430		430		
1328	012372	004715	INT23:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1329	012374	000434		434		
1330	012376	004715	INT24:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1331	012400	000440		440		
1332	012402	004715	INT25:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1333	012404	000444		444		
1334	012406	004715	INT26:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1335	012410	000450		450		
1336	012412	004715	INT27:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1337	012414	000454		454		
1338	012416	004715	INT28:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1339	012420	000460		460		
1340	012422	004715	INT29:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1341	012424	000464		464		
1342	012426	004715	INT30:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1343	012430	000470		470		
1344	012432	004715	INT31:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1345	012434	000474		474		
1346	012436	004715	INT32:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1347	012440	000500		500		
1348	012442	004715	INT33:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1349	012444	000504		504		
1350	012446	004715	INT34:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1351	012450	000510		510		

1352	012452	004715	INT35:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1353	012454	000514		514		
1354	012456	004715	INT36:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1355	012460	000520		520		
1356	012462	004715	INT37:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1357	012464	000524		524		
1358	012466	004715	INT38:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1359	012470	000530		530		
1360	012472	004715	INT39:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1361	012474	000534		534		
1362	012476	004715	INT40:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1363	012500	000540		540		
1364	012502	004715	INT41:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1365	012504	000544		544		
1366	012506	004715	INT42:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1367	012510	000550		550		
1368	012512	004715	INT43:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1369	012514	000554		554		
1370	012516	004715	INT44:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1371	012520	000560		560		
1372	012522	004715	INT45:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1373	012524	000564		564		
1374	012526	004715	INT46:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1375	012530	000570		570		
1376	012532	004715	INT47:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1377	012534	000574		574		
1378	012536	004715	INT48:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1379	012540	000600		600		
1380	012542	004715	INT49:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1381	012544	000604		604		
1382	012546	004715	INT50:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1383	012550	000610		610		
1384	012552	004715	INT51:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1385	012554	000614		614		
1386	012556	004715	INT52:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1387	012560	000620		620		
1388	012562	004715	INT53:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1389	012564	000624		624		
1390	012566	004715	INT54:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1391	012570	000630		630		
1392	012572	004715	INT55:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1393	012574	000634		634		
1394	012576	004715	INT56:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1395	012600	000640		640		
1396	012602	004715	INT57:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1397	012604	000644		644		
1398	012606	004715	INT58:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1399	012610	000650		650		
1400	012612	004715	INT59:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1401	012614	000654		654		
1402	012616	004715	INT60:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1403	012620	000660		660		
1404	012622	004715	INT61:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1405	012624	000664		664		
1406	012626	004715	INT62:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN	::GPA
1407	012630	000670		670		

1408 012632 004715  
1409 012634 000674  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418 012636 001 376  
1419 012640 002 375  
1420 012642 004 373  
1421 012644 010 367  
1422 012646 020 357  
1423 012650 040 337  
1424 012652 100 277  
1425 012654 200 177  
1426 012656 000000  
1427  
1428

INT63: JSR PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;:GPA  
674

.SBTTL PATTERNS FOR REGISTER R/W  
: PATTERNS USED FOR LOADING/READING REGISTERS  
:ISR INTERRUPT SERVICE REGISTER  
:IRR INTERRUPT REQUEST REGISTER  
:IMR INTERRUPT MASK REGISTER

BGCHP3: .BYTE 1,376  
.BYTE 2,375  
.BYTE 4,373  
.BYTE 10,367  
.BYTE 20,357  
.BYTE 40,337  
.BYTE 100,277  
.BYTE 200,177

EDCHP3: 000000

```
1430 .SBTTL SYSMAC ROUTINES
1431
1432 .SBTTL TYPE ROUTINE
(1) *****
(2) *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1) *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1) *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1) *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1) *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1) *
(1) *CALL:
(1) *1) USING A TRAP INSTRUCTION
(1) * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1) *OR
(1) * TYPE
(1) * MESADR
(1) *
(1) $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
(1) 012660 105737 002157 BPL 1$ ;;BR IF YES
(1) 012664 100002 HALT ;;HALT HERE IF NO TERMINAL
(1) 012666 000000 BR 3$ ;;LEAVE
(1) 012670 000430 1$: MOV R0,-(SP) ;;SAVE R0
(1) 012672 010046 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
(1) 012674 017600 000002 002210 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 012700 122737 000001 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
(1) 012706 001011 BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1) 012710 132737 000100 002211 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
(1) 012716 001405 MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
(1) 012720 010037 012730 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
(1) 012724 004737 013222 61$: .WORD 0 ;;MESSAGE ADDRESS
(1) 012730 000000 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 012732 132737 000040 002211 BNE 60$ ;;YES,SKIP TYPE OUT
(1) 012740 001003 2$: MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 012742 112046 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
(1) 012744 001005 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
(1) 012746 005726 60$: MOV (SP)+,R0 ;;RESTORE R0
(1) 012750 012600 3$: ADD #2,(SP) ;;ADJUST RETURN PC
(1) 012752 062716 000002 RTI ;;RETURN
(1) 012756 000002 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
(1) 012760 122716 000011 BEQ 8$
(1) 012764 001430 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
(1) 012766 122716 BNE 5$
(1) 012772 001006 TST (SP)+ ;;POP <CR><LF> EQUIV
(1) 012774 005726 TYPE ;;TYPE A CR AND LF
(1) 012776 104401 $CRLF
(1) 013000 002165 CLR B $CHARCNT ;;CLEAR CHARACTER COUNT
(1) 013002 105037 013210 BR 2$ ;;GET NEXT CHARACTER
(1) 013006 000755 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
(1) 013010 004737 013072 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
(1) 013014 123726 002156 BNE 2$ ;;IF NO GO GET NEXT CHAR.
(1) 013020 001350 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
(1) 013022 013746 002154 AND THE NULL CHAR.
(1) 013026 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
(1) 013032 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

```
(1) 013034 004737 013072 JSR PC,$TYPEC ::GO TYPE A NULL
(1) 013040 105337 013210 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
(1) 013044 000770 BR 7$ ::LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 013046 112716 000040 8$: MOVB #' (SP) ::REPLACE TAB WITH SPACE
(1) 013052 004737 013072 9$: JSR PC,$TYPEC ::TYPE A SPACE
(1) 013056 132737 000007 013210 BITB #7,$CHARCNT ::BRANCH IF NOT AT
(1) 013064 001372 BNE 9$ ::TAB STOP
(1) 013066 005726 TST (SP)+ ::POP SPACE OFF STACK
(1) 013070 000724 BR 2$ ::GET NEXT CHARACTER
(1) 013072 $TYPEC:
(1) 013072 105777 167046 TSTB @STKS ::CHAR IN KYBD BUFFER? :MJD001
(1) 013076 100022 BPL 10$ ::BR IF NOT :MJD001
(1) 013100 017746 167042 MOV @STKB,-(SP) ::GET CHAR :MJD001
(1) 013104 042716 177600 BIC #177600,(SP) ::STRIP EXTRANEIOUS BITS :MJD001
(1) 013110 122716 000023 CMPB #$XOFF,(SP) ::WAS CHAR XOFF :MJD001
(1) 013114 001012 BNE 102$ ::BR IF NOT :MJD001
(1) 013116 105777 167022 101$: TSTB @STKS ::WAIT FOR CHAR :MJD001
(1) 013122 100375 BPL 101$ :MJD001
(1) 013124 117716 167016 MOVB @STKB,(SP) ::GET CHAR :MJD001
(1) 013130 042716 177600 BIC #177600,(SP) ::STRIP IT :MJD001
(1) 013134 122716 000021 CMPB #$XON,(SP) ::WAS IT XON? :MJD001
(1) 013140 001366 BNE 101$ ::BR IF NOT :MJD001
(1) 013142 105777 167000 102$: TST (SP)+ ::FIX STACK :MJD001
(1) 013144 105777 167000 10$: TSTB @STPS ::WAIT UNTIL PRINTER IS READY :MJD001
(1) 013150 100375 BPL 10$ :MJD001
(1) 013152 116677 000002 166772 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 013160 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
(1) 013166 001003 BNE 1$ ::BRANCH IF NO
(1) 013170 105037 013210 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT
(1) 013174 000406 BR $TYPEX ::EXIT
(1) 013176 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
(1) 013204 001402 BEQ $TYPEX ::BRANCH IF YES
(1) 013206 105227 INCB (PC)+ ::COUNT THE CHARACTER
(1) 013210 000000 $CHARCNT: WORD 0 ::CHARACTER COUNT STORAGE
(1) 013212 000207 $TYPEX: RTS PC

