

DRV11J

DRV11J DIAG TST PRT 2
CVDRDA0

AH-F761A-MC

COPYRIGHT 1980
FICHE 1 OF 1

JAN 1980

digital
MADE IN USA

The image shows a grid of 12 columns and 12 rows of small, illegible data tables or charts, likely diagnostic test results. Each cell in the grid contains a small table with multiple columns and rows of text, which is too small to read. The overall layout is a dense grid of these small tables.

IDENTIFICATION

PRODUCT CODE: AC-F759A-MC .
PRODUCT NAME: CVDRDAO DRV11J DIAG TST PRT2
DATE CREATED: OCTOBER 1979
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 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.

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 BINARY FORMATTED PAPERTAPES OR XXDP MEDIA (FILES WITH .BIC OR .BIN EXTENSIONS ONLY).

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 CVDRDA 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.

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.

5.2 CONTROL

* 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. IRK 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 XSDP/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.

9.0 PROGRAM TEST DESCRIPTIONS

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

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

18	OPERATIONAL SWITCH SETTINGS
28	BASIC DEFINITIONS
176	TRAP CATCHER
185	STARTING ADDRESS(ES)
190	ACT11 HOOKS
204	APT PARAMETER BLOCK
226	COMMON TAGS
269	APT MAILBOX-ETABLE
318	ERROR POINTER TABLE
439	PROGRAM START
441	INITIALIZE THE COMMON TAGS
534	T1 TEST CHIP INTERRUPT ,GROUP 1,GROUP 2
649	T2 TEST ACR/ISR INTERRUPT ,GROUP 1,GROUP 2
741	T3 TEST VECTOR ADDR MEM,LEV 0-7,BY0-3,(204-1774),GRPS 1,2
865	T4 TEST VECTOR ADDR MEM,LVLS 0-7,BY0-3,(0-200)GRPS 1,2
1008	T5 TEST VECTOR ADDR UNIQUENESS IN FIXED MODE FOR GRPS 1,2
1139	T6 TEST VECTOR ADDR UNIQUENESS IN ROTATING MODE FOR GRPS 1,2
1312	END OF PASS ROUTINE
1350	PROGRAM SUBROUTINES
1732	PATTERNS FOR REGISTER R/W
1750	SYSMAC ROUTINES
1752	TYPE ROUTINE
1831	APT COMMUNICATIONS ROUTINE
1888	BINARY TO OCTAL (ASCII) AND TYPE
1965	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2032	ERROR HANDLER ROUTINE
2087	ERROR MESSAGE TYPEOUT ROUTINE
2134	SCOPE HANDLER ROUTINE
2199	TTY INPUT ROUTINE
2338	POWER DOWN AND UP ROUTINES
2386	TRAP DECODER
2409	TRAP TABLE
2429	ASCII MESSAGES

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
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
50
51
52
53
54
55
56

000001
160000
165400
000001

001100

000011
000012
000015
000200
177776

177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

```
.TITLE CVDRDA DRV11J DIAG TST PRT2
.*COPYRIGHT (C) 1979
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY BILL HEAVEY
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
$TN=1
$SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
$SWR=165400
$TN=1
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.*      SWITCH          USE
.*      -----
.*      15             HALT ON ERROR
.*      14             LOOP ON TEST
.*      13             INHIBIT ERROR TYP0UTS
.*      11             INHIBIT ITERATIONS
.*      9              LOOP ON ERROR
.*      8              LOOP ON TEST IN SWR<7:0>
.SBTTL BASIC DEFINITIONS
.*INITIAL ADDRESS OF THE STACK POINTER ***1100***
STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
.*MISCELLANEOUS DEFINITIONS
HT= 11               ;;CODE FOR HORIZONTAL TAB
LF= 12               ;;CODE FOR LINE FEED
CR= 15               ;;CODE FOR CARRIAGE RETURN
CRLF= 200            ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776           ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774       ;;STACK LIMIT REGISTER
PIRQ= 177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570         ;;HARDWARE SWITCH REGISTER
DDISP= 177570        ;;HARDWARE DISPLAY REGISTER
.*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0               ;;GENERAL REGISTER
R1= %1               ;;GENERAL REGISTER
R2= %2               ;;GENERAL REGISTER
R3= %3               ;;GENERAL REGISTER
R4= %4               ;;GENERAL REGISTER
R5= %5               ;;GENERAL REGISTER
R6= %6               ;;GENERAL REGISTER
R7= %7               ;;GENERAL REGISTER
SP= %6               ;;STACK POINTER
PC= %7               ;;PROGRAM COUNTER
.*PRIORITY LEVEL DEFINITIONS
```

BASIC DEFINITIONS

57	000000	PR0=	0	::PRIORITY LEVEL	0
58	000040	PR1=	40	::PRIORITY LEVEL	1
59	000100	PR2=	100	::PRIORITY LEVEL	2
60	000140	PR3=	140	::PRIORITY LEVEL	3
61	000200	PR4=	200	::PRIORITY LEVEL	4
62	000240	PR5=	240	::PRIORITY LEVEL	5
63	000300	PR6=	300	::PRIORITY LEVEL	6
64	000340	PR7=	340	::PRIORITY LEVEL	7

.*'SWITCH REGISTER' SWITCH DEFINITIONS

65					
66		SW15=	100000		
67	100000	SW14=	40000		
68	040000	SW13=	20000		
69	020000	SW12=	10000		
70	010000	SW11=	4000		
71	004000	SW10=	2000		
72	002000	SW09=	1000		
73	001000	SW08=	400		
74	000400	SW07=	200		
75	000200	SW06=	100		
76	000100	SW05=	40		
77	000040	SW04=	20		
78	000020	SW03=	10		
79	000010	SW02=	4		
80	000004	SW01=	2		
81	000002	SW00=	1		
82	000001	.EQUIV	SW09,SW9		
83		.EQUIV	SW08,SW8		
84		.EQUIV	SW07,SW7		
85		.EQUIV	SW06,SW6		
86		.EQUIV	SW05,SW5		
87		.EQUIV	SW04,SW4		
88		.EQUIV	SW03,SW3		
89		.EQUIV	SW02,SW2		
90		.EQUIV	SW01,SW1		
91		.EQUIV	SW00,SW0		
92					
93					

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

94					
95	100000	BIT15=	100000		
96	040000	BIT14=	40000		
97	020000	BIT13=	20000		
98	010000	BIT12=	10000		
99	004000	BIT11=	4000		
100	002000	BIT10=	2000		
101	001000	BIT09=	1000		
102	000400	BIT08=	400		
103	000200	BIT07=	200		
104	000100	BIT06=	100		
105	000040	BIT05=	40		
106	000020	BIT04=	20		
107	000010	BIT03=	10		
108	000004	BIT02=	4		
109	000002	BIT01=	2		
110	000001	BIT00=	1		
111		.EQUIV	BIT09,BIT9		
112		.EQUIV	BIT08,BIT8		

```
113 .EQUIV BIT07,BIT7
114 .EQUIV BIT06,BIT6
115 .EQUIV BIT05,BIT5
116 .EQUIV BIT04,BIT4
117 .EQUIV BIT03,BIT3
118 .EQUIV BIT02,BIT2
119 .EQUIV BIT01,BIT1
120 .EQUIV BIT00,BIT0
121
122 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
123 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
124 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
125 000014 TBITVEC=14 ;:'T' BIT
126 000014 TRTVEC= 14 ;:TRACE TRAP
127 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
128 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
129 000024 PWRVEC= 24 ;:POWER FAIL
130 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
131 000034 TRAPVEC=34 ;:'TRAP' TRAP
132 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
133 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
134 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
135 164160 ABASE= 164160 ;:BASE ADDRESS
136 000001 ADEVM= 1 ;:DEFAULT TO ONE DRV'1J
137 100000 RDY= BIT15
138 000400 DIR= BIT8
139 001000 IE= BIT9
140
141 ;CHIP COMMAND SUMMARY
142 000020 CIRMR= 20 ;:CLEAR IRR AND IMR
143 000030 CSIRMR= 30 ;:CLEAR SINGLE IRR AND IMR BIT
144
145 000040 CIMR= 40 ;:CLEAR IMR
146 000050 CSIMR= 50 ;:CLEAR SINGLE IMR BIT
147 000060 SIMR= 60 ;:SET ALL IMR BITS
148 000070 SSIMR= 70 ;:SET SINGLE IMR BITS
149
150 000100 CIRR= 100 ;:CLEAR IRR
151 000110 CSIRR= 110 ;:CLEAR SINGLE IRR BITS
152 000120 SIRR= 120 ;:SET ALL IRR BITS
153 000130 SSIRR= 130 ;:SET SINGLE IRR BITS
154
155 000140 CHPISR= 140 ;:CLEAR HIGHEST PRIORITY ISR BIT
156 000160 CISR= 160 ;:CLEAR ISR
157 000170 CSISR= 170 ;:CLEAR SINGLE ISR BIT
158
159 000200 LMD04= 200 ;:LOAD MODE BITS M0-M4
160 000240 LMD57= 240 ;:LOAD MODE BITS M5-M7
161
162 ;CHIP MODE BIT PRESELECTION
163 000240 MISR= 240
164 000244 MIMR= 244
165 000250 MIRR= 250
166 000254 MACR= 254
167
168 ;CHIP WRITE PRESELECTION
```

```

169          000300          PACR= 300          ;PRESELECT AUTO CLEAR REG. FOR WRITING
170          000260          PIMR= 260          ;PRESELECT IMR REG. FOR WRITING
171          000340          PVMA= 340          ;PRESELECT VECTOR MEMORY ADDRESS
172
173          .SBTTL TRAP CATCHER
174
175          000000          .=0
176          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
177          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
178          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
179          000 74          .=174
180 000174 000000          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
181 000176 000000          SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER
182          .SBTTL STARTING ADDRESS(ES)
183 000200 000137 000 76          .WORD @START ;;JUMP TO STARTING ADDRESS OF PROGRAM
184          000100          .-100
185 000100 000104 000200 000002          .WORD 104,200,2          ;IF 'B EVENT' ON Q BUS IS CONNECTED
186          ;IGNORE IT'S INTERRUPT - JUST DO A RTI

```

187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222

000106
000046 010614
000052 000052
000052 000000
000106
002000

002000
000024 000200
000044 000044
000044 002000
002000

002000 000000
002002 002170
002004 000014
002006 000036
002010 000036
002012 000031