1433 .SBTTL APT COMMUNICATIONS ROUTINE
(1)
(2) *****
(1) 013214 112737 000001 013460 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
(1) 013222 112737 000001 013456 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
(1) 013230 000403 BR $ATYC
(1) 013232 112737 000001 013460 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
(1) 013240 $ATYC:
(3) 013240 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
(3) 013242 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
(1) 013244 105737 013456 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
(1) 013250 001450 BEQ 5$ ::IF NOT: BR
(1) 013252 122737 000001 002210 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
(1) 013260 001031 BNE 3$ ::IF NOT: BR
```



```

(1) 013262 132737 000100 002211 BITB #APTSPOOL,SENV :: SHOULD SPOOL MESSAGES?
(1) 013270 001425 BEQ 3$ :: IF NOT: BR
(1) 013272 017600 000004 MOV @4(SP),R0 :: GET MESSAGE ADDR.
(1) 013276 062766 000002 000004 ADD #2,4(SP) :: BUMP RETURN ADDR.
(1) 013304 005737 002170 1$: TST SMSGTYPE :: SEE IF DONE W/ LAST XMISSION?
(1) 013310 001375 BNE 1$ :: IF NOT: WAIT
(1) 013312 010037 002204 MOV R0,SMSGAD :: PUT ADDR IN MAILBOX
(1) 013316 105720 2$: TSTB (R0)+ :: FIND END OF MESSAGE
(1) 013320 001376 BNE 2$
(1) 013322 163700 002204 SUB SMSGAD,R0 :: SUB START OF MESSAGE
(1) 013326 006200 ASR R0 :: GET MESSAGE LGTH IN WORDS
(1) 013330 010037 002206 MOV R0,SMSGLGT :: PUT LENGTH IN MAILBOX
(1) 013334 012737 000004 002170 MOV #4,SMSGTYPE :: TELL APT TO TAKE MSG.
(1) 013342 000413 BR 5$
(1) 013344 017637 000004 013370 3$: MOV @4(SP),4$ :: PUT MSG ADDR IN JSR LINKAGE
(1) 013352 062766 000002 000004 ADD #2,4(SP) :: BUMP RETURN ADDRESS
(3) 013360 013746 177776 MOV 177776,-(SP) :: PUSH 177776 ON STACK
(1) 013364 004737 012660 JSR PC,$TYPE :: CALL TYPE MACRO
(1) 013370 000000 4$: .WORD 0
(1) 013372 5$:
(1) 013372 105737 013460 10$: TSTB $FFLG :: SHOULD REPORT FATAL ERROR?
(1) 013376 001416 BEQ 12$ :: IF NOT: BR
(1) 013400 005737 002210 TST $ENV :: RUNNING UNDER APT?
(1) 013404 001413 BEQ 12$ :: IF NOT: BR
(1) 013406 005737 002170 11$: TST SMSGTYPE :: FINISHED LAST MESSAGE?
(1) 013412 001375 BNE 11$ :: IF NOT: WAIT
(1) 013414 017637 000004 002172 MOV @4(SP),$FATAL :: GET ERROR #
(1) 013422 062766 000002 000004 ADD #2,4(SP) :: BUMP RETURN ADDR.
(1) 013430 005237 002170 INC SMSGTYPE :: TELL APT TO TAKE ERROR
(1) 013434 105037 013460 12$: CLRB $FFLG :: CLEAR FATAL FLAG
(1) 013440 105037 013457 CLRB $LFLG :: CLEAR LOG FLAG
(1) 013444 105037 013456 CLRB $MFLG :: CLEAR MESSAGE FLAG
(3) 013450 012601 MOV (SP)+,R1 :: POP STACK INTO R1
(3) 013452 012600 MOV (SP)+,R0 :: POP STACK INTO R0
(1) 013454 000207 RTS PC :: RETURN
(1) 013456 000 .SMFLG: .BYTE 0 :: MESSG. FLAG
(1) 013457 000 .SLFLG: .BYTE 0 :: LOG FLAG
(1) 013460 000 .SFFLG: .BYTE 0 :: FATAL FLAG
(1) 013462 .EVEN
(1) 000200 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTSPOOL=100
(1) 000040 APTCSUP=040

```

1434

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

(1) *****
(2) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1) *OCTAL (ASCII) NUMBER AND TYPE IT.
(1) *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1) *CALL:
(1) * MOV NUM,-(SP) :: NUMBER TO BE TYPED
(1) * TYPOS :: CALL FOR TYPEOUT
(1) * .BYTE N :: N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1) * .BYTE M :: M=1 OR 0
(1) * :: 1=TYPE LEADING ZEROS
(1) * :: 0=SUPPRESS LEADING ZEROS

```

```

(1)          : *
(1)          : *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)          : *$TYPOS OR $TYPOC
(1)          : *CALL:
(1)          : *      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
(1)          : *      TYPON      ::CALL FOR TYPEOUT
(1)          : *
(1)          : *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)          : *CALL:
(1)          : *      MOV      NUM,-(SP)      ::NUMBER TO BE TYPED
(1)          : *      TYPOC      ::CALL FOR TYPEOUT
(1)          :
(1) 013462 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ::PICKUP THE MODE
(1) 013466 116637 000001 013705  MOVB     1(SP), $OFILL      ::LOAD ZERO FILL SWITCH
(1) 013474 112637 013707          MOVB     (SP)+, $OCNT+1    ::NUMBER OF DIGITS TO TYPE
(1) 013500 062716 000002          ADD      #2, (SP)      ::ADJUST RETURN ADDRESS
(1) 013504 000406          BR       $TYPON
(1) 013506 112737 000001 013705  $TYPOC: MOVB     #1, $OFILL      ::SET THE ZERO FILL SWITCH
(1) 013514 112737 000006 013707  MOVB     #6, $OCNT+1    ::SET FOR SIX(6) DIGITS
(1) 013522 112737 000005 013704  $TYPON: MOVB     #5, $OCNT      ::SET THE ITERATION COUNT
(1) 013530 010346          MOV      R3,-(SP)      ::SAVE R3
(1) 013532 010446          MOV      R4,-(SP)      ::SAVE R4
(1) 013534 010546          MOV      R5,-(SP)      ::SAVE R5
(1) 013536 113704 013707          MOVB     $OCNT+1, R4    ::GET THE NUMBER OF DIGITS TO TYPE
(1) 013542 005404          NEG      R4
(1) 013544 062704 000006          ADD      #6, R4        ::SUBTRACT IT FOR MAX. ALLOWED
(1) 013550 110437 013706          MOVB     R4, $OCNT     ::SAVE IT FOR USE
(1) 013554 113704 013705          MOVB     $OFILL, R4    ::GET THE ZERO FILL SWITCH
(1) 013560 016605 000012          MOV      12(SP), R5    ::PICKUP THE INPUT NUMBER
(1) 013564 005003          CLR      R3            ::CLEAR THE OUTPUT WORD
(1) 013566 006105          1$:     ROL      R5      ::ROTATE MSB INTO 'C'
(1) 013570 000404          BR       3$
(1) 013572 006105          2$:     ROL      R5      ::FORM THIS DIGIT
(1) 013574 006105          ROL      R5
(1) 013576 006105          ROL      R5
(1) 013600 010503          MOV      R5, R3
(1) 013602 006103          3$:     ROL      R3      ::GET LSB OF THIS DIGIT
(1) 013604 105337 013706          DECB     $OCNT        ::TYPE THIS DIGIT?
(1) 013610 100016          BPL      7$           ::BR IF NO
(1) 013612 042703 177770          BIC      #177770, R3  ::GET RID OF JUNK
(1) 013616 001002          BNE      4$           ::TEST FOR 0
(1) 013620 005704          TST      R4           ::SUPPRESS THIS 0?
(1) 013622 001403          BEQ      5$           ::BR IF YES
(1) 013624 005204          4$:     INC      R4      ::DON'T SUPPRESS ANYMORE 0'S
(1) 013626 052703 000060          BIS      #'0, R3      ::MAKE THIS DIGIT ASCII
(1) 013632 052703 000040          5$:     BIS      #' , R3  ::MAKE ASCII IF NOT ALREADY
(1) 013636 110337 013702          MOVB     R3, 8$       ::SAVE FOR TYPING
(1) 013642 104401 013702          TYPE     , 8$         ::GO TYPE THIS DIGIT
(1) 013646 105337 013704          7$:     DECB     $OCNT    ::COUNT BY 1
(1) 013652 003347          BGT      2$           ::BR IF MORE TO DO
(1) 013654 002402          BLT      6$           ::BR IF DONE
(1) 013656 005204          INC      R4           ::INSURE LAST DIGIT ISN'T A BLANK
(1) 013660 000744          BR       2$           ::GO DO THE LAST DIGIT
(1) 013662 012605          6$:     MOV      (SP)+, R5  ::RESTORE R5
(1) 013664 012604          MOV      (SP)+, R4    ::RESTORE R4
(1) 013666 012603          MOV      (SP)+, R3    ::RESTORE R3

```

```
(1) 013670 016666 000002 000004      MOV     2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
(1) 013676 012616      MOV     (SP)+,(SP)
(1) 013700 000002      RTI                       ;;RETURN
(1) 013702 000        8$:      .BYTE  0                ;;STORAGE FOR ASCII DIGIT
(1) 013703 000        .BYTE  0                ;;TERMINATOR FOR TYPE ROUTINE
(1) 013704 000        $OCNT: .BYTE  0                ;;OCTAL DIGIT COUNTER
(1) 013705 000        $OFILL: .BYTE  0                ;;ZERO FILL SWITCH
(1) 013706 000000     $OMODE: .WORD  0                ;;NUMBER OF DIGITS TO TYPE
1435 .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
(1)
(1)      $TYPDS:
(3) 013710 010046      MOV     R0,-(SP)         ;;PUSH R0 ON STACK
(3) 013712 010146      MOV     R1,-(SP)         ;;PUSH R1 ON STACK
(3) 013714 010246      MOV     R2,-(SP)         ;;PUSH R2 ON STACK
(3) 013716 010346      MOV     R3,-(SP)         ;;PUSH R3 ON STACK
(3) 013720 010546      MOV     R5,-(SP)         ;;PUSH R5 ON STACK
(1) 013722 012746 020200      MOV     #20200,-(SP)     ;;SET BLANK SWITCH AND SIGN
(1) 013726 016605 000020      MOV     20(SP),R5        ;;GET THE INPUT NUMBER
(1) 013732 100004      BPL     1$               ;;BR IF INPUT IS POS.
(1) 013734 005405      NEG     R5               ;;MAKE THE BINARY NUMBER POS.
(1) 013736 112766 000055 000001  1$:      MOV     #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
(1) 013744 005000      CLR     R0               ;;ZERO THE CONSTANTS INDEX
(1) 013746 012703 014124      MOV     $#DBLK,R3        ;;SETUP THE OUTPUT POINTER
(1) 013752 112723 000040      MOV     #' ,(R3)+        ;;SET THE FIRST CHARACTER TO A BLANK
(1) 013756 005002      CLR     R2               ;;CLEAR THE BCD NUMBER
(1) 013760 016001 014114      MOV     $DTBL(R0),R1     ;;GET THE CONSTANT
(1) 013764 160105      SUB     R1,R5            ;;FORM THIS BCD DIGIT
(1) 013766 002402      BLT    4$               ;;BR IF DONE
(1) 013770 005202      INC     R2               ;;INCREASE THE BCD DIGIT BY 1
(1) 013772 000774      BR     3$
(1) 013774 060105      4$:      ADD     R1,R5            ;;ADD BACK THE CONSTANT
(1) 013776 005702      TST    R2               ;;CHECK IF BCD DIGIT=0
(1) 014000 001002      BNE    5$               ;;FALL THROUGH IF 0
(1) 014002 105716      TSTB   (SP)             ;;STILL DOING LEADING 0'S?
(1) 014004 100407      BMI    7$               ;;BR IF YES
(1) 014006 106316      5$:      ASLB   (SP)             ;;MSD?
(1) 014010 103003      BCC    6$               ;;BR IF NO
(1) 014012 116663 000001 177777  6$:      MOV     1(SP),-1(R3)     ;;YES--SET THE SIGN
(1) 014020 052702 000060      BIS    #'0,R2           ;;MAKE THE BCD DIGIT ASCII
(1) 014024 052702 000040      7$:      BIS    #' ,R2           ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 014030 110223      MOV     R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 014032 005720      TST    (R0)+            ;;JUST INCREMENTING
(1) 014034 020027 000010      CMP    R0,#10           ;;CHECK THE TABLE INDEX
(1) 014040 002746      BLT    2$               ;;GO DO THE NEXT DIGIT
(1) 014042 003002      BGT    8$               ;;GO TO EXIT
(1) 014044 010502      MOV     R5,R2           ;;GET THE LSD
```



```

(1) 014254 000000 HALT ::HALT ON ERROR!
(1) 014256 104407 CKSWR ::TEST FOR CHANGE IN SOFT-SWR
(1) 014260 032777 001000 165652 3$: BIT #BIT09,@SWR ::LOOP ON ERROR SWITCH SET?
(1) 014266 001402 BEQ 4$ ::BR IF NO
(1) 014270 013716 002110 MOV $LPERR,(SP) ::FUDGE RETURN FOR LOOPING
(1) 014274 005737 002162 4$: TST $ESCAPE ::CHECK FOR AN ESCAPE ADDRESS
(1) 014300 001402 BEQ 5$ ::BR IF NONE
(1) 014302 013716 002162 MOV $ESCAPE,(SP) ::FUDGE RETURN ADDRESS FOR ESCAPE
(1) 014306 5$:
(1) 014306 000002 RTI ::RETURN
1437 *****
1438 ::GO TYPE ERROR
1439 ::GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
1440 *****
1441 014310 113737 002102 002432 SWRCK: MOV $TSTNM,TSTNUM ::SET UP TEST # ON ER
1442 014316 004737 014326 JSR PC,$ERRTYP ::GO TYPE ERROR
1443 014322 104407 CKSWR ::GO LOOK FOR SWR CHANGE
1444 014324 000207 RTS PC ::RETURN TO ERROR HANDLER
1445 .SBTTL ERROR MESSAGE TIMEOUT ROUTINE
(1) *****
(2) ::*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1) ::*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1) ::*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1) $ERRTYP:
(1) 014326 TYPE $SCRLF ::"CARRIAGE RETURN" & "LINE FEED"
(1) 014326 104401 002165 MOV R0,-(SP) ::SAVE R0
(1) 014332 010046 CLR R0 ::PICKUP THE ITEM INDEX
(1) 014334 005000 BISB @/$ITEMB,R0
(1) 014336 153700 002114 BNE 1$ ::IF ITEM NUMBER IS ZERO, JUST
(1) 014342 001004 MOV SERRPC,-(SP) ::TYPE THE PC OF THE ERROR
(2) 014344 013746 002116 ::SAVE SERRPC FOR TIMEOUT
(2) ::ERROR ADDRESS
(2) 014350 104402 TYP0C ::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 014352 000426 B: 6$ ::GET OUT
(1) 014354 005300 1$: DEC R0 ::ADJUST THE INDEX SO THAT IT WILL
(1) 014356 006300 ASL R0 :: WORK FOR THE ERROR TABLE
(1) 014360 006300 ASL R0
(1) 014362 006300 ASL R0
(1) 014364 062700 002252 ADD #$ERRTB,R0 ::FORM TABLE POINTER
(1) 014370 012037 014400 MOV (R0)+,2$ ::PICKUP "ERROR MESSAGE" POINTER
(1) 014374 001404 BEQ 3$ ::SKIP TIMEOUT IF NO POINTER
(1) 014376 104401 TYPE ::TYPE THE "ERROR MESSAGE"
(1) 014400 000000 2$: .WORD 0 ::"ERROR MESSAGE" POINTER GOES HERE
(1) 014402 104401 002165 TYPE $SCRLF ::"CARRIAGE RETURN" & "LINE FEED"
(1) 014406 012037 014416 3$: MOV (R0)+,4$ ::PICKUP "DATA HEADER" POINTER
(1) 014412 001404 BEQ 5$ ::SKIP TIMEOUT IF 0
(1) 014414 104401 TYPE ::TYPE THE "DATA HEADER"
(1) 014416 000000 4$: .WORD 0 ::"DATA HEADER" POINTER GOES HERE
(1) 014420 104401 002165 TYPE $SCRLF ::"CARRIAGE RETURN" & "LINE FEED"
(1) 014424 011000 5$: MOV (R0),R0 ::PICKUP "DATA TABLE" POINTER
(1) 014426 001004 BNE 7$ ::GO TYPE THE DATA
(1) 014430 012600 6$: MOV (SP)+,R0 ::RESTORE R0
(1) 014432 104401 002165 TYPE $SCRLF ::"CARRIAGE RETURN" & "LINE FEED"
(1) 014436 000207 RTS PC ::RETURN