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

.SBTTL APT PARAMETER BLOCK
:*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
:*****
      .SX=.           ;;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:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:  .WORD 12.    ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 30.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 30.    ;;ADDITIONAL RUN TIME (SEC$) OF A PASS FOR EACH ADDITIONAL UNIT
          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```



```
223 .SBTTL COMMON TAGS
224
225 ::*****
226 ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
227 ::*USED IN THE PROGRAM.
228
229 002100 .=2100
230 002100 $CMTAG: ::START OF COMMON TAGS
231 002100 000000 .WORD 0
232 002102 000 $STNM: .BYTE 0 ::CONTAINS THE TEST NUMBER
233 002103 000 $ERFLG: .BYTE 0 ::CONTAINS ERROR FLAG
234 002104 000000 $ICNT: .WORD 0 ::CONTAINS SUBTEST ITERATION COUNT
235 002106 000000 $LPADR: .WORD 0 ::CONTAINS SCOPE LOOP ADDRESS
236 002110 000000 $LPERR: .WORD 0 ::CONTAINS SCOPE RETURN FOR ERRORS
237 002112 000000 $ERTTL: .WORD 0 ::CONTAINS TOTAL ERRORS DETECTED
238 002114 000 $ITEMB: .BYTE 0 ::CONTAINS ITEM CONTROL BYTE
239 002115 001 $ERMAX: .BYTE 1 ::CONTAINS MAX. ERRORS PER TEST
240 002116 000000 $ERRPC: .WORD 0 ::CONTAINS PC OF LAST ERROR INSTRUCTION
241 002120 000000 $GDADR: .WORD 0 ::CONTAINS ADDRESS OF 'GOOD' DATA
242 002122 000000 $BDADR: .WORD 0 ::CONTAINS ADDRESS OF 'BAD' DATA
243 002124 000000 $GDDAT: .WORD 0 ::CONTAINS 'GOOD' DATA
244 002126 000000 $BDDAT: .WORD 0 ::CONTAINS 'BAD' DATA
245 002130 000000 .WORD 0 ::RESERVED--NOT TO BE USED
246 002132 000000 .WORD 0
247 002134 000 $AUTOB: .BYTE 0 ::AUTOMATIC MODE INDICATOR
248 002135 000 $INTAG: .BYTE 0 ::INTERRUPT MODE INDICATOR
249 002136 000000 .WORD 0
250 002140 177570 SWR: .WORD DSWR ::ADDRESS OF SWITCH REGISTER
251 002142 177570 DISPLAY: .WORD DDISP ::ADDRESS OF DISPLAY REGISTER
252 002144 177560 $TKS: 177560 ::TTY KBD STATUS
253 002146 177562 $TKB: 177562 ::TTY KBD BUFFER
254 002150 177564 $TPS: 177564 ::TTY PRINTER STATUS REG. ADDRESS
255 002152 177566 $TPB: 177566 ::TTY PRINTER BUFFER REG. ADDRESS
256 002154 000 $NULL: .BYTE 0 ::CONTAINS NULL CHARACTER FOR FILLS
257 002155 002 $FILLS: .BYTE 2 ::CONTAINS # OF FILLER CHARACTERS REQUIRED
258 002156 012 $FILLC: .BYTE 12 ::INSERT FILL CHARS. AFTER A 'LINE FEED'
259 002157 000 $TPFLG: .BYTE 0 ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
260 002160 000000 $TIMES: 0 ::MAX. NUMBER OF ITERATIONS
261 002162 000000 $ESCAPE: 0 ::ESCAPE ON ERROR ADDRESS
262 002164 077 $QUES: .ASCII /?/ ::QUESTION MARK
263 002165 015 $CRLF: .ASCII <15> ::CARRIAGE RETURN
264 002166 000012 $LF: .ASCIIZ <12> ::LINE FEED
265 ::*****
266 .SBTTL APT MAILBOX-ETABLE
267
268 ::*****
269 .EVEN
270 002170 $MAIL: ::APT MAILBOX
271 002170 000000 $MSGTY: .WORD MSGTY ::MESSAGE TYPE CODE
272 002172 000000 $FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
273 002174 000000 $TESTN: .WORD ATESTN ::TEST NUMBER
274 002176 000000 $PASS: .WORD APASS ::PASS COUNT
275 002200 000000 $DEVCT: .WORD ADEVCT ::DEVICE COUNT
276 002202 000000 $UNIT: .WORD AUNIT ::I/O UNIT NUMBER
277 002204 000000 $MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
278 002206 000000 $MSGLG: .WORD AMSLG ::MESSAGE LENGTH
```

279	002210		SETABLE:		::: APT ENVIRONMENT TABLE
280	002210	000	\$ENV:	.BYTE AENV	::: ENVIRONMENT BYTE
281	002211	000	\$ENVM:	.BYTE AENVM	::: ENVIRONMENT MODE BITS
282	002212	000000	\$SWREG:	.WORD ASWREG	::: APT SWITCH REGISTER
283	002214	000000	\$USWR:	.WORD AUSWR	::: USER SWITCHES
284	002216	000000	\$CPUOP:	.WORD ACPUOP	::: CPU TYPE, OPTIONS
285					BITS 15-11=CPU TYPE
286					11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
287					11/70=06, PDQ=07, 0/10
288					BIT 10=REAL TIME CLOCK
289					BIT 9=FLOATING POINT PROCESSOR
290					BIT 8=MEMORY MANAGEMENT
291	002220	000	\$MAMS1:	.BYTE AMAMS1	::: HIGH ADDRESS, M.S. BYTE
292	002221	000	\$MTYP1:	.BYTE AMTYP1	::: MEM. TYPE, BLK#1
293					MEM. TYPE BYTE -- (HIGH BYTE)
294					900 NSEC CORE=001
295					300 NSEC BIPOLAR=002
296					500 NSEC MOS=003
297	002222	000000	\$MADR1:	.WORD AMADR1	::: HIGH ADDRESS, BLK#1
298					MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
299	002224	000	\$MAMS2:	.BYTE AMAMS2	::: HIGH ADDRESS, M.S. BYTE
300	002225	000	\$MTYP2:	.BYTE AMTYP2	::: MEM. TYPE, BLK#2
301	002226	000000	\$MADR2:	.WORD AMADR2	::: MEM. LAST ADDRESS, BLK#2
302	002230	000	\$MAMS3:	.BYTE AMAMS3	::: HIGH ADDRESS, M.S. BYTE
303	002231	000	\$MTYP3:	.BYTE AMTYP3	::: MEM. TYPE, BLK#3
304	002232	000000	\$MADR3:	.WORD AMADR3	::: MEM. LAST ADDRESS, BLK#3
305	002234	000	\$MAMS4:	.BYTE AMAMS4	::: HIGH ADDRESS, M.S. BYTE
306	002235	000	\$MTYP4:	.BYTE AMTYP4	::: MEM. TYPE, BLK#4
307	002236	000000	\$MADR4:	.WORD AMADR4	::: MEM. LAST ADDRESS, BLK#4
308	002240	000000	\$VECT1:	.WORD AVECT1	::: INTERRUPT VECTOR#1, BUS PRIORITY#1
309	002242	000000	\$VECT2:	.WORD AVECT2	::: INTERRUPT VECTOR#2, BUS PRIORITY#2
310	002244	164160	\$BASE:	.WORD ABASE	::: BASE ADDRESS OF EQUIPMENT UNDER TEST
311	002246	000001	\$DEVN:	.WORD ADEVN	::: DEVICE MAP
312	002250	000000	\$CDW1:	.WORD ACDW1	::: CONTROLLER DESCRIPTION WORD#1
313	002252		\$TEND:		
314			.NEXT		

315
 316
 317
 318
 319
 320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370

.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 \$ITEMP. 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
 EM1 :REG TIMEOUT ER
 DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
 DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDA:
 0

:ERROR 2

EM2 :REG READ/WRITE ER
 DH2 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
 DT2 :\$ERRPC TSTNUM \$BDADR VECVAL VECLOC \$GDDAT \$BDDAT
 0

:ERROR 3

EM3 :IRR REG ER
 DH2 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
 DT2 :\$ERRPC TSTNUM \$BDADR VECVAL VECLOC \$GDDAT \$BDDAT
 0

:ERROR 4

EM4 :ACR REG ER
 DH2 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
 DT2 :\$ERRPC TSTNUM \$BDADR VECVAL VECLOC \$GDDAT \$BDDAT
 0

:ERROR 5

EM5 :IMR REG ERROR
 DH2 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
 DT2 :\$ERRPC TSTNUM \$BDADR VECVAL VECLOC \$GDDAT \$BDDAT
 0

:ERROR 6

EM6 :ISR REG ERROR
 DH2 :ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD
 DT2 :\$ERRPC TSTNUM \$BDADR VECVAL VECLOC \$GDDAT \$BDDAT
 0

:ERROR 7

EM14 :VECTOR ADDR MEM ER
 DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
 DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT
 0

002252
 002252 016031
 002254 016354
 002256 016624
 002260 000000
 002262 016050
 002264 016421
 002266 016640
 002270 000000
 002272 015072
 002274 016421
 002276 016640
 002300 000000
 002302 016105
 002304 016421
 002306 016640
 002310 000000
 002312 016120
 002314 016421
 002316 016640
 002320 000000
 002322 016133
 002324 016421
 002326 016640
 002330 000000
 002332 016333
 002334 016354
 002336 016624
 002340 000000

371									
372			:ERROR	10					
373	002342	016201		EM10	:CHIP STAT ER				
374	002344	016421		DH2	:ERRPC TSTNUM	BUSADR	VAM	ADDR	EXPCY
375	002346	016640		DT2	:SERRPC TSTNUM	SBDADR	VECVAL	VECLOC	SGDDAT
376	002350	000000		0					SBDDAT
377									
378			:ERROR	11					
379	002352	016216		EM11	:ILLEGAL INTERRUPT RECEIVED				
380	002354	016506		DH3	:ERRPC TSTNUM	BUSADR	VAM	ADDR	
381	002356	016660		DT3	:SERRPC TSTNUM	SBDADR	VECVAL	VECLOC	
382	002360	000000		0					
383									
384			:ERROR	12					
385	002362	016251		EM12	:INTERRUPT TEST ERROR				
386	002364	016421		DH2	:ERRPC TSTNUM	BUSADR	VAM	ADDR	EXPCY
387	002366	016640		DT2	:SERRPC TSTNUM	SBDADR	VECVAL	VECLOC	SGDDAT
388	002370	000000		0					SBDDAT
389									
390			:ERROR	13	:MULTIPLE INTERRUPTS RECEIVED				
391	002372	016276		EM13	:MULTIPLE INTERRUPTS RECEIVED				
392	002374	016421		DH2	:ERRPC TSTNUM	BUSADR	VAM	ADDR	EXPCY
393	002376	016640		DT2	:SERRPC TSTNUM	SBDADR	VECVAL	VECLOC	SGDDAT
394	002400	000000		0					SBDDAT
395									
396			:ERROR	14					
397	002402	016146		EM7	:ILLEGAL VECTOR ADDR MEM ERROR				
398	002404	016553		DH4	:ERRPC TSTNUM	TESTRC	BUSADR	VAM	ADDR
399	002406	016674		DT4	:SERRPC TSTNUM	SGDADR	SBDADR	VECVAL	VECLOC
400	002410	000000		0					
401									
402									
403									
404									
405									
406	002412	164160			DRCSA:	ABASE			
407	002414	164162			DRDBA:	ABASE+2			
408	002416	164164			DRCSB:	ABASE+4			
409	002420	164166			DRDBB:	ABASE+6			
410	002422	164170			DRDSC:	ABASE+10			
411	002424	164172			DRDBC:	ABASE+12			
412	002426	164174			DRDSD:	ABASE+14			
413	002430	164176			DRDBD:	ABASE+16			
414									
415									
416									
417									
418	002432	000000			TSTNUM:	0			:CONTAINS TEST NUMBER ON ERROR
419	002434	000001			DMAP:	1			
420	002436	000000			INTFLG:	.WORD	0		
421	002440	000000			XXDP:	.WORD	0		
422	002442	000000			IMRLOC:	.WORD	0		
423	002444	000000			ISRLOC:	.WORD	0		
424	002446	000000			IRKLOC:	.WORD	0		
425	002450	000000			ACRLOC:	.WORD	0		
426	002452	000000			VECLOC:	.WORD	0		

427	002454	000000	VECPAT:	.WORD	0
428	002456	000000	VECVAL:	.WORD	0
429	002460	000000	SWRSV:	.WORD	0
430	002462	000000	GRPCNT:	.WORD	0
431	002464	000000	BDSAV:	.WORD	0
432	002466	000000	LVL CNT:	.WORD	0
433	002470	000000	BYCNT:	.WORD	0
434	002472	000000	BYNUM:	.WORD	0
435	002474	000000	LVL SAV:	.WORD	0

```
436          .SBTTL PROGRAM START
437 002476 START:
438          .SBTTL INITIALIZE THE COMMON TAGS
439          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
440 002476 012706 002100 MOV    # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
441 002502 005026 CLR      (R6)+          ;;CLEAR MEMORY LOCATION.
442 002504 022706 002140 CMP    # $SWR,R6      ;;DONE?
443 002510 001374 BNE     -6              ;;LOOP BACK IF NO
444 002512 012706 002100 MOV    #2100,SP        ;;SETUP THE STACK POINTER
445          ;;INITIALIZE A FEW VECTORS
446 002516 012737 014416 000020 MOV    # $SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
447 002524 012737 000340 000022 MOV    #340,@IOTVEC+2  ;;LEVEL 7
448 002532 012737 014070 000030 MOV    # $ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
449 002540 012737 000340 000032 MOV    #340,@EMTVEC+2  ;;LEVEL 7
450 002546 012737 015656 000034 MOV    # $TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
451 002554 012737 000340 000036 MOV    #340,@TRAPVEC+2 ;;LEVEL 7
452 002562 012737 015452 000024 MOV    # $PWRDN,@PWRVEC ;;POWER FAILURE VECTOR
453 002570 012737 000340 000026 MOV    #340,@PWRVEC+2  ;;LEVEL 7
454 002576 005037 002160 CLR    $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
455 002602 005037 002162 CLR    $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
456 002606 112737 000001 002115 MOVB  #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
457 002614 012737 002614 002106 MOV    #,$SLPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
458 002622 012737 002622 002110 MOV    #,$SLPERR      ;;SETUP THE ERROR LOOP ADDRESS
459          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
460          ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
461 002630 013746 000004 MOV    @ $ERRVEC,-(SP) ;;SAVE ERROR VECTOR
462 002634 012737 002670 000004 MOV    #64,$@ $ERRVEC  ;;SET UP ERROR VECTOR
463 002642 012737 177570 002140 MOV    # $DSWR,$SWR    ;;SETUP FOR A HARDWARE SWICH REGISTER
464 002650 012737 177570 002142 MOV    # $DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
465 002656 022777 177777 177254 CMP    #-1,@ $SWR     ;;TRY TO REFERENCE HARDWARE SWR
466 002664 001012 BNE     66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
467          ;;AND THE HARDWARE SWR IS NOT -1
468 002666 000403 BR     65$          ;;BRANCH IF NO TIMEOUT
469 002670 012716 002676 64$: MOV    #65$,(SP)      ;;SET UP FOR TRAP RETURN
470 002674 000002 RTI
471 002676 012737 000176 002140 65$: MOV    # $SWREG,$SWR    ;;POINT TO SOFTWARE SWR
472 002704 012737 000174 002142 MOV    # $DISPREG,$DISPLAY
473 002712 012637 000004 66$: MOV    (SP)+,@ $ERRVEC ;;RESTORE ERROR VECTOR
474
475 002716 005037 002176 CLR    $PASS          ;;CLEAR PASS COUNT
476 002722 132737 000200 002211 BITB  # $APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
477 002730 001403 BEQ    67$          ;;YES,USE NON-APT SWITCH
478 002732 012737 002212 002140 MOV    # $SSWREG,$SWR ;;NO,USE APT SWITCH REGISTER
479 002740 67$:
480 002740 005037 002172 CLR    $FATAL         ;;CLEAR ERROR NUMBER
481 002744 005037 002170 CLR    $MSGTYP        ;;CLEAR MESSAGE TYPE
482 002750 005037 002174 CLR    $TFSTN        ;;CLEAR TEST NUMBER
483          ;CHECK OPERATING ENVIRONMENT
484 002754 005737 000042 TST    @ #42          ;;ARE WE IN ACT/XXDP AUTO MODE?
485 002760 001410 BEQ    1$           ;;BRANCH IF NO
486 002762 023737 000042 000046 CMP    @ #42,@ #46    ;;IS IT ACT AUTO MODE?
487 002770 001410 BEQ    2$           ;;BRANCH IF YES
488 002772 012737 177777 002440 MOV    #-1,$XDP      ;;SET XXDP CHAIN MODE INDICATOR
489 003000 000404 BR     2$
490 003002 123727 002210 000001 1$. CMPB  $ENV,#1        ;;ARE WE IN APT AUTO MODE?
491 003010 001003 BNE     3$           ;;BRANCH IF NO
```

```

492 003012 112737 000001 002134 2S:   MOVB   #1,SAUTOB      ;SET AUTO MODE INDICATOR
493
494
495 003020 005227 177777          3S:   .PRINT TITLE IF NOT IN ACT OR APT AUTO MODE
496 003024 001014          :INC   #-1           ;FIRST TIME?
497 003026 004537 011164          :BNE   5S            ;SKIP TITLE IF NO
498 003032 005737 002134          :JSR   R5,TOBUF     ;STORE 0-202 INTO BUFFER AREA
499 003036 001403          :TST   $AUTOB       ;ARE WE IN AUTO MODE?
500 003040 005737 002440          :BEQ   4S            ;BRANCH TO TITLE TYPEOUT IF NOT
501 003044 001404          :TST   XXDP         ;IS THE AUTO MODE UNDER XXDP?
502 003046 104401 015736          :BEQ   5S            ;SKIP TITLE IF NOT
503 003052 104401 016002          4S:   TYPE   ,TITLED   ;PRINT OUT THE TITLE
504
505
506 003056 005737 002134          :TYPE  ,TLCABL      ;PRINT DRV11J CABLE REQ'D
507
508
509 003062 001001          :GET THE VALUE IN THE SOFTWARE SWITCH REGISTER
510 003064 104406          5S:   TST   $AUTOB   ;ARE WE IN AUTOMATIC MODE?
511 003066 005037 002202          :BNE   START1       ;BRANCH IF YES
512 003072 013737 002246 002434   :GTSWR
513 003100 042737 177760 002434   START1: CLR   $UNIT   ;ASK FOR SWR INPUT FROM CONSOLE
514 003106 013701 002244          :MOV   $DEVN,DMAP   ;CLEAR UNIT NUMBER
515 003112 010137 002412          :BIC   #177760,DMAP ;UP TO 4 DRV11J'S
516 003116 032737 000001 002434   :MOV   $BASE,R1     ;POSITION OF DRV11J'S
517 003124 001002          :MOV   R1,DRCSA     ;GET BASE ADDRESS
518 003126 000137 010472          :BIT   #1,DMAP      ;MAKE BUS REG.POINTER = BASE
519 003132 012700 002412          :BNE   NEXPAS       ;IS FIRST DRV11-J SELECTED
520 003144 022700 002432          :JMP   NXDEV1       ;YES
521 003150 001372          :MOV   #DRCSA,R0    ;NO,ADVANCE BASE DRV11J ADDRESS
522 003152 004537 011334          NEXPAS: MOV   R1,(R0)+ ;SET UP REGISTER ADDRESS POINTERS
523 003156 012706 002100          NEXPA1: MOV   #2,R1   ;LOAD EM
524 003162 013737 002202 002200   :ADD   #DRDBD+2,R0  ;ALL DONE?
525 003170 123727 002210 000001   :CMP   NEXPA1       ;BR IF NOT
526 003176 001401          :JSR   R5,TRPCAT    ;EXTEND TRAP CATCHER 204-1774
527 003200 000005          :MOV   #2100,SP     ;ALWAYS RESET STACK
528 003202 106427 000340          :MOV   $UNIT,$DEVCT ;LOAD APT COUNTER WITH UNIT #
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```
530
531
532
533
534 003206 000004
535 003210 013700 002412
536 003214 013701 002416
537 003220 013702 002412
538 003224 012737 003276 002110
539 003232 012737 000002 002462
540 003240 004537 010650 120$:
541 003244 012703 012644
542 003250 012737 000050 002442
543 003256 004537 010730
544 003262 012737 011454 000300
545 003270 012737 000340 000302
546 003276 005037 002436 111$:
547 003302 012703 002100
548 003306 112710 000340
549 003312 112710 000060
550 003316 010037 002122
551 003322 112710 000060
552 003326 113710 002442
553 003332 112710 000202
554 003336 106427 000000
555 003342 112712 000241
556
557
558 003346 112710 000241
559 003352 005737 002436
560 003356 001401
561 003360 104011
562 003362 005037 002436 1$:
563 003366 112710 000120
564 003372 106427 000340
565 003376 005737 002436
566 003402 001401
567 003404 104011
568 003406 005037 002436 2$:
569 003412 106427 000000
570 003416 152762 000002 000001
571 003424 012737 000001 002124
572 003432 013737 002436 002126
573 003440 023737 002124 002126
574 003446 001401
575 003450 104012
576 003452 106427 000340 3$:
577 003456 012737 101710 002124
578 003464 010237 002122
579 003470 011237 002126
580 003474 042737 000007 002126
581 003502 023737 002124 002126
582 003510 001401
583 003512 104002
584 003514 010137 002122 4$:
585 003520 005037 002126
```

: *TEST 1 TFST 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\$, \$LPERR ;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, LVO
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
MOVB #241, (R0)
TST INTFLG
BEQ 1\$
ERROR 11 ;ERROR, ILLEGAL INTERRUPT
1\$: CLR INTFLG ;RESET INT. COUNTER
MOVB #SIARR, (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

586	003524	005037	002124	CLR	\$GDDAT	:SET UP FOR BYTE EXPECTED
587	003530	111337	002124	MOVB	(R3), \$GDDAT	:EXPECTED DATA
588	003534	112710	000240	MOVB	#MISR, (R0)	:LOAD MODE TO READ ISR
589	003540	111137	002126	MOVB	(R1), \$BDDAT	
590	003544	023737	002124	CMP	\$GDDAT, \$BDDAT	
591	003552	001401		BEQ	5\$	
592	003554	104006		ERROR	6	:ISR ERROR
593	003556	116337	000001	MOVB	1(R3), \$GDDAT	:STORE EXPECTED
594	003564	112710	000244	MOVB	#MIMR, (R0)	:LOAD MODE FOR IMR
595	003570	111137	002126	MOVB	(R1), \$BDDAT	
596	003574	023737	002124	CMP	\$GDDAT, \$BDDAT	
597	003602	001401		BEQ	6\$	
598	003604	104005		ERROR	5	:IMR ERROR
599	003606	116337	000001	MOVB	1(R3), \$GDDAT	:EXPECTED IRR
600	003614	112710	000250	MOVB	#MIRR, (R0)	:LOAD MODE BITS FOR IRR
601	003620	111137	002126	MOVB	(R1), \$BDDAT	
602	003624	023737	002124	CMP	\$GDDAT, \$BDDAT	
603	003632	001401		BEQ	7\$	
604	003634	104003		ERROR	3	:IRR ERROR
605	003636	005037	002124	CLR	\$GDDAT	
606	003642	112710	000254	MOVB	#MACR, (R0)	
607	003646	111137	002126	MOVB	(R1), \$BDDAT	
608	003652	023737	002124	CMP	\$GDDAT, \$BDDAT	
609	003660	001401		BEQ	10\$	
610	003662	104004		ERROR	4	:ACR ERROR
611	003664	005037	002124	CLR	\$GDDAT	
612	003670	112710	000140	MOVB	#CHIPISR, (R0)	:CLEAR HIGHEST PRIOR ISR
613	003674	112710	000240	MOVB	#MISR, (R0)	:LOAD MODE TO READ ISR
614	003700	111137	002126	MOVB	(R1), \$BDDAT	
615	003704	023737	002124	CMP	\$GDDAT, \$BDDAT	
616	003712	001401		BEQ	11\$	
617	003714	104006		ERROR	6	:ISR REG ERROR
618	003716	012737	101710	MOV	#101710, \$GDDAT	:RDY, DIR, I/E, CHIP - 31X
619	003724	010237	002122	MOV	R2, \$BDDADR	:CSRA ADDRESS
620	003730	011237	002126	MOV	(R2), \$BDDAT	
621	003734	042737	000007	BIC	#7, \$BDDAT	:CLEAR UNDEFINED BITS
622	003742	023737	002124	CMP	\$GDDAT, \$BDDAT	
623	003750	001401		BEQ	12\$	
624	003752	104002		ERROR	2	:CSRA REG ERROR
625	003754	012737	100710	MOV	#100710, \$GDDAT	:RDY, DIR, CHIP - 31X
626	003762	042712	001000	BIC	#BIT9, (R2)	:CLEAR I/E
627	003766	011237	002126	MOV	(R2), \$BDDAT	:READ CSRA
628	003772	042737	000007	BIC	#7, \$BDDAT	:CLEAR UNDEFINED BITS
629	004000	023737	002124	CMP	\$GDDAT, \$BDDAT	
630	004006	001401		BEQ	13\$	
631	004010	104002		ERROR	2	:CSRA REG ERROR
632	004012	005723		TST	(R3)+	:GET NEXT ISR, IMR EXPECTED RESULTS
633	004014	020327	012664	CMP	R3, #EDCHP3	:CHECK FOR END
634	004020	001404		BEQ	14\$	
635	004022	005237	002442	INC	IMRLOC	:INCREMENT MASK BIT TO CLEAR
636	004026	000137	003276	JMP	111\$:TEST NEXT ISR BIT
637	004032	005337	002462	DEC	GRPCNT	:TESTED BOTH GROUPS?
638	004036	001406		BEQ	TST2	:BR IF DONE
639	004040	013700	002422	MOV	DRCSC, R0	:SETUP FOR GROUP 2
640	004044	013701	002426	MOV	DRCSD, R1	:CSRC = CONTROL GROUP 2
641						:CSRSD = DATA GROUP 2