```



```
(1) 014632 005737 002176          TST      $PASS          ::IF FIRST PASS OF PROGRAM
(1) 014636 001406                   BEQ      1$              ::      INHIBIT ITERATIONS
(1) 014640 005237 002104          INC      $ICNT          ::      INCREMENT ITERATION COUNT
(1) 014644 023737 002160 002104  CMP      $TIMES,$ICNT   ::      CHECK THE NUMBER OF ITERATIONS MADE
(1) 014652 002024                   BGE      $OVER          ::      BR IF MORE ITERATION REQUIRED
(1) 014654 012737 000001 002104 1$:  MOV      #1,$ICNT      ::      REINITIALIZE THE ITERATION COUNTER
(1) 014662 013737 014740 002160  MOV      $MXCNT,$TIMES ::      SET NUMBER OF ITERATIONS TO DO
(1) 014670 105237 002102          $SVLAD: INCB      $TSTNM  ::      COUNT TEST NUMBERS
(1) 014674 113737 002102 002174  MOVB     $TSTNM,$TESTN  ::      SET TEST NUMBER IN APT MAILBOX
(1) 014702 011637 002106          MOV      (SP),$LPADR   ::      SAVE SCOPE LOOP ADDRESS
(1) 014706 011637 002110          MOV      (SP),$LPERR   ::      SAVE ERROR LOOP ADDRESS
(1) 014712 005037 002162          CLR      $ESCAPE      ::      CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 014716 112737 000001 002115  MOVB     #1,$ERMAX     ::      ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 014724 013777 002102 165210 $OVER:  MOV      $TSTNM,@DISPLAY ::      DISPLAY TEST NUMBER
(1) 014732 013716 002106          MOV      $LPADR,(SP)  ::      FUDGE RETURN ADDRESS
(1) 014736 000002                   RTI                      ::      FIXES PS
(1) 014740 000001          $MXCNT: 1.             ::      MAX. NUMBER OF ITERATIONS
1447 .SBTTL  TTY INPUT ROUTINE
(1)
(2)  ::*****
(1)  .ENABL  LSB
(1)
(2)  ::*****
(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) 014742 022737 000176 002140 $CKSWR: CMP      #SWREG,SWR  ::      IS THE SOFT-SWR SELECTED?
(1) 014750 001074                   BNE      15$            ::      BRANCH IF NO
(1) 014752 105777 165166          TSTB     @STKS         ::      CHAR THERE?
(1) 014756 100071                   BPL      15$            ::      IF NO, DON'T WAIT AROUND
(1) 014760 117746 165102          MOVB     @STKB,-(SP)   ::      SAVE THE CHAR
(1) 014764 042716 177600          BIC      #^C177,(SP)  ::      STRIP-OFF THE ASCII
(1) 014770 022726 000007          CMP      #7,(SP)+     ::      IS IT A CONTROL G?
(1) 014774 001062                   BNE      15$            ::      NO, RETURN TO USER
(1) 014776 123727 002134 000001  CMPB     $AUTOB,#1    ::      ARE WE RUNNING IN AUTO-MODE?
(1) 015004 001456                   BEQ      15$            ::      BRANCH IF YES
(1)
(1) 015006 104401 015477          $GTSWR: TYPE     ,SCNTLG  ::      ECHO THE CONTROL-G (^G)
(1) 015012 104401 015504          TYPE     ,SMSWR       ::      TYPE CURRENT CONTENTS
(2) 015016 013746 000176          MOV      SWREG,-(SP)  ::      SAVE SWREG FOR TYPEOUT
(2) 015022 104402                   TYPOC    ::      GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 015024 104401 015515          TYPE     ,SMNEW       ::      PROMPT FOR NEW SWR
(1) 015030 005046          19$:  CLR      -(SP)     ::      CLEAR COUNTER
(1) 015032 005046          CLR      -(SP)     ::      THE NEW SWR
(1) 015034 105777 165104          7$:  TSTB     @STKS         ::      CHAR THERE?
(1) 015040 100375                   BPL      7$            ::      IF NOT TRY AGAIN
(1)
(1) 015042 117746 165100          MOVB     @STKB,-(SP)  ::      PICK UP CHAR
(1) 015046 042716 177600          BIC      #^C177,(SP)  ::      MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1) 015052 021627 000025          9$:  CMP      (SP),#25   ::      IS IT A CONTROL-U?
(1) 015056 001005                   BNE      10$           ::      BRANCH IF NOT
(1) 015060 104401 015472          TYPE     ,SCNTLU     ::      YES, ECHO CONTROL-U (^U)
```

```
(1) 015064 062706 000006 20$: ADD #6,SP      ::IGNORE PREVIOUS INPUT
(1) 015070 000757                                ::LET'S TRY IT AGAIN
(1)
(1)
(1) 015072 021627 000015 10$: CMP (SP),#15    ::IS IT A <CR>?
(1) 015076 001022                                ::BRANCH IF NO
(1) 015100 005766 000004 TST 4(SP)      ::YES, IS IT THE FIRST CHAR?
(1) 015104 001403 BEQ 11$      ::BRANCH IF YES
(1) 015106 016677 000002 165024 MOV 2(SP),@SWR  ::SAVE NEW SWR
(1) 015114 062706 000006 11$: ADD #6,SP      ::CLEAR UP STACK
(1) 015120 104401 002165 14$: TYPE ,SCRLF    ::ECHO <CR> AND <LF>
(1) 015124 123727 002135 000001 CMPB $INTAG,#1  ::RE-ENABLE TTY KBD INTERRUPTS?
(1) 015132 001003 BNE 15$      ::BRANCH IF NOT
(1) 015134 012777 000100 165002 MOV #100,@$TKS  ::RE-ENABLE TTY KBD INTERRUPTS
(1) 015142 000002 15$: RTI      ::RETURN
(1) 015144 004737 013072 16$: JSR PC,$TYPEC  ::ECHO CHAR
(1) 015150 021627 000060 CMP (SP),#60    ::CHAR < 0?
(1) 015154 002420 BLT 18$      ::BRANCH IF YES
(1) 015156 021627 000067 CMP (SP),#67    ::CHAR > 7?
(1) 015162 003015 BGT 18$      ::BRANCH IF YES
(1) 015164 042726 000060 DIC #60,(SP)+  ::STRIP-OFF ASCII
(1) 015170 005766 000002 TST 2(SP)      ::IS THIS THE FIRST CHAR
(1) 015174 001403 BEQ 17$      ::BRANCH IF YES
(1) 015176 006316 ASL (SP)      ::NO, SHIFT PRESENT
(1) 015200 006316 ASL (SP)      :: CHAR OVER TO MAKE
(1) 015202 006316 ASL (SP)      :: ROOM FOR NEW ONE.
(1) 015204 005266 000002 17$: INC 2(SP)      ::KEEP COUNT OF CHAR
(1) 015210 056616 177776 BIS -2(SP), (SP) ::SET IN NEW CHAR
(1) 015214 000707 BR 7$      ::GET THE NEXT ONE
(1) 015216 104401 002164 18$: TYPE ,SQUES  ::TYPE ?<CR><LF>
(1) 015222 000720 BR 20$      ::SIMULATE CONTROL-U
(1) .DSABL LSB
```

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

::\*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

::\*CALL:

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

```
(1) 015224 011646 $RDCHR: MOV (SP),-(SP)  ::PUSH DOWN THE PC
(1) 015226 016666 000004 000002 MOV 4(SP),2(SP) ::SAVE THE PS
(1) 015234 105777 164704 1$: TSTB @$TKS  ::WAIT FOR
(1) 015240 100375 BPL 1$      ::A CHARACTER
(1) 015242 117766 164700 000004 MOVB @$TKB,4(SP) ::READ THE TTY
(1) 015250 042766 177600 000004 BIC #^C<177>,4(SP) ::GET RID OF JUNK IF ANY
(1) 015256 026627 000004 000023 CMP 4(SP),#23  ::IS IT A CONTROL-S?
(1) 015264 001013 BNE 3$      ::BRANCH IF NO
(1) 015266 105777 164652 2$: TSTB @$TKS  ::WAIT FOR A CHARACTER
(1) 015272 100375 BPL 2$      ::LOOP UNTIL ITS THERE
(1) 015274 117746 MOVB @$TKB,-(SP) ::GET CHARACTER
(1) 015300 042716 BIC #^C177,(SP) ::MAKE IT 7-BIT ASCII
(1) 015304 022627 000021 CMP (SP)+,#21  ::IS IT A CONTROL-Q?
(1) 015310 001366 BNE 2$      ::IF NOT DISCARD IT
```



```

(1) 015312 000750          BR      1$          ::YES, RESUME
(1) 015314 026627 000004 000021 3$:    CMP      4(SP),#$XON  ::IS IT A RANDOM XON?          :RAN001
(1) 015322 001744          BEQ      1$          ::BRANCH IF YES                  :RAN001
(1) 015324 026627 000004 000140          CMP      4(SP),#140  ::IS IT UPPER CASE?
(1) 015332 002407          BLT      4$          ::BRANCH IF YES
(1) 015334 026627 000004 000175          CMP      4(SP),#175  ::IS IT A SPECIAL CHAR?
(1) 015342 003003          BGT      4$          ::BRANCH IF YES
(1) 015344 042766 000040 000004          BIC      #40,4(SP)  ::MAKE IT UPPER CASE
(1) 015352 000002          4$:    RTI          ::GO BACK TO USER
(2)
(1) *****
(1) *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1) *CALL:
(1) *      RDLIN          ::INPUT A STRING FROM THE TTY
(1) *      RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1) *
(1) *      *              ::TERMINATOR WILL BE A BYTE OF ALL 0'S
(1) 015354 010346          SRDLIN: MOV      R3,-(SP)  ::SAVE R3
(1) 015356 012703 015462          1$:    MOV      #$TTYIN,R3  ::GET ADDRESS
(1) 015362 022703 015472          2$:    CMP      #$TTYIN+8.,R3  ::BUFFER FULL?
(1) 015366 101405          BLOS     4$          ::BR IF YES
(1) 015370 104410          RDCHR          ::GO READ ONE CHARACTER FROM THE TTY
(1) 015372 112613          MOV      (SP)+,(R3)  ::GET CHARACTER
(1) 015374 122713 000177          10$:   CMP      #177,(R3)  ::IS IT A RUBOUT
(1) 015400 001003          BNE     3$          ::SKIP IF NOT
(1) 015402 104401 002164          4$:    TYPE     $QUES  ::TYPE A '?'
(1) 015406 000763          BR      1$          ::CLEAR THE BUFFER AND LOOP
(1) 015410 111337 015460          3$:    MOV      (R3),9$  ::ECHO THE CHARACTER
(1) 015414 104401 015460          TYPE     9$
(1) 015420 122723 000015          CMP      #15,(R3)+  ::CHECK FOR RETURN
(1) 015424 001356          BNE     2$          ::LOOP IF NOT RETURN
(1) 015426 105063 177777          CLRB    -1(R3)     ::CLEAR RETURN (THE 15)
(1) 015432 104401 002166          TYPE     $LF       ::TYPE A LINE FEED
(1) 015436 012603          MOV      (SP)+,R3  ::RESTORE R3
(1) 015440 011646          MOV      (SP),-(SP) ::ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 015442 016666 000004 000002          MOV      4(SP),2(SP) ::FIRST ASCII CHARACTER ON IT
(1) 015450 012766 015462 000004          MOV      #$TTYIN,4(SP)
(1) 015456 000002          RTI          ::RETURN
(1) 015460 000          9$:    .BYTE    0          ::STORAGE FOR ASCII CHAR. TO TYPE
(1) 015461 000          .BYTE    0          ::TERMINATOR
(1) 015462 000010          $TTYIN: .BLKB    8.   ::RESERVE 8 BYTES FOR TTY INPUT
(1) 015472 052536 005015 000          $CNTLU: .ASCIZ  /^U/<15><12> ::CONTROL 'U'
(1) 015477 136 006507 000012          $CNTLG: .ASCIZ  /^G/<15><12> ::CONTROL 'G'
(1) 015504 005015 053523 020122          $MSWR:  .ASCIZ  <15><12>/SWR = /
(1) 015515 040 047040 053505          $MNEW:  .ASCIZ  / NEW = /
1448 .SBTTL  POWER DOWN AND UP ROUTINES
(1)
(2) *****
(1) *POWER DOWN ROUTINE
(1) 015526 012737 015672 000024          $PWDRN: MOV      #SILLUP,#PWVVEC ::SET FOR FAST UP
(1) 015534 012737 000340 000026          MOV      #340,#PWVVEC+2 ::PRIO:7
(3) 015542 010046          MOV      R0,-(SP)  ::PUSH R0 ON STACK
(3) 015544 010146          MOV      R1,-(SP)  ::PUSH R1 ON STACK
(3) 015546 010246          MOV      R2,-(SP)  ::PUSH R2 ON STACK
(3) 015550 010346          MOV      R3,-(SP)  ::PUSH R3 ON STACK
(3) 015552 010446          MOV      R4,-(SP)  ::PUSH R4 ON STACK
(3) 015554 010546          MOV      R5,-(SP)  ::PUSH R5 ON STACK