642 004050 000137 003240
 643
 644
 645
 646
 647
 648 004054 000004
 649 004056 013700 002412
 650 004062 013701 002416
 651 004066 013702 002412
 652 004072 012737 004144 002110
 653 004100 012737 000002 002462
 654 004106 004537 010650 120\$:
 655 004112 012703 012644
 656 004116 012737 000050 002442
 657 004124 004537 010730
 658 004130 012737 011454 000300
 659 004136 012737 000340 000302
 660 004144 005037 002436 111\$:
 661 004150 012706 002100
 662 004154 042712 001000
 663 004160 112712 000242
 664 004164 112710 000242
 665 004170 112710 000300
 666 004174 111311
 667 004176 112710 000340
 668 004202 112711 000060
 669 004206 010037 002122
 670 004212 112710 000060
 671 004216 113710 002442
 672 004222 112710 000202
 673 004226 106427 000000
 674 004232 152762 000002 000001
 675 004240 005737 002436
 676 004244 001401
 677 004246 104011
 678 004250 005037 002436 1\$:
 679 004254 112710 000120
 680 004260 106427 000340
 681 004264 005737 002436
 682 004270 001401
 683 004272 104011
 684 004274 005037 002436 2\$:
 685 004300 106427 000000
 686 004304 112712 000241
 687
 688
 689 004310 112710 000241
 690 004314 012737 000001 002124
 691 004322 013737 002436 002126
 692 004330 023737 002124 002126
 693 004336 001401
 694 004340 104012
 695 004342 106427 000340 3\$:
 696 004346 010137 002122 4\$:
 697 004352 005037 002126

```

    JMP 120$ ;DO GROUP 2
    *****
    *TEST 2 TEST ACR/ISR INTERRUPT ,GROUP 1, GROUP 2
    *****
    TST2: SCOPE
    MOV DRCSA,R0
    MOV DRCSB,R1
    MOV DRCSA,R2
    MOV #111$,SLPERR ;SET UP LOOP,RETURN
    MOV #2,GRPCNT ;DO TWO GROUPS
    JSR R5,CLRCSR ;CLEAR ALL CSRS
    MOV #BGCHP3,R3 ;EXPECTED GDDAT, FOR ISR
    MOV #CSIMR,IMRLOC ;STORE CLEAR SINGLE IMR CODE
    JSR R5,CLRIRR ;CLEAR IRR REGS
    MOV #INTSR1,300 ;STORE VECTOR ROUTINE
    MOV #340,302 ;STORE PSW
    111$: CLR INTFLG ;CLEAR INT. FLAG
    MOV #2100,SP ;INIT STACK IN CASE OF ILLEGAL INT. LOOP
    BIC #BIT9,(R2) ;CLEAR INTERRUPT ENABLE
    MOVB #242,(R2) ;CLEAR MASTER MASK GP1
    MOVB #242,(R0) ;CLEAR MASTER MASK GP1,GP2
    MOVB #PACR,(R0) ;PRESELECT ACR FOR WRITING
    MOVB #R3,(R1) ;NEXT BIT INTO ACR
    MOVB #PVMA,(R0) ;PRESELECT VECTOR MEM ADDR
    MOVB #60,(R1) ;WRITE VECTOR 300,BY0,LV0
    MOV R0,$BDADR ;STORE BUS ADDRESS
    MOVB #SIMR,(R0) ;SET ALL IMR BITS
    MOVB IMRLOC,(R0) ;CLEAR MASK ON SINGLE BIT
    MOVB #202,(R0) ;LMD04 - COMMON VECTOR
    MTPS #PRO
    BISB #BIT1,1(R2) ;SET I/E,NO INTERRUPT
    TST INTFLG ;CHECK FOR INTERRUPTS
    BEQ 1$
    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$: CLR INTFLG ;CLEAR INT. COUNTER
    MTPS #PRO
    MOVB #241,(R2) ;USED IN TEST OF GROUP 2
    ;ARM GROUP 1 CHIP TO PASS
    ;ENABLE TO GROUP 2 CHIP.
    ;ARM THE CHIP
    MOVB #241,(R0)
    MOV #1,$GDDAT ;EXPECT ONE INTERRUPT
    MOV INTFLG,$BDDAT ;SHOULD HAVE ONE INTERRUPT
    CMP $GDDAT,$BDDAT
    BEQ 3$
    ERROR 12 ;INTERRUPT TEST ERROR
    3$: MTPS #PR7
    4$: MOV R1,$BDADR ;READY TO READ ISR REG.
    CLR $BDDAT ;SET UP FOR BYTE READS
    
```

```

698 004356 005037 002124 CLR $GDDAT ;SET UP FOR BYTE EXPECTED
699 004362 112710 000240 MOVB #MISR,(R0) ;LOAD MODE TO READ ISR
700 004366 111137 002126 MOVB (R1),$BDDAT
701 004372 023737 002124 002126 CMP $GDDAT,$BDDAT ;ACR SHOULD CLEAR ISR
702 004400 001401 BEQ 5$
703 004402 104006 ERROR 6 ;ISR ERROR
704 004404 116337 000001 002124 5$: MOVB 1(R3),$GDDAT ;STORE EXPECTED
705 004412 112710 000244 MOVB #MIMR,(R0) ;LOAD MODE FOR IMR
706 004416 111137 002126 MOVB (R1),$BDDAT
707 004422 023737 002124 002126 CMP $GDDAT,$BDDAT
708 004430 001401 BEQ 6$
709 004432 104005 ERROR 5 ;IMR ERROR
710 004434 116337 000001 002124 6$: MOVB 1(R3),$GDDAT ;EXPECTED IRR
711 004442 112710 000250 MOVB #MIRR,(R0) ;LOAD MODE BITS FOR IRR
712 004446 111137 002126 MOVB (R1),$BDDAT
713 004452 023737 002124 002126 CMP $GDDAT,$BDDAT
714 004460 001401 BEQ 7$
715 004462 104003 ERROR 3 ;IRR ERROR
716 004464 111337 002124 7$: MOVB (R3),$GDDAT ;EXPECTED ACR
717 004470 112710 000254 MOVB #MACR,(R0)
718 004474 111137 002126 MOVB (R1),$BDDAT
719 004500 023737 002124 002126 CMP $GDDAT,$BDDAT
720 004506 001401 BEQ 10$
721 004510 104004 ERROR 4 ;ACR ERROR
722 004512 005723 10$: TST (R3)+ ;GET NEXT ISR,IMR,IRR EXPECTED RESULTS
723 004514 020327 012664 CMP R3,#EDCHP3 ;CHECK FOR END
724 004520 001404 BEQ 11$ ;DONE WITH GROUP TEST
725 004522 005237 002442 INC IMRLOC ;INCREMENT MASK BIT TO CLEAR
726 004526 000137 004144 JMP 111$ ;TEST NEXT ISR BIT
727 004532 005337 002462 11$: DEC GRPCNT ;TESTED BOTH GROUPS?
728 004536 001406 BEQ TST3 ;BR IF DONE
729 004540 013700 002422 MOV DRCSA,R0 ;SETUP FOR GROUP 2
730 004544 013701 002426 MOV DRCSA,R1 ;CSRC = CONTROL GROUP 2
731 ;CSRD = DATA GROUP 2
732 004550 000137 004106 JMP 120$ ;DO GROUP 2
733
734
735
736
737 ;*****
738 ;*TEST 3 TEST VECTOR ADDR MEM,LEV 0-7,BY0-3,(204-1774),GRPS 1,2
739 ;*****
739 004554 000004 TST3: SCOPE
740 004556 013700 002412 MOV DRCSA,R0
741 004562 013701 002416 MOV DRCSB,R1
742 004566 013702 002412 MOV DRCSA,R2
743 004572 012737 000002 002462 MOV #2,GRPCNT ;DO TWO GROUPS
744 004600 012737 000001 002472 120$: MOV #1,BYNUM ;INIT FOR BYTE COUNT 0
745 004606 004537 010650 JSR R5,CLRCSR ;CLEAR ALL CSRS
746 004612 012737 000340 002456 MOV #PVMA,VECVL ;START VECTOR LEVEL 0,BYTE COUNT 0
747 004620 012704 012644 121$: MOV #BGCHP3,R4 ;EXPECTED ISR,IMR PATTERN
748 004624 012737 000010 002466 MOV #8,LVLCNT ;COUNTER FOR 0-7 LEVELS
749 004632 012737 000050 002442 MOV #CSIMR,IMRLOC ;STORE SINGLE IMR CODE
750 004640 012737 000130 002446 MOV #SSIRR,IRRLOC ;STORE SINGLE IRR CODE
751 004646 012737 000170 002444 MOV #CSISR,ISRLOC ;STORE CLEAR SINGLE ISR CODE
752 004654 004537 011334 JSR R5,TRPCAT ;RESTORE TRAP CATCHER
753 004660 012737 004674 002110 MOV #112$,$LPERR ;STORE SCOPE LOOP

```

```

754 004666 012737 000204 002452 111$: MOV #204,VECLOC ;START AT VECTOR 204
755 004674 106427 000340 112$: MTPS #340 ;
756 004700 012706 002100 MOV #2100,SP ;INIT STACK IN CASE OF ILLEGAL INT. LOOP
757 004704 042712 001000 BIC #BIT9,(R2) ;CLEAR INTERRUPT ENABLE
758 004710 004537 010730 JSR R5,CLRIRR ;CLEAR IRR REGS
759 004714 112712 000241 MOVB #241,(R2) ;USED IN TEST OF GROUP 2
760 ;ARM GROUP 1 CHIP TO PASS
761 ;ENABLE TO GROUP2
762 004720 013703 002452 MOV VECLOC,R3 ;GET VECTOR
763 004724 010337 002454 MOV R3,VECPAT
764 004730 006037 002454 ROR VECPAT ;ROTATE VECTOR ADDR TWO RIGHT
765 004734 006037 002454 ROR VECPAT ;TO WRITE VECTOR MEM ADDR.
766 004740 012713 011510 MOV #INTBY4,(R3) ;STORE VECTOR ROUTINE
767 004744 012763 000340 000002 MOV #340,2(R3) ;STORE PSW
768 004752 005037 002436 CLR INTFLG ;CLEAR INT. FLAG
769 004756 013737 002472 002470 MOV BYNUM,BYCNT ;BYTE COUNTS OF 0-3
770 004764 113710 002456 MOVB VECVAL,(R0) ;PRESFLECT VECTOR MEM ADDR
771 004770 113711 002454 113$: MOVB VECPAT,(R1) ;WRITE VECTOR
772 004774 005337 002470 DEC BYCNT ;DONE WITH BYTE COUNT VALUE
773 005000 001373 BNE 113$ ;NO LOAD VECTOR FOR NEXT BYTE COUNT
774 005002 010037 002122 MOV R0,$BDADR ;SAVE BUS ADDRESS
775 005006 112710 000060 MOVB #SIMR,(R0) ;SET ALL IMR BITS
776 005012 113710 002442 MOVB IMRLOC,(R0) ;CLEAR MASK ON SINGLE BIT
777 005016 106427 000000 MTPS #PRO
778 005022 112710 000241 MOVB #241,(R0) ;LMD57 - ARM THE CHIP
779 005026 005737 002436 TST INTFLG ;CHECK FOR INTERRUPTS
780 005032 001401 BEQ 1$
781 005034 104011 ERROR 11 ;ERROR,ILLEGAL INTERRUPT
782 005036 005037 002436 1$: CLR INTFLG ;RESET INT. COUNTER
783 005042 113710 002446 MOVB IRRLOC,(R0) ;SET SINGLE IRR BIT
784 005046 106427 000340 MTPS #PR7
785 005052 005737 002436 TST INTFLG ;CHECK FOR NO INTERRUPTS
786 005056 001401 BEQ 2$
787 005060 104011 ERROR 11 ;ILLEGAL INTERRUPTS
788 005062 005037 002436 2$: CLR INTFLG ;CLEAR INT. COUNTER
789 005066 106427 000000 MTPS #PRO
790 005072 152762 000002 000002 000002 MOV BISB #BIT1,1(R2) ;SET I/E,EXPECT INTERRUPT
791 005100 013737 002472 002124 MOV BYNUM,$GDDAT ;EXPECT 1 TO 4 INTERRUPTS
792 ;BASED ON BYTE COUNT VALUE
793 005106 013737 002436 002126 MOV INTFLG,$BDDAT ;SHOULD HAVE 1 TO 4 INTERRUPTS
794 ;BASED ON BYTE COUNT VALUE
795 005114 023737 002124 002126 CMP $GDDAT,$BDDAT
796 005122 001401 BEQ 3$
797 005124 104012 ERROR 12 ;INTERRUPT TEST ERROR
798 005126 106427 000340 3$: MTPS #PR7
799 005132 010137 002122 4$: MOV R1,$BDADR ;READY TO READ ISR REG.
800 005136 005037 002126 CLR $BDDAT ;SET UP FOR BYTE READS
801 005142 005037 002124 CLR $GDDAT ;SET UP FOR BYTE EXPECTED
802 005146 111437 002124 MOVB (R4),$GDDAT ;EXPECTED DATA
803 005152 112710 000240 MOVB #MISR,(R0) ;LOAD MODE TO READ ISR
804 005156 111137 002126 MOVB (R1),$BDDAT
805 005162 023737 002124 002126 CMP $GDDAT,$BDDAT
806 005170 001401 BEQ 5$
807 005172 104006 ERROR 6 ;ISR ERROR
808 005174 116437 000001 002124 5$: MOVB 1(R4),$GDDAT ;STORE EXPECTED
809 005202 112710 000244 MOVB #MIMR,(R0) ;LOAD MODE FOR IMR

```

```

810 005206 111137 002126      MOVB  (R1), $BDDAT
811 005212 023737 002124 002126      CMP   $GDDAT, $BDDAT
812 005220 001401                BEQ   6$
813 005222 104005                ERROR 5 ;IMR ERROR
814 005224 005037 002124      6$: CLR   $GDDAT
815 005230 112710 000250      MOVB  #IMIR, (R0) ;LOAD MODE BITS FOR IRR
816 005234 111137 002126      MOVB  (R1), $BDDAT
817 005240 023737 002124 002126      CMP   $GDDAT, $BDDAT
818 005246 001401                BEQ   7$
819 005250 104003                ERROR 3 ;IRR ERROR
820 005252 005037 002124      7$: CLR   $GDDAT
821 005256 112710 000254      MOVB  #MACR, (R0)
822 005262 111137 002126      MOVB  (R1), $BDDAT
823 005266 023737 002124 002126      CMP   $GDDAT, $BDDAT
824 005274 001401                BEQ   10$
825 005276 104004                ERROR 4 ;ACR ERROR
826 005300 005037 002124      10$: CLR  $GDDAT
827 005304 113710 002444      MOVB  ISRLOC, (R0) ;CLEAR SINGLE ISR BIT
828 005310 112710 000240      MOVB  #MISR, (R0) ;LOAD MODE TO READ ISR
829 005314 111137 002126      MOVB  (R1), $BDDAT
830 005320 023737 002124 002126      CMP   $GDDAT, $BDDAT
831 005326 001401                BEQ   11$
832 005330 104006                ERROR 6 ;ISR REG ERROR
833 005332 004537 011242      11$: JSR  R5, RESTRP ;RESTORE VECTOR JUST TESTED
834 005336 022737 002000 002452      CMP   #2000, VECLOC ;FINISHED?
835 005344 001402                BEQ   12$ ;FINISHED VECTORS 204-1774?
836 005346 000137 004674      JMP   112$ ;TEST NEXT VECTOR ADDR
837 005352 005724      12$: TST  (R4)+ ;INDEX EXPECTED ISR AND IMR PATTERN
838 005354 005237 002442      INC   IMRLOC ;STORE CLEAR FOR NEXT MASK BIT
839 005360 005237 002446      INC   IRRLOC ;STORE SET FOR NEXT IRR BIT
840 005364 005237 002444      INC   ISRLOC ;STORE CLEAR FOR NEXT ISR BIT
841 005370 005237 002456      INC   VECVAL ;SETUP TO TEST NEXT VECTOR LEVEL
842 005374 005337 002466      DEC   LVL CNT ;FINISHED LEVELS 0-7
843 005400 001402                BEQ   13$ ;YES, CHECK BYTE COUNT END
844 005402 000137 004666      JMP   111$ ;NO, DO NEXT LEVEL
845 005406 005237 002472      13$: INC  BYNUM ;INDEX BYTE COUNT
846 005412 104407                CKSWR
847 005414 022737 000400 002456      CMP   #400, VECVAL ;LOOK FOR SWITCH CHANGE REQUEST
848 005422 001402                BEQ   14$ ;FINISHED ALL BYTE COUNTS 0-3
849 005424 000137 004620      JMP   121$ ;FINISHED TWO GROUPS?
850                                ;DO NEXT BYTE COUNT FOR
851 005430 005337 002462      14$: DEC  CRPCNT ;LEVELS 0-7
852 005434 001406                BEQ   TCT4 ;FINISHED TWO GROUPS?
853 005436 013700 002422      MOV   DRCSA, R0 ;;BR IF DONE
854 005442 013701 002426      MOV   DRCSB, R1 ;CSRC = CONTROL GROUP 2
855 005446 000137 000600      JMP   120$ ;CSRD = DATA GROUP 2
856                                ;DO GROUP 2 FOR BYTE COUNTS 0-3
857                                ;AND LEVELS 0-7 FOR EACH BYTE COUNT.
858
859
860
861
862 005452 000004      *****
863 005454 013700 002412      *TEST 4 TEST VECTOR ADDR MEM, LVLS 0-7, BY0-3, (0-200)GRPS 1,2
864 005460 013701 002416      *****
865 005464 013702 002412      TST4: SCOPE
      MOV   DRCSA, R0
      MOV   DRCSB, R1
      MOV   DRCSA, R2
    
```


922	006022	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
923	006030	001405				BEQ	3\$	
924	006032	004537	011204			JSR	R5,FRMBUF	:RESTORE 0-202
925	006036	104012				ERROR	12	:INTERRUPT TEST ERROR
926	006040	004537	011634			JSR	R5,CAT200	:RESTORE TRAP CATCHER SUB
927	006044	106427	000340		3\$:	MTPS	#PR7	
928	006050	010137	002122		4\$:	MOV	R1,\$BADDR	:READY TO READ ISR REG.
929	006054	005037	002126			CLR	\$BDDAT	:SET UP FOR BYTE READS
930	006060	005037	002124			CLR	\$GDDAT	:SET UP FOR BYTE EXPECTED
931	006064	111437	002124			MOVB	(R4),\$GDDAT	:EXPECTED DATA
932	006070	112710	000240			MOVB	#MISR,(R0)	:LOAD MODE TO READ ISR
933	006074	111137	002126			MOVB	(R1),\$BDDAT	
934	006100	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
935	006106	001405				BEQ	5\$	
936	006110	004537	011204			JSR	R5,FRMBUF	:RESTORE 0-200
937	006114	104006				ERROR	6	:ISR ERROR
938	006116	004537	011634			JSR	R5,CAT200	:RESTORE TRAP CATCHER
939	006122	116437	000001	002124	5\$:	MOVB	1(R4),\$GDDAT	:STORE EXPECTED
940	006130	112710	000244			MOVB	#MIMR,(R0)	:LOAD MODE FOR IMR
941	006134	111137	002126			MOVB	(R1),\$BDDAT	
942	006140	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
943	006146	001405				BEQ	6\$	
944	006150	004537	011204			JSR	R5,FRMBUF	:RESTORE 0-202
945	006154	104005				ERROR	5	:IMR ERROR
946	006156	004537	011634			JSR	R5,CAT200	:RESTORE CATCHER
947	006162	005037	002124		6\$:	CLR	\$GDDAT	
948	006166	112710	000250			MOVB	#MIRR,(R0)	:LOAD MODE BITS FOR IRR
949	006172	111137	002126			MOVB	(R1),\$BDDAT	
950	006176	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
951	006204	001405				BEQ	7\$	
952	006206	004537	011204			JSR	R5,FRMBUF	:RESTORE 0-202
953	006212	104003				ERROR	3	:IRR ERROR
954	006214	004537	011634			JSR	R5,CAT200	:RESTORE TRAP CATCHER
955	006220	005037	002124		7\$:	CLR	\$GDDAT	
956	006224	112710	000254			MOVB	#MACR,(R0)	
957	006230	111137	002126			MOVB	(R1),\$BDDAT	
958	006234	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
959	006242	001405				BEQ	10\$	
960	006244	004537	011204			JSR	R5,FRMBUF	:RESTORE 0-202
961	006250	104004				ERROR	4	:ACR ERROR
962	006252	004537	011634			JSR	R5,CAT200	:RESTORE TRAP CATCHER
963	006256	005037	002124		10\$:	CLR	\$GDDAT	
964	006262	112710	000160			MOVB	#CISR,(R0)	:CLEAR ISR
965	006266	112710	000240			MOVB	#MISR,(R0)	:LOAD MODE TO READ ISR
966	006272	111137	002126			MOVB	(R1),\$BDDAT	
967	006276	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
968	006304	001405				BEQ	11\$	
969	006306	004537	011204			JSR	R5,FRMBUF	:RESTORE 0-202
970	006312	104006				ERROR	6	:ISR REG ERROR
971	006314	004537	011634			JSR	R5,CAT200	:RESTORE TRAP CATCHER
972	006320	004537	011572		11\$:	JSR	R5,RES200	:RESTORE VECTOR JUST TESTED
973	006324	004537	011204			JSR	R5,FRMBUF	:RESTORE 0-200 TO SYSMAC CONTROL
974	006330	022737	000204	002452		CMP	#204,VECLOC	:FINISHED?
975	006336	001402				BEQ	12\$	
976	006340	000137	005556			JMP	112\$:TEST NEXT VECTOR ADDR
977	006344	005724			12\$:	TST	(R4)	:INDEX EXPECTED ISR AND IMR PATERN