```

```
(3) 015556 017746 164356          MOV   @SWR,-(SP)      ::PUSH @SWR ON STACK
(1) 015562 010637 015676          MOV   SP,$SAVR6     ::SAVE SP
(1) 015566 012737 015600 000024   MOV   #SPWRUP,@#PWRVEC ::SET UP VECTOR
(1) 015574 000000                   HALT
(1) 015576 000776                   BR    -2             ::HANG UP
(1)
(2) ::*****
(1) ::POWER UP ROUTINE
(1) 015600 012737 015672 000024   $PWRUP: MOV   #SILLUP,@#PWRVEC ::SET FOR FAST DOWN
(1) 015606 013706 015676          MOV   $SAVR6,SP     ::GET SP
(1) 015612 005037 015676          CLR   $SAVR6        ::WAIT LOOP FOR THE TTY
(1) 015616 005237 015676          1$:  INC   $SAVR6     ::WAIT FOR THE INC
(1) 015622 001375                   BNE   1$            ::OF WORD
(3) 015624 012677 164310          MOV   (SP)+,@SWR    ::POP STACK INTO @SWR
(3) 015630 012605                   MOV   (SP)+,R5     ::POP STACK INTO R5
(3) 015632 012604                   MOV   (SP)+,R4     ::POP STACK INTO R4
(3) 015634 012603                   MOV   (SP)+,R3     ::POP STACK INTO R3
(3) 015636 012602                   MOV   (SP)+,R2     ::POP STACK INTO R2
(3) 015640 012601                   MOV   (SP)+,R1     ::POP STACK INTO R1
(3) 015642 012600                   MOV   (SP)+,R0     ::POP STACK INTO R0
(1) 015644 012737 015526 000024   MOV   #SPWRDN,@#PWRVEC ::SET UP THE POWER DOWN VECTOR
(1) 015652 012737 000340 000026   MOV   #340,@#PWRVEC+2 ::PRIO:7
(1) 015660 104401                   TYPE
(1) 015662 015700          SPWRMG: .WORD  PWRMSG      ::REPORT THE POWER FAILURE
(1) 015664 012716          MOV   (PC)+,(SP)     ::POWER FAIL MESSAGE POINTER
(1) 015666 003134          SPWRAD: .WORD  START1   ::RESTART AT START1
(1) 015670 000002                   RTI
(1) 015672 000000          SILLUP: HALT        ::THE POWER UP SEQUENCE WAS STARTED
(1) 015674 000776                   BR    -2             :: BEFORE THE POWER DOWN WAS COMPLETE
(1) 015676 000000          $SAVR6: 0           ::PUT THE SP HERE
1449 015700 005015 042522 052123 PWRMSG: .ASCIZ <15><12>/RESTARTED FROM PWR FAIL/
1450 .EVEN
1451 .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)
(1) 015732 010046          STRAP: MOV   R0,-(SP)   ::SAVE R0
(1) 015734 016600 000002          MOV   2(SP),R0      ::GET TRAP ADDRESS
(1) 015740 005740          TST   -(R0)         ::BACKUP BY 2
(1) 015742 111000          MOVB  (R0),R0       ::GET RIGHT BYTE OF TRAP
(1) 015744 006300          ASL   R0            ::POSITION FOR INDEXING
(1) 015746 016000 015766          MOV   $TRPAD(R0),R0 ::INDEX TO TABLE
(1) 015752 000200          RTS   R0            ::GO TO ROUTINE
(1)
(1)
(1) ::THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1) 015754 011646          STRAP2: MOV   (SP)-,(SP) ::MOVE THE PC DOWN
(1) 015756 016666 000004 000002   MOV   4(SP),2(SP)   ::MOVE THE PSW DOWN
(1) 015764 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) :  
(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) 016002 015012 \$GTSWR ::CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING  
(1)  
(3) 016004 014742 \$CKSWR ::CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR  
(3) 016006 015224 \$RDCHR ::CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE  
(3) 016010 015354 \$RDLIN ::CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE

1452  
1453 .SBTTL ASCII MESSAGES  
1454 : :GPA TITLED: .ASCIZ <15><12>/CVDRDA DRV11J DIAG TEST PART 2/<15><12>  
1455 016012 041600 042126 042122 TITLED: .ASCIZ <200>/CVDRDB DRV11J DIAG TEST PART 2/<200> ::GPA  
1456 : :GPA TLCABL: .ASCIZ <15><12>/DRV11J CABLE REQ'D/<15><12>  
1457 016054 042200 053122 030461 TLCABL: .ASCIZ <200>/DRV11J CABLE REQ'D/<200> ::GPA  
1458  
1459 016101 122 043505 052040 EM1: .ASCIZ /REG TIMEOUT ER/  
1460 016120 042522 020107 042522 EM2: .ASCIZ 'REG READ/WRITE ER'  
1461 016142 051111 020122 042522 EM3: .ASCIZ /IRR REG ER/  
1462 016155 101 051103 051040 EM4: .ASCIZ /ACR REG ER/  
1463 016170 046511 020122 042522 EM5: .ASCIZ /IMR REG ER/  
1464 016203 111 051123 051040 EM6: .ASCIZ /ISR REG ER/  
1465 016216 046111 042514 040507 EM7: .ASCIZ /ILLEGAL VECTOR MEM ADDR ER/  
1466 016251 103 044510 020120 EM10: .ASCIZ /CHIP STAT ER/  
1467 : :GPA EM11: .ASCIZ /ILLEGAL INTERRUPT RECEIVED/  
1468 016266 046111 042514 040507 EM11: .ASCIZ /ILLEGAL INTERRUPT REC'D/ ::GPA  
1469 016316 047111 042524 051122 EM12: .ASCIZ /INTERRUPT TEST ERROR/  
1470 : :GPA EM13: .ASCIZ /MULTIPLE INTERRUPTS RECEIVED/  
1471 016343 115 046125 044524 EM13: .ASCIZ /MULTIPLE INTERRUPTS REC'D/ ::GPA  
1472 016375 126 041505 020124 EM14: .ASCIZ /VECT ADDR MEM ER/  
1473 016416 051105 050122 020103 DH1: .ASCIZ /ERRPC TSTNUM BUSADR EXPCT RCVD/  
1474 016463 105 051122 041520 DH2: .ASCIZ /ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD/  
1475 016550 051105 050122 020103 DH3: .ASCIZ /ERRPC TSTNUM BUSADR VAM ADDR/  
1476 016615 105 051122 041520 DH4: .ASCIZ /ERRPC TSTNUM TESTPC BUSADR VAM ADDR/  
1477  
1478 .EVEN  
1479 016666 002116 002432 002122 DT1: \$ERRPC,TSTNUM,\$BDADR,\$GDDAT,\$BDDAT,0  
1480 016702 002116 002432 002122 DT2: \$ERRPC,TSTNUM,\$BDADR,VECVAL,VECLOC,\$GDDAT,\$BDDAT,0  
1481 016722 002116 002432 002122 DT3: \$ERRPC,TSTNUM,\$BDADR,VECVAL,VECLOC,0  
1482 016736 002116 002432 002120 DT4: \$ERRPC,TSTNUM,\$GDADR,\$BDADR,VECVAL,VECLOC,0  
1483  
1484  
1485 : :\*\*\*\*\*  
1486 : :DBUF' IS THE STORAGE AREA FOR SYSMAC LOCATIONS 0-202  
1487 : :PRESERVED IN ORDER TO TEST DRV11J VECTORING TO  
1488 : :VECTOR SPACE 0-200.  
1489 : :\*\*\*\*\*

CVDRDB DRV11J DIAG TST PRT2  
CVDRDB.P11 10-AUG-81 10:52

MACY11 30G(1063) 10-AUG-81 11:00 <sup>N 5</sup> PAGE 2-36  
ASCII MESSAGES

SEQ 0059

1490 016754 000102  
1491  
1492  
1493  
1494  
1495  
1496 017160 000100  
1497 017360

DBUF: .BLKW 66. ;1ST ADRS OF DATA STORAGE AREA

::\*\*\*\*\*  
:VECTOR BUFFER USED FOR VECTOR UNIQUENESS TEST  
:FOR 64 INTERRUPT VECTORS.  
:\*\*\*\*\*  
VBUF: .BLKW 64.  
ENDBUF:

1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515 017360 005227 177777  
1516 017364 001002  
1517 017366 004737 000400  
1518 017372 005727  
1519 017374 000000  
1520 017376 000207  
1521  
1522 017400  
1523 000400  
1524 000400 005037 017374  
1525 000404 013746 000004  
1526 000410 012737 000504 000004  
1527 000416 012700 160010  
1528 000422 005720  
1529 000424 000240  
1530 000426 020027 174000  
1531 000432 103773  
1532 000434 010037 017374  
1533 000440 012700 000040  
1534 000444 040037 000006  
1535 000450 040037 000016  
1536 000454 040037 000022  
1537 000460 040037 000032  
1538 000464 040037 000036  
1539 000470 012737 170000 000140  
1540 000476 012637 000004  
1541 000502 000207  
1542  
1543 000504 012716 000512  
1544 000510 000002  
1545 000512 012637 000004  
1546 000516 012700 000402  
1547 000522 013701 000376  
1548 000526 010602  
1549 000530 012704 000570  
1550 000534 014446  
1551 000536 020427 000546  
1552 000542 101374  
1553 000544 010607