T4 TEST VECTOR ADDR MEM, LVLS 0-7, BY0-3, (0-200)GRPS 1,2

978	006346	005237	002442		INC	IMRLOC	:STORE CLEAR NEXT IMR BIT
979	006352	005237	002446		INC	IRRLOC	:STORE SET NEXT IRR BIT
980	006356	005237	002456		INC	VECVL	:SETUP TO TEST NEXT VECTOR LEVEL
981	006362	005337	002466		DEC	LVL CNT	:FINISHED LEVELS 0-7?
982	006366	001402			BEQ	13\$:YES,CHECK BYTE COUNT END
983	006370	000137	005552		JMP	111\$:NO,DO NEXT LEVEL,SAME BYTE COUNT
984	006374	005237	002472	13\$:	INC	BYNUM	:INDEX BYTE COUNT
985	006400	104407			CKSWR		:LOOK FOR SWITCH CHANGE
986	006402	022737	000400	002456	CMR	#400,VECVL	:FINISHED ALL FOUR BYTE COUNTS
987	006410	001402			BEQ	14\$:YES,FINISHED GROUP
988	006412	000137	005516		JMP	121\$:NO,DO NEXT BYTE COUNT WITH
989							:LEVELS 0-7.
990	006416	005337	002462	14\$:	DEC	GRPCNT	:FINISHED BOTH GROUP 1 AND GROUP 2?
991	006422	001406			BEQ	15\$:YES,BOTH GROUPS TESTED
992	006424	013700	002422		MOV	DRCSC,R0	:NO,TEST GROUP 2
993							:CSRC = CONTROL GROUP 2
994	006430	013701	002426		MOV	DRCSD,R1	:CSRD = DATA GROUP 2
995	006434	000137	005476		JMP	120\$:RETURN AND TEST GROUP2
996							:FOR BYTE COUNTS 0-3 AND
997							:LEVELS 0-7 FOR EACH BYTE COUNT.
998	006440	004537	011204	15\$:	JSR	R5,FRMBUF	:RESTORE 0-202,EXIT TEST
999							

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1004	006444	000004					
1007	006446	004537	010650		JSR	R5,CLRCSR	:CLEAR ALL CSRS
1008	006452	004537	011144		JSR	R5,CLRBF	:CLEAR VECTOR BUFFER AREA
1009	006456	004537	011334		JSR	R5,TRPCAT	:RESTORE TRAP CATCHER
1010	006462	004537	011110		JSR	R5,STRVEL	:STORE VECTOR AREA 300-674
1012	006466	012737	000377	002456	MOV	#377,VECVL	:VECTOR MEM FINAL VALUE
1013	006474	004537	010730		JSR	R5,CLRIRR	:CLEAR IRR REGS
1014	006500	012704	017116		MOV	#VBUF,R4	:VECTOR ADDRESS BUFFER
1015	006504	004537	010766		JSR	R5,VECFIL	:FILL VECTOR MEMORY FOR GROUPS 1,2
1017	006510	013700	002412		MOV	DRCSA,R0	:GROUP 1 CSR
1018	006514	013701	002422		MOV	DRCSC,R1	:GROUP 2 CSR
1019	006520	013702	002416		MOV	DRCSE,R2	
1020	006524	013703	002426		MOV	DRCSD,R3	
1021	006530	010037	002122		MOV	R0,\$BDADR	:STORE BUS ADDRESS
1022	006534	112710	000300		MOVB	#PACR,(R0)	:PRESELECT GP1 ACR FOR WRITING
1023	006540	112712	000377		MOVB	#377,(R2)	:SET ALL ACR BITS GP1
1024	006544	112710	000040		MOVB	#CIMR,(R0)	:CLEAR ALL IMR BITS GP1
1025	006550	005037	002436		CLR	INTFLG	:CLEAR INT. FLAG
1026	006554	106427	000000		MTPS	#PRO	
1027	006560	112710	000120		MOVB	#SIRR,(R0)	:SET ALL IRR BITS
1028	006564	112710	000241		MOVB	#241,(R0)	:LMD57 - ARM THE CHIP
1029	006570	106427	000340		MTPS	#340	
1030	006574	005737	002436		TST	INTFLG	:CHECK FOR NO INTERRUPTS
1031	006600	001401			BEQ	1\$	
1032	006602	104011			ERROR	11	:ILLEGAL INTERRUPT
1033	006604	005037	002436	1\$:	CLR	INTFLG	

 :TEST 5 TEST VECTOR ADDR UNIQUENESS IN FIXED MODE FOR GRPS 1,2

TST5: SCOPE

:BYTE COUNT = 4
 :LEVELS = 0-7
 :CLEAR ALL CSRS
 :CLEAR VECTOR BUFFER AREA
 :RESTORE TRAP CATCHER
 :STORE VECTOR AREA 300-674
 :WITH INT. SERV ROUTINES
 :VECTOR MEM FINAL VALUE
 :CLEAR IRR REGS
 :VECTOR ADDRESS BUFFER
 :FILL VECTOR MEMORY FOR GROUPS 1,2
 :WITH VECTORS 300-674
 :GROUP 1 CSR
 :GROUP 2 CSR

Line	Address	PC	Next PC	Value	Op	Comment
1034	006610	112711	000300		MOVB #PACR, (R1)	;PRESELECT GROUP2 FOR WRITING
1035	006614	112713	000377		MOVB #377, (R3)	;ALL ACR BITS SET GP2
1036	006620	112711	000040		MOVB #CIMR, (R1)	;CLEAR ALL IMR BITS GP2
1037	006624	106427	000000		MTPS #PRO	
1038	006630	112711	000120		MOVB #SIRR, (R1)	;SET ALL IRR BITS
1039	006634	112711	000241		MOVB #241, (R1)	;LMD57 - ARM THE CHIP
1040	006640	106427	000340		MTPS #PR7	
1041	006644	005737	002436		TST INTFLG	;CHECK FOR NO INTERRUPTS
1042	006650	001401			BEQ 2\$	
1043	006652	104011			ERROR 11	;ILLEGAL INTERRUPT
1044	006654	005037	002436	2\$:	CLR INTFLG	
1045	006660	106427	000000		MTPS #PRO	
1046	006664	152760	000002	000071	BISB #BIT1, 1(R0)	;SET INT. ENABLE, EXPECT INTERRUPTS
1047	006672	012737	000100	002124	MOV #64, \$GDDAT	;EXPECT 64 INTERRUPTS
1048	006700	013737	002436	002126	MOV INTFLG, \$BDDAT	;SHOULD HAVE 64 INTERRUPTS
1049	006706	106427	000340		MTPS #PR7	
1050	006712	023737	002124	002126	CMP \$GDDAT, \$BDDAT	
1051	006720	001401			BEQ 3\$	
1052	006722	104012			ERROR 12	;INTERRUPT TEST ERROR
1053	006724	012737	101710	002124	MOV #101710, \$GDDAT	;RDY, DIR, I/E, CHIP - 31X
1054	006732	010037	002122		MOV R0, \$BDADR	;CSRA
1055	006736	011037	002126		MOV (R0), \$BDDAT	
1056	006742	042737	000007	002126	BIC #7, \$BDDAT	;CLEAR UNDEFINED BITS
1057	006750	023737	002124	002126	CMP \$GDDAT, \$BDDAT	
1058	006756	001401			BEQ 4\$	
1059	006760	104002			ERROR 2	;CSRA REG ERROR
1060	006762	012737	000310	002124	MOV #310, \$GDDAT	;STORE EXPECTED
1061	006770	010137	002122		MOV R1, \$BDADR	;CSRC
1062	006774	011137	002126		MOV (R1), \$BDDAT	
1063	007000	042737	000007	002126	BIC #7, \$BDDAT	;CLEAR UNDEFINED BITS
1064	007006	023737	002124	002126	CMP \$GDDAT, \$BDDAT	
1065	007014	001401			BEQ 5\$	
1066	007016	104002			ERROR 2	;CSRC ERROR
1067	007020	010237	002122		MOV R2, \$BDADR	;CSRB ADDRESS
1068	007024	005037	002126		CLR \$BDDAT	
1069	007030	005037	002124		CLR \$GDDAT	
1070	007034	112710	000240		MOVB #MISR, (R0)	;READ ISR
1071	007040	111237	002126		MOVB (R2), \$BDDAT	
1072	007044	023737	002124	002126	CMP \$GDDAT, \$BDDAT	
1073	007052	001401			BEQ 6\$	
1074	007054	104006			ERROR 6	;ISR ERROR, GP1
1075	007056	112710	000244		MOVB #MIMR, (R0)	;MODE BITS FOR IMR
1076	007062	111237	002126		MOVB (R2), \$BDDAT	
1077	007066	023737	002124	002126	CMP \$GDDAT, \$BDDAT	
1078	007074	001401			BEQ 7\$	
1079	007076	104005			ERROR 5	;IMR ERROR, GP1
1080	007100	112710	000250		MOVB #MIRR, (R0)	;MODE BITS FOR IRR
1081	007104	111237	002126		MOVB (R2), \$BDDAT	
1082	007110	023737	002124	002126	CMP \$GDDAT, \$BDDAT	
1083	007116	001401			BEQ 10\$	
1084	007120	104003			ERROR 3	;IRR ERROR
1085	007122	012737	000377	002124	MOV #377, \$GDDAT	;EXPECTED ACR
1086	007130	112710	000254		MOVB #MACR, (R0)	;READ ACR
1087	007134	111237	002126		MOVB (R2), \$BDDAT	
1088	007140	023737	002124	002126	CMP \$GDDAT, \$BDDAT	
1089	007146	001401			BEQ 11\$	

```
1090 007150 104004          ERROR 4          ;ACR ERROR
1091 007152 005037 002124 11$: CLR $GDDAT
1092 007156 005037 002126      CLR $BDDAT
1093 007162 010337 002122      MOV R3,$BDADR    ;CSRD ADDRESS
1094 007166 112711 000240      MOVB #MISR,(R1)  ;READ ISR ,GP2
1095 007172 111337 002126      MOVB (R3),$BDDAT
1096 007176 023737 002124 002126  CMP $GDDAT,$BDDAT
1097 007204 001401          BEQ 12$
1098 007206 104006          ERROR 6          ;ISR ERROR,GP1
1099 007210 112711 000244 12$: MOVB #MIMR,(R1)  ;READ IMR GP2
1100 007214 111337 002126      MOVB (R3),$BDDAT
1101 007220 023737 002124 002126  CMP $GDDAT,$BDDAT
1102 007226 001401          BEQ 13$
1103 007230 104005          ERROR 5          ;IMR ERROR,GP2
1104
1105 007232 112711 000250 13$: MOVB #MIRR,(R1)  ;READ IRR
1106 007236 111337 002126      MOVB (R3),$BDDAT
1107 007242 023737 002124 002126  CMP $GDDAT,$BDDAT
1108 007250 001401          BEQ 14$
1109 007252 104003          ERROR 3          ;IRR ERROR,GP2
1110 007254 112711 000254 14$: MOVB #MACR,(R1)  ;READ ACR
1111 007260 012737 000377 002124  MOV #377,$GDDAT
1112 007266 111337 002126      MOVB (R3),$BDDAT
1113 007272 023737 002124 002126  CMP $GDDAT,$BDDAT
1114 007300 001401          BEQ 15$
1115 007302 104004          ERROR 4          ;ACR ERROR,GP2
1116 007304 012702 000100 15$: MOV #64.,R2
1117 007310 012704 017116      MOV #VBUF,R4    ;VECTOR BUFFER
1118 007314 012737 000300 002124  MOV #300,$GDDAT ;START WITH FIRST VECTOR 300
1119 007322 010437 002122 16$: MOV R4,$BDADR    ;VECTOR BUFFER ADDRESS
1120 007326 011437 002126      MOV (R4),$BDDAT
1121 007332 023737 002124 002126  CMP $GDDAT,$BDDAT
1122 007340 001401          BEQ 17$
1123 007342 104007          ERROR 7          ;VECTORED PROPERLY
1124 007344 062737 000004 002124 17$: ADD #4,$GDDAT    ;VECTOR ADDR MEM ERROR
1125 007352 062704 000002      ADD #2,R4       ;NEXT BUFFER ADDRESS
1126 007356 005302          DEC R2          ;COMPLETED 64 BUFFER CHECKS?
1127 007360 001360          BNE 16$        ;CHECK NEXT VECTOR BUFFER LOCATION
1128                          ;THAT WAS STORED BY THE INTERRUPT
1129                          ;SERVICE ROUTINES.
```

```
1130
1131
1132 ::*****
1133 :*TEST 6 TEST VECTOR ADDR UNIQUENESS IN ROTATING MODE FOR GRPS 1,2
1134 :*****
1135 TST6: SCOPE
1136 ;BYTE COUNT 4
1137 ;LEVELS = 0-7
1138 JSR R5,CLRCSR ;CLEAR ALL CSRS
1139 JSR R5,CLRBF ;CLEAR VECTOR BUFFER AREA
1140 JSR R5,TRPCAT ;RESTORE TRAP CATCHER
1141 JSR R5,STRVEC ;STORE VECTOR AREA 300-674
1142 ;WITH INT. SERV ROUTINES
1143 MOV #377,VECVAL ;VECTOR MEM FINAL VALUE
1144 JSR R5,CLRIRR ;CLEAR IRR REGS
1145 MOV #VBUF,R4 ;VECTOR ADDRESS BUFFER
1146 JSR R5,VECFIL ;FILL VECTOR MEMORY FOR GROUP 1,2
```

```
1146                                     ;WITH VECTORS 300-674
1147 007426 013700 002412          MOV    DRCSA,R0          ;GROUP 1 CSR
1148 007432 013701 002422          MOV    DRCSB,R1          ;GROUP 2 CSR
1149 007436 013702 002416          MOV    DRCSB,R2
1150 007442 013703 002426          MOV    DRCSB,R3
1151 007446 010037 002122          MOV    R0,$BDADR        ;STORE BUS ADDRESS
1152 007452 012737 000010 002474  MOV    #8,LVLSAV        ;COUNTER FOR 7 LEVELS
1153 007460 112710 000201          MOVB   #201,(R0)        ;ROTATING PRIORITY GROUP 1
1154 007464 112710 000040          MOVB   #CIMR,(R0)       ;CLEAR ALL IMR BITS GP1
1155 007470 005037 002436          CLR    INTFLG           ;CLEAR INT. FLAG
1156 007474 106427 000000          MTPS   #PRO
1157 007500 112710 000120          MOVB   #SIRR,(R0)       ;SET ALL IRR BITS
1158 007504 112710 000241          MOVB   #241,(R0)       ;LMD57 - ARM THE CHIP
1159 007510 106427 000340          MTPS   #340
1160 007514 005737 002436          TST    INTFLG           ;CHECK FOR NO INTERRUPTS
1161 007520 001401                    BEQ    1$
1162 007522 104011                    ERROR  11                ;ILLEGAL INTERRUPT
1163 007524 005037 002436          CLR    INTFLG           ;ILLEGAL INTERRUPT
1164 007530 112711 000201          MOVB   #201,(R1)        ;ROTATING PRIORITY FOR GROUP 2
1165 007534 112711 000040          MOVB   #CIMR,(R1)       ;CLEAR ALL IMR BITS GP2
1166 007540 106427 000000          MTPS   #PRO
1167 007544 112711 000120          MOVB   #SIRR,(R1)       ;SET ALL IRR BITS
1168 007550 112711 000241          MOVB   #241,(R1)       ;LMD57 - ARM THE CHIP
1169 007554 106427 000340          MTPS   #PR7
1170 007560 005737 002436          TST    INTFLG           ;CHECK FOR NO INTERRUPTS
1171 007564 001401                    BEQ    2$
1172 007566 104011                    ERROR  11                ;ILLEGAL INTERRUPT
1173 007570 005037 002436          CLR    INTFLG           ;ILLEGAL INTERRUPT
1174 007574 106427 000000          MTPS   #PRO
1175 007600 152760 000002 000001  BISB   #BIT1,1(R0)      ;SET INT. ENABLE,EXPECT INTERRUPTS
1176 007606 012737 000004 002124  MOV    #4,$GDDAT        ;EXPECT 4 INTERRUPTS
1177 007614 013737 002436 002126  MOV    INTFLG,$BDDAT    ;SHOULD HAVE 4 INTERRUPTS
1178 007622 106427 000340          MTPS   #PR7
1179 007626 023737 002124 002126  CMP    $GDDAT,$BDDAT
1180 007634 001401                    BEQ    100$
1181 007636 104012                    ERROR  12                ;INTERRUPT TEST ERROR
1182 007640 112710 000140 100$:   MOVB   #CHPISR,(R0)     ;CLEAR HIGHEST PRIORITY ISR
1183 007644 112710 000120          MOVB   #SIRR,(R0)       ;SET ALL IRR BITS
1184 007650 005337 002474          DEC    LVLSAV           ;DONE 8 LEVELS,GROUP 1
1185 007654 001345                    BNE    2$               ;DO NEXT LEVEL IN ROTATION
1186 007656 112710 000100          MOVB   #CIRR,(R0)       ;CLEAR IRR BITS GROUP 1
1187 007662 012737 000010 002474  MOV    #8,LVLSAV        ;DO 8 LEVELS,GROUP 2
1188 007670 005037 002436          CLR    INTFLG           ;CLEAR INT FLAG
1189 007674 106427 000000          MTPS   #PRO
1190 007700 012737 000004 002124  MOV    #4,$GDDAT
1191 007706 013737 002436 002126  MOV    INTFLG,$BDDAT    ;SHOULD HAVE FOUR INTERRUPTS
1192 007714 106427 000340          MTPS   #PR7
1193 007720 023737 002124 002126  CMP    $GDDAT,$BDDAT
1194 007726 001401                    BEQ    102$
1195 007730 104012                    ERROR  12                ;INTERRUPT TEST ERROR
1196 007732 112711 000140 102$:   MOVB   #CHPISR,(R1)     ;CLEAR HIGHEST PRIORITY ISR
1197 007736 112711 000120          MOVB   #SIRR,(R1)       ;SET ALL IRR BITS
1198 007742 005337 002474          DEC    LVLSAV           ;DONE 8 LEVELS?
1199 007746 001350                    BNE    101$            ;DO NEXT LEVEL IN ROTATION,GP2
1200 007750 112711 000100          MOVB   #CIRR,(R1)       ;CLEAR IRR BITS,GROUP 2
1201 007754 012737 101750 002124 3$:   MOV    #101750,$GDDAT  ;RDY,DIR,I/E,CHIP - 35X
```

1202	007762	010037	002122			MOV	R0,\$BDADR	:CSRA
1203	007766	011037	002126			MOV	(R0),\$BDDAT	
1204	007772	042737	000007	002126		BIC	#7,\$BDDAT	:CLEAR UNDEFINED BITS
1205	010000	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1206	010006	001401				BEQ	4\$	
1207	010010	104002				ERROR	2	:CSRA REG ERROR
1208	010012	012737	000350	002124	4\$:	MOV	#350,\$GDDAT	:STORE EXPECTED
1209	010020	010137	002122			MOV	R1,\$BDADR	:CSRC
1210	010024	011137	002126			MOV	(R1),\$BDDAT	
1211	010030	042737	000007	002126		BIC	#7,\$BDDAT	:CLEAR UNDEFINED BITS
1212	010036	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1213	010044	001401				BEQ	5\$	
1214	010046	104002				ERROR	2	:CSRC ERROR
1215	010050	010237	002122		5\$:	MOV	R2,\$BDADR	:CSRB ADDRESS
1216	010054	005037	002126			CLR	\$BDDAT	
1217	010060	005037	002124			CLR	\$GDDAT	
1218	010064	112710	000240			MOVB	#MISR,(R0)	:READ ISR
1219	010070	111237	002126			MOVB	(R2),\$BDDAT	
1220	010074	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1221	010102	001401				BEQ	6\$	
1222	010104	104006				ERROR	6	:ISR ERROR,GP1
1223	010106	112710	000244		6\$:	MOVB	#MIMR,(R0)	:MODE BITS FOR IMR
1224	010112	111237	002126			MOVB	(R2),\$BDDAT	
1225	010116	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1226	010124	001401				BEQ	7\$	
1227	010126	104005				ERROR	5	:IMR ERROR,GP1
1228	010130	112710	000250		7\$:	MOVB	#MIRR,(R0)	:MODE BITS FOR IRR
1229	010134	111237	002126			MOVB	(R2),\$BDDAT	
1230	010140	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1231	010146	001401				BEQ	10\$	
1232	010150	104003				ERROR	3	:IRR ERROR
1233	010152	112710	000254		10\$:	MOVB	#MACR,(R0)	:READ ACR
1234	010156	111237	002126			MOVB	(R2),\$BDDAT	
1235	010162	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1236	010170	001401				BEQ	11\$	
1237	010172	104004				ERROR	4	:ACR ERROR
1238	010174	010337	002122		11\$:	MOV	R3,\$BDADR	:CSRB ADDRESS
1239	010200	112711	000240			MOVB	#MISR,(R1)	:READ ISR,GP2
1240	010204	111337	002126			MOVB	(R3),\$BDDAT	
1241	010210	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1242	010216	001401				BEQ	12\$	
1243	010220	104006				ERROR	6	:ISR ERROR,GP1
1244	010222	112711	000244		12\$:	MOVB	#MIMR,(R1)	:READ IMR,GP2
1245	010226	111337	002126			MOVB	(R3),\$BDDAT	
1246	010232	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1247	010240	001401				BEQ	13\$	
1248	010242	104005				ERROR	5	:IMR ERROR,GP2
1249								
1250	010244	112711	000250		13\$:	MOVB	#MIRR,(R1)	:READ IRR
1251	010250	111337	002126			MOVB	(R3),\$BDDAT	
1252	010254	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	
1253	010262	001401				BEQ	14\$	
1254	010264	104003				ERROR	3	:IRR ERROR,GP2
1255	010266	112711	000254		14\$:	MOVB	#MACR,(R1)	:READ ACR
1256	010272	111337	002126			MOVB	(R3),\$BDDAT	
1257	010276	023737	002124	002126		CMP	\$GDDAT,\$BDDAT	

```
1258 010304 001401 BEQ 200$
1259 010306 104004 ERROR 4 :ACR ERROR, GP2
1260 010310 105010 200$: CLRB (R0) :INIT GROUP 1 MODE BITS
1261 010312 105011 CLRB (R1) :INIT GROUP 2 MODE BITS
1262 010314 012737 101700 002124 MOV #101700, $GDDAT :EXPECTED CSRA
1263 010322 010037 002122 MOV R0, $BDDADR :CSRA
1264 010326 011037 002126 MOV (R0), $BDDAT
1265 010332 042737 000007 002126 BIC #7, $BDDAT :CLEAR UNDEFINED BITS
1266 010340 023737 002124 002126 CMP $GDDAT, $BDDAT
1267 010346 001401 BEQ 201$
1268 010350 104002 ERROR 2 :CSRA REG ERROR
1269 010352 012737 000200 002124 201$: MOV #200, $GDDAT :STORE EXPECTED
1270 010360 010137 002122 MOV R1, $BDDADR :CSRC
1271 010364 011137 002126 MOV (R1), $BDDAT
1272 010370 042737 000007 002126 BIC #7, $BDDAT :CLEAR UNDEFINED BITS
1273 010376 023737 002124 002126 CMP $GDDAT, $BDDAT
1274 010404 001401 BEQ 15$
1275 010406 104002 ERROR 21 :CSRC ERROR
1276 010410 012702 000100 15$: MOV #64, R2
1277 010414 012704 017116 MOV #VBUF, R4 :VECTOR BUFFER
1278 010420 012737 000300 002124 MOV #300, $GDDAT :START WITH FIRST VECTOR 300
1279 010426 010437 002122 16$: MOV R4, $BDDADR :VECTOR BUFFER ADDRESS
1280 010432 011437 002126 MOV (R4), $BDDAT
1281 010436 023737 002124 002126 CMP $GDDAT, $BDDAT
1282 010444 001401 BEQ 17$
1283 010446 104007 ERROR 7 :VECTORED PROPERLY
1284 010450 062737 000004 002124 17$: ADD #4, $GDDAT :VECTOR ADDR MEM ERROR
1285 010456 062704 000002 ADD #2, R4 :NEXT BUFFER ADDRESS
1286 010462 005302 DEC R2 :COMPLETED 64 BUFFER CHECKS?
1287 010464 001360 BNE 16$ :CHECK NEXT VECTOR BUFFER LOCATION
1288 :THAT WAS STORED BY THE INTERRUPT
1289 :SERVICE ROUTINES.
1290
1291
1292 :DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11J'S HAVE BEEN TESTED.
1293 010466 013701 002412 NXDEV: MOV DRCSA, R1 :INIT TO SETUP DRV11J ADDRESS
1294 010472 000241 NXDEV1: CLC :CLEAR CARRY FOR DEVICE MAP
1295 010474 006037 002434 ROR DMAP :LOOK FOR NEXT DRV11J
1296 010500 001412 BEQ $EOP :BR IF ALL TESTED
1297 010502 162701 000020 SUB #20, R1 :NEXT DRV11J STARTS -20 FROM PAST CSR
1298 010506 005237 002202 INC $UNIT :UPDAT UNIT #
1299 010512 032737 000001 002434 BIT #1, DMAP :IS UNIT SELECTED?
1300 010520 001764 BEQ NXDEV1 :BR IF NOT
1301 010522 000137 003132 JMP NEXPAS :TEST NEXT DRV11J
1302
1303 .SBTTL END OF PASS ROUTINE
1304
1305 :*****
1306 :*INCREMENT THE PASS NUMBER ($PASS)
1307 :*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
1308 :*IF THERES A MONITOR GO TO IT
1309 :*IF THERE ISN'T JUMP TO START
1310
1311 010526 SEOP:
1312 010526 000240 NOP
1313 010530 005037 002102 CLR $STNUM :ZERO THE TEST NUMBER
```