```

.SBTTL FALCON (KXT-11) UPGRADE ROUTINES.                :::GPA
:
: THE FOLLOWING ROUTINES HAVE BEEN ADDED TO ALLOW DIAGNOSTIC(S)
: TO RUN ON A FALCON (KXT-11) BASED SYSTEM.
: TO DETERMINE WHETHER WE'RE A FALCON OR NOT, WE'LL SIZE THE 1ST 3/4 OF
: THE I/O PAGE (28K TO 31K). FALCON HAS 2KW LOCAL RAM AT 28K(+4) TO 30K
: AND A MACRO-ODT AT 30K TO 31K. CONSEQUENTLY, ALL I/O DEVICES MUST
: BE PLACED BETWEEN 174000 AND 177776. ADDITIONALLY, WE'LL STRAP THE
: EMT AND TRAP SERVICE LEVEL TO PRI6, AND SET THE HALT VECTOR SO THAT
: WE CAN STOP THE SUCKER !!
:
: TO MINIMIZE THE IMPACT OF THESE CHANGES ON FINAL PROGRAM SIZE, THE
: BULK OF THIS CODE IS PLACED IN THE FLOATING VECTOR SPACE (400-776).
: IF THE CPU AT HAND IS A FALCON (KXT11), IT STAYS THERE (NO HARM DONE).
: OTHERWISE, THE AREA IS RESTORED TO ITS ORIGINAL 'TRAP-CATCHER' STATE.
:
FALCON: INC      #-1          ; ONCE-ONLY !!!                :::GPA
        BNE      1$          ;                               :::GPA
        CALL     KXTCHK       ; EXECUTE FALCON CHECK          :::GPA
1$:     TST      (PC)+        ; TEST FALCON FLAG...    :::GPA
KXTFLAG: 0          ; ...NZ = FALCON...                :::GPA
        RETURN              ; ...AND RETURN TO CALLER...    :::GPA
:
        $$VPC= .           ;                               :::GPA
        = 400              ; RESTORE FROM 374:376 AT END  :::GPA
KXTCHK: CLR      KXTFLAG     ; ASSUME NOT FALCON.      :::GPA
        MOV      @#4, -(SP)   ; SAVE ERROR VECTOR.    :::GPA
        MOV      #2$, @#4     ; SET A TRAP CATCHER.   :::GPA
        MOV      #160010, R0  ; FALCON RAVI STARTS AT 28K+4. :::GPA
1$:     TST      (R0)+        ;                               :::GPA
        240                ;                               :::GPA
        CMP      R0, #174000  ; SIZE TO 31K.          :::GPA
        BLO      1$          ;                               :::GPA
        MOV      R0, KXTFLAG  ; MUST BE FALCON, SET THE FLAG :::GPA
        MOV      #40, R0      ; GET PRI1 BIT...      :::GPA
        BIC      R0, @#6      ; ...AND LOWER BUS-ERROR... :::GPA
        BIC      R0, @#16     ; ...BPT...            :::GPA
        BIC      R0, @#22     ; ...IOT...            :::GPA
        BIC      R0, @#32     ; ...EMT...            :::GPA
        BIC      R0, @#36     ; ...AND TRAP SERVICE TO PRI6 :::GPA
        MOV      #170000, @#140 ; ENABLE 'BREAK' HALT.  :::GPA
        MOV      (SP)+, @#4   ; RESTORE ERROR VECTOR... :::GPA
        RETURN              ; ...AND RETURN.                :::GPA
:
2$:     MOV      #3$, (SP)    ; TRAP -- NOT A FALCON... :::GPA
        RTI                          ; ...CONTINUE.          :::GPA
3$:     MOV      (SP)+, @#4   ; RESET ERROR VECTOR    :::GPA
        MOV      #402, R0     ; SET-UP TO RESTORE FLOATING... :::GPA
        MOV      @#376, R1    ; ...VECTORS (400 - 776). :::GPA
        MOV      SP, R2      ; SAVE STACK POINTER IN R2  :::GPA
        MOV      #6$, R4     ;                               :::GPA
4$:     MOV      -(R4), -(SP) ; PUSH THE RESTORE CODE... :::GPA
        CMP      R4, #5$     ; ...ONTO THE STACK.    :::GPA
        BHI      4$          ;                               :::GPA
        MOV      SP, PC      ; AND EXECUTE IT.        :::GPA

```

```
1555  
1556  
1557  
1558 000546 010060 177776  
1559 000552 010110  
1560 000554 022020  
1561 000556 020027 000776  
1562 000562 101771  
1563 000564 010206  
1564 000566 000207  
1565 000570  
1566  
1567  
1568  
1569  
1570  
1571 000104  
1575  
1576 017400  
1577 017400  
1578 000001
```