1314	010534	005037	002160	CLR	\$TIMES	::ZERO THE NUMBER OF ITERATIONS
1315	010540	005237	002176	INC	\$PASS	::INGREMENT THE PASS NUMBER
1316	010544	042737	100000	BIG	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER
1317	010552	005327		DEC	(PC)+	::LOOP?
1318	010554	000001		\$EOPCT: .WORD	1	
1319	010556	003022		BGT	\$DOAGN	::YES
1320	010560	012737		MOV	(PC)+,@(PC)+	::RESTORE COUNTER
1321	010562	000001		\$ENDCT: .WORD	1	
1322	010564	010554		\$EOPCT		
1323	010566	104401	010633	TYPE	\$ENDMG	::TYPE 'END PASS #'
1324	010572	013746	002176	MOV	\$PASS,-(SP)	::SAVE \$PASS FOR TYPEOUT
1325	010576	104405		TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
1326	010600	104401	010630	TYPE	\$ENULL	::TYPE A NULL CHARACTER
1327	010604	013700	000042	\$GET42: MOV	@#42,R0	::GET MONITOR ADDRESS
1328	010610	001405		BEQ	\$DOAGN	::BRANCH IF NO MONITOR
1329	010612	000005		RESET		::CLEAR THE WORLD
1330	010614	004710		\$ENDAD: JSR	PC,(R0)	::GO TO MONITOR
1331	010616	000240		NOP		::SAVE ROOM
1332	010620	000240		NOP		::FOR
1333	010622	000240		NOP		::ACT11
1334	010624			\$DOAGN:		
1335	010624	000137		JMP	@(PC)+	::RETURN
1336	010626	003066		\$RTNAD: .WORD	START1	
1337	010630	377	377 1 000	\$ENULL: .BYTE	-1,-1,0	::NULL CHARACTER STRING
1338	010633	015	042412 042116	\$ENDMG: .ASCIZ	<15><12> /END PASS #/	
1339	010640	050040	051501 020123			
1340	010646	000043				

1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396

010650 010046
010652 106427 000340
010656 012737 000340 002456
010664 012737 000300 002452
010672 013700 002412
010676 005037 002124
010702 005037 002126
010706 005010
010710 105060 000005
010714 005060 000010
010720 105060 000015
010724 012600
010726 000205

010730 010046
010732 013700 002412
010736 105060 000011
010742 112760 000001 000001
010750 005060 000002
010754 105010
010756 105060 000010
010762 012600
010764 000205

010766 013700 002412
010772 013701 002416
010776 012702 000002
011002 012737 000060 002454
011010 012737 000370 002456 1\$:
011016 012737 000010 002466
011024 113710 002456 2\$:
011030 012737 000004 002470
011036 113711 002454 3\$:
011042 005237 002454
011046 005337 002470
011052 001371
011054 005337 002466 4\$:
011060 001403
011062 005237 002456 5\$:
011066 000756
011070 005302
011072 001405
011074 013700 002422
011100 013701 002426
011104 000741

```
.SBTTL PROGRAM SUBROUTINES  
  
:*****  
: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 GROUP 1  
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)  
3$: MOV #4,BYCNT ;DO FOUR BYTE COUNTS  
MOVB VECPAT,(R1) ;WRITE VECTOR  
INC VECPAT ;SETUP FOR NEXT VECTOR  
DEC BYCNT ;DONE FOUR BYTE COUNTS?  
BNE 3$ ;DO NEXT VECTOR  
4$: DEC LVLCNT ;DONE ALL LEVELS IN GROUP?  
BEQ 5$  
INC VECVAL ;INC NEXT LEVEL  
BR 2$ ;DO NEXT LEVEL  
5$: DEC R2 ;DONE BOTH GROUPS?  
BEQ 6$ ;DONE,EXIT  
MOV DRCSA,R0 ;DO GROUP 2  
MOV DRCSB,R1  
BR 1$
```



```

1453 011314 012723 011544      1$:  MOV      #TRPOUT,(R3)+ ;RESTORE APT SUB TRAP
1454                                ;IF ON APT
1455 011320 012713 000340      11$: MOV      #340,(R3) ;RESTORE PSW SAVE
1456 011324 062737 000004 002452 ADD      #4,VECLOC ;UPDATE FOR NEXT VECTOR
1457 011332 000205      2$:  RTS      R5 ; RETURN
1458
1459                                ;ROUTINE TO SET UP TRAP CATCHER 204-1774 IN STANDALONE MODE OR
1460                                ;A COMMON VECTOR ROUTINE SETUP UNDER APT.
1461                                ;VECTOR ROUTINE IS USED ONLY IN AN APT ENVIRONMENT AND WILL STORE
1462                                ;A COMMON SUBROUTINE IN VECTOR AREA 204-1774.
1463
1464 011334 010046      TRPCAT: MOV      R0,-(SP) ;SAVE R0
1465 011336 010146      MOV      R1,-(SP)
1466 011340 022737 000001 002210 CMP      #1,$ENV ;CHECK FOR APT
1467 011346 001426      BEQ      APTVEC ;YES,SET UP TRAPS FOR APT
1468 011350 012700 000204      MOV      #204,R0 ;START CATCHER AT 204
1469 011354 032777 010070 170556 BIT      #SW12,@SWR ;TEST IF SW12 IS SET
1470 011362 001010      BNE      1$ ;YES,BR AND STORE REGULAR TRAP
1471                                ;CATCHER(+2,HALT) IN LOCATIONS
1472                                ;204-1774
1473 011364 012720 011562      11$:  MOV      #TRPALL,(R0)+ ;NO,PLACE ILLEGAL VECTOR RETURN
1474 011370 012720 000340      MOV      #340,(R0)+ ;ROUTINES IN LOCATIONS 204-1774
1475 011374 022700 002000      CMP      #2000,R0 ;STORED ALL VECTORS?
1476 011400 001371      BNE      11$
1477 011402 000421      BR       ENDTRP ;RETURN AFTER VECTOR STORAGE
1478 011404 010001      1$:  MOV      R0,R1 ;DRV11J CAN VECTOR 0-1774
1479 011406 005721      TST      (R1)+
1480 011410 010120      MOV      R1,(R0)+
1481 011412 005020      CLR      (R0)+
1482 011414 022701 001775      CMP      #1776,R1 ;CHECK FOR VECTOR END
1483 011420 001371      BNE      1$
1484 011422 000411      BR       ENDTRP ;RETURN IN STAND ALONE MODE
1485 011424 012701 000204      APTVE . MOV      #204,R1 ;STARTING VECTOR ADDRESS
1486 011430 012721 011544      1$:  MOV      #TRPOUT,(R1)+ ;ILL INTERRUPT ROUTINE
1487 011434 012721 000340      MOV      #340,(R1)+ ;PSW NEXT
1488 011440 022701 002000      CMP      #2000,R1 ;DONE?
1489 011444 001371      BNE      1$ ;DO NEXT VECTOR
1490 011446 012601      ENDTRP: MOV      (SP)+,R1
1491 011450 012600      MOV      (SP)+,R0
1492 011452 000205      RTS      R5 ;EXIT
1493
1494                                ;INTERP'T SERVICE ROUTINE USED TO VERIFY INTERRUPTS 204-1774
1495 011454 005237 002436      INTSR1 INC      INTFLG ;COUNT INTERRUPT
1496 011460 012737 000001 002124 MOV      #1,$GDDAT ;STORE EXPECTED
1497 011466 013737 002436 002126 MOV      INTFLG,$BDDAT ;SAVE INT. COUNT
1498 011474 023737 002124 002126 CMP      $GDDAT,$BDDAT ;SHOULD BE ONE INTERRUPT
1499 011502 001401      BEQ      1$
1500 011504 104013      ERROR 13 ;MULTIPLE INTERRUPTS RECEIVED
1501 011506 000002      1$:  RTI ;RETURN FROM INT.
1502
1503                                ;INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 204-1774
1504                                ;FOR A BYTE COUNT OF 1 TO 4 VECTORS IN VECTOR ADDR MEMORY.
1505 011510 005237 002436      INTBY4: INC      INTFLG ;COUNT INTERRUPT
1506 011514 013737 002472 002124 MOV      BYNUM,$GDDAT ;STORE EXPECTED
1507 011522 013737 002436 002126 MOV      INTFLG,$BDDAT ;SAVE INT. COUNT
1508 011530 023737 002124 002126 CMP      $GDDAT,$BDDAT ;INTERRUPTS SHOULD NOT EXCEED BYTE COUNT

```

```

1509 011536 002001          BGE      1$
1510 011540 104013          ERROR    13          ;MULTIPLE INTERRUPTS RECEIVED
1511                                     ;MORE INTERRUPTS THAN BYTE COUNT
1512 011542 000002          1$:      RTI          ;RETURN FROM INT.
1513
1514                                     ;COMMON ILLEGAL VECTOR RETURN ROUTINE FOR APT MODE.
1515 011544 011637 002120  TRP0UT: MOV      (SP), $GDADR      ;SAVE ADDRESS OF TEST
1516 011550 004537 011204          JSR      R5, FRMBUF      ;RETURN LOCATIONS 0-202
1517 011554 104014          ERROR    14          ;ILLEGAL VECTOR ADDR MEM ERROR
1518 011556 000000          HALT
1519 011560 000000          HALT          ;CANNOT CONTINUE
1520
1521                                     ;COMMON ILLEGAL VECTOR RETURN ROUTINE FOR VECTOR AREA
1522                                     ;204-1774
1523 011562 011637 002120  TRPALL: MOV      (SP), $GDADR      ;SAVE PC OF TEST
1524 011566 104014          ERROR    14          ;ILLEGAL VECTOR ADDR MEM ERROR
1525 011570 000002          RTI          ;RETURN
1526
1527                                     ;RESTORE VECTOR JUST TESTED TO ERROR SUBROUTINE FOR 0-202
1528 011572 022737 000001 002210 RES200: CMP      #1, $ENV          ;