```
::: THIS CODE IS RELOCATED TO AND EXECUTED IN THE STACK AREA.  
5$:  MOV R0,-2(R0) : RESTORE +2... : :GPA  
     MOV R1,(R0) : ...HALT (OR IOT). : :GPA  
     CMP (R0)+,(R0)+ : :GPA  
     CMP R0,#776 : :GPA  
     BLOS 5$ : LOOP 'TIL DONE : :GPA  
     MOV R2,SP : THEN RESTORE SP... : :GPA  
     RETURN : ...AND RETURN TO CALLER : :GPA  
6$:  
::: IF FALCON, THIS AREA IS FREE FOR ANY PROGRAM UNIQUE  
::: CHANGES OR DATA STRUCTURES.  
::: IF USED, YOU'D BETTER PROTECT IT !!!  
$FREE= <1000-.>/2 : FREE WORDS LEFT. : :GPA  
LASTAD= .=SSVPC : :GPA  
        : :GPA  
        .END
```

ABASE = 164160  
ACDW1 = 000000  
ACDW2 = 000000  
ACPUOP = 000000  
ACRLOC = 002450  
ADDW0 = 000000  
ADDW1 = 000000  
ADDW10 = 000000  
ADDW11 = 000000  
ADDW12 = 000000  
ADDW13 = 000000  
ADDW14 = 000000  
ADDW15 = 000000  
ADDW2 = 000000  
ADDW3 = 000000  
ADDW4 = 000000  
ADDW5 = 000000  
ADDW6 = 000000  
ADDW7 = 000000  
ADDW8 = 000000  
ADDW9 = 000000  
ADEVCT = 000000  
ADEVN = 000001  
AENV = 000000  
AENVN = 000000  
AFATAL = 000000  
AMADR1 = 000000  
AMADR2 = 000000  
AMADR3 = 000000  
AMADR4 = 000000  
AMAMS1 = 000000  
AMAMS2 = 000000  
AMAMS3 = 000000  
AMAMS4 = 000000  
AMSGAD = 000000  
AMSGLG = 000000  
AMSGTY = 000000  
AMTYP1 = 000000  
AMTYP2 = 000000  
AMTYP3 = 000000  
AMTYP4 = 000000  
APASS = 000000  
APRIOR = 000000  
APTC SU = 000040  
APTENV = 000001  
APTSIZ = 000200  
APTSPO = 000100  
APTVEC = 011614  
APT200 = 012066  
ASMREG = 000000  
ATESTN = 000000  
AUNIT = 000000  
AUSWR = 000000

AVECT1 = 000000  
AVECT2 = 000000  
BDSAV = 002464  
BGCHP3 = 012636  
BIT0 = 000001  
BIT00 = 000001  
BIT01 = 000002  
BIT02 = 000004  
BIT03 = 000010  
BIT04 = 000020  
BIT05 = 000040  
BIT06 = 000100  
BIT07 = 000200  
BIT08 = 000400  
BIT09 = 001000  
BIT1 = 000002  
BIT10 = 002000  
BIT11 = 004000  
BIT12 = 010000  
BIT13 = 020000  
BIT14 = 040000  
BIT15 = 100000  
BIT2 = 000004  
BIT3 = 000010  
BIT4 = 000020  
BIT5 = 000040  
BIT6 = 000100  
BIT7 = 000200  
BIT8 = 000400  
BIT9 = 001000  
BPTVEC = 000014  
BYCNT = 002470  
BYNUM = 002472  
CAT200 = 012024  
CHPISR = 000140  
CIMR = 000040  
CIRMR = 000020  
CIRR = 000100  
CISR = 000160  
CKSWR = 104407  
CLRBF = 011334  
CLRC SR = 011014  
CLRIRR = 011074  
CR = 000015  
CRLF = 000200  
CSIMR = 000050  
CSIRMR = 000030  
CSIRR = 000110  
CSISR = 000170  
DBUF = 016754  
DISP = 177570  
DH1 = 016416  
DH2 = 016463

DH3 = 016550  
DH4 = 016615  
DIR = 000400  
DISPLA = 002142  
DISPRE = 000174  
DMAP = 002434  
DRCSA = 002412  
DRCSB = 002416  
DRCS C = 002422  
DRCS D = 002426  
DRDBA = 002414  
DRDBB = 002420  
DRDBC = 002424  
DRDBD = 002430  
DSWR = 177570  
DT1 = 016666  
DT2 = 016702  
DT3 = 016722  
DT4 = 016736  
EDCHP3 = 012656  
EMTVEC = 000030  
EM1 = 016101  
EM10 = 016251  
EM11 = 016266  
EM12 = 016316  
EM13 = 016343  
EM14 = 016375  
EM2 = 016120  
EM3 = 016142  
EM4 = 016155  
EM5 = 016170  
EM6 = 016203  
EM7 = 016216  
ENDBUF = 017360  
ENDTRP = 011636  
END200 = 012106  
ERRVEC = 000004  
FALCON = 017360  
FRMBUF = 011374  
GRPCNT = 002462  
GTSWR = 104406  
HT = 000011  
IE = 001000  
IMRLOC = 002442  
INTBY4 = 011700  
INTFLG = 002436  
INTSR1 = 011644  
INTSR3 = 012124  
INTO = 012236  
INT1 = 012242  
INT10 = 012306  
INT11 = 012312  
INT12 = 012316

INT13 = 012322  
INT14 = 012326  
INT15 = 012332  
INT16 = 012336  
INT17 = 012342  
INT18 = 012346  
INT19 = 012352  
INT2 = 012246  
INT20 = 012356  
INT21 = 012362  
INT22 = 012366  
INT23 = 012372  
INT24 = 012376  
INT25 = 012402  
INT26 = 012406  
INT27 = 012412  
INT28 = 012416  
INT29 = 012422  
INT3 = 012252  
INT30 = 012426  
INT31 = 012432  
INT32 = 012436  
INT33 = 012442  
INT34 = 012446  
INT35 = 012452  
INT36 = 012456  
INT37 = 012462  
INT38 = 012466  
INT39 = 012472  
INT4 = 012256  
INT40 = 012476  
INT41 = 012502  
INT42 = 012506  
INT43 = 012512  
INT44 = 012516  
INT45 = 012522  
INT46 = 012526  
INT47 = 012532  
INT48 = 012536  
INT49 = 012542  
INT5 = 012262  
INT50 = 012546  
INT51 = 012552  
INT52 = 012556  
INT53 = 012562  
INT54 = 012566  
INT55 = 012572  
INT56 = 012576  
INT57 = 012602  
INT58 = 012606  
INT59 = 012612  
INT6 = 012266  
INT60 = 012616

INT61 = 012622  
INT62 = 012626  
INT63 = 012632  
INT7 = 012272  
INT8 = 012276  
INT9 = 012302  
IOTVEC = 000020  
IRRLOC = 002446  
ISRLOC = 002444  
KXTCHK = 000400  
KXTFLA = 017374  
LASTAD = 017400  
LF = 000012  
LMD04 = 000200  
LMD57 = 000240  
LVLCNT = 002466  
LVLSAV = 002474  
MACR = 000254  
MIMR = 000244  
MIRR = 000250  
MISR = 000240  
NEXPAS = 003200  
NEXPA1 = 003204  
NEXPA2 = 003250  
NXDEV = 010632  
NXDEV1 = 010636  
PACR = 000300  
PIMR = 000260  
PIRQ = 177772  
PIRQVE = 000240  
PRO = 000000  
PR1 = 000040  
PR2 = 000100  
PR3 = 000140  
PR4 = 000200  
PR5 = 000240  
PR6 = 000300  
PR7 = 000340  
PS = 177776  
PSW = 177776  
PVMA = 000340  
PWRMSG = 015700  
PWRVEC = 000024  
RDCHR = 104410  
RDLIN = 104411  
RDY = 100000  
RESTRP = 011432  
RESVEC = 000010  
RES200 = 011762  
R6 = 000006  
R7 = 000007  
SETVEC = 012210  
SIMR = 000060

SIRR = 000120	TOBUF = 011354	SCPUOP = 002216	SLPADR = 002106	\$STUP = 177777
SSIMR = 000070	TPVEC = 000064	SCRFLF = 002165	SLPERR = 002110	\$SVLAD = 014670
SSIRR = 000130	TRAPVE = 000034	SDBLK = 014124	SMADR1 = 002222	\$SVPC = 017400
STACK = 001100	TRPALL = 011752	SDEVCT = 002200	SMADR2 = 002226	\$SWR = 165400
START = 002476	TRPCAT = 011524	SDEVM = 002246	SMADR3 = 002232	\$SWREG = 002212
START1 = 003134	TRPOUT = 011734	SDOAGN = 010770	SMADR4 = 002236	\$SWRPMK = 000000
STKLMT = 177774	TRP200 = 012170	SDTBL = 014114	SMAIL = 002170	\$STESTN = 002174
STRVEC = 011300	TRTVEC = 000014	SENDAD = 010760	SMAMS1 = 002220	\$TIMES = 002160
SWR = 002140	TSTNUM = 002432	SENDCT = 010726	SMAMS2 = 002224	\$TKB = 002146
SWRCK = 014310	TST1 = 003254	SENDMG = 010777	SMAMS3 = 002230	\$TKS = 002144
SWREG = 000176	TST2 = 004122	SENULL = 010774	SMAMS4 = 002234	\$TN = 000007
SWRSAV = 002460	TST3 = 004622	SENV = 002210	SMBADR = 002002	\$TPB = 002152
SWO = 000001	TST4 = 005536	SEVM = 002211	SMFLG = 013456	\$TPFLG = 002157
SWO0 = 000001	TST5 = 006530	SEOP = 010672	SMNEW = 015515	\$TPS = 002150
SWO1 = 000002	TST6 = 007476	SEOPCT = 010720	SMSGAD = 002204	\$STRAP = 015732
SWO2 = 000004	TYPDS = 104405	SERFLG = 002103	SMSGLG = 002206	\$STRAP2 = 015754
SWO3 = 000010	TYPE = 104401	SERMAX = 002115	SMSGTY = 002170	\$STRP = 000012
SWO4 = 000020	TYPOC = 104402	SERROR = 014134	SMSWR = 015504	\$STRPAD = 015766
SWO5 = 000040	TYPON = 104404	SERRPC = 002116	SMTYP1 = 002221	\$STSTM = 002004
SWO6 = 000100	TYPOS = 104403	SERRTB = 002252	SMTYP2 = 002225	\$STSTM = 002102
SWO7 = 000200	VBUF = 017160	SERTY = 014326	SMTYP3 = 002231	\$STTYIN = 015462
SWO8 = 000400	VECFIL = 011132	SERTTL = 002112	SMTYP4 = 002235	\$STYPDS = 013710
SWO9 = 001000	VECLOC = 002452	SESCAP = 002162	SMXCNT = 014740	\$STYPE = 012660
SW1 = 000002	VECPAT = 002454	SETABL = 002210	SNULL = 002154	\$STYPEC = 013072
SW10 = 002000	VECVAL = 002456	SETEND = 002252	SNWTST = 000001	\$STYPEX = 013?12
SW11 = 004000	XXDP = 002440	SFATAL = 002172	SOCNT = 013704	\$STYPOC = 013506
SW12 = 010000	SAPTHD = 002000	SFFLG = 013460	SOMODE = 013706	\$STYPON = 013522
SW13 = 020000	SATYC = 013240	SFILLC = 002156	SOVER = 014724	\$STYPOS = 013462
SW14 = 040000	SATY1 = 013214	SFILLS = 002155	SPASS = 002176	\$SUNIT = 002202
SW15 = 100000	SATY3 = 013222	SFREE = 000104	SPASTM = 002006	\$SUNITM = 002010
SW2 = 000004	SATY4 = 013232	\$GDADR = 002120	SPWRAD = 015666	\$SUSWR = 002214
SW3 = 000010	SAUTOB = 002134	\$GDDAT = 002124	SPWRDN = 015526	\$SVECT1 = 002240
SW4 = 000020	SBASE = 002244	\$GET42 = 010750	SPWRMG = 015662	\$SVECT2 = 002242
SW5 = 000040	SBDADR = 002122	\$GTSWR = 015012	SPWRUP = 015600	\$XOFF = 000023
SW6 = 000100	SBDDAT = 002126	\$HD = 000003	SQUES = 002164	\$XON = 000021
SW7 = 000200	SCDW1 = 002250	\$HIBTS = 002000	SRDCHR = 015224	\$XTSTR = 014474
SW8 = 000400	SCHARC = 013210	\$ICNT = 002104	SRDLIN = 015354	\$SGET4 = 000000
SW9 = 001000	SCKSWR = 014742	\$ILLUP = 015672	\$RDSZ = 000010	\$OFILL = 013705
TBITVE = 000014	SCMTAG = 002100	\$INTAG = 002135	\$RTNAD = 010772	= 017400
TITLED = 016012	SCM3 = 000000	\$ITEMB = 002114	\$SAVR6 = 015676	.\$X = 002000
TKVEC = 000060	SCNTLG = 015477	\$LF = 002166	\$SCOPE = 014462	
TLCABL = 016054	SCNTLU = 015472	\$LFLG = 013457	\$SETUP = 000117	

. ABS. 017400 000 CON RW ABS LCL I

ERRORS DETECTED: 0

CVDRDB, CVDRDB=CVDRDB  
RUN-TIME: 19 7 .4 SECONDS  
RUN-TIME RATIO: 76/27=2.8  
CORE USED: 25K (49 PAGES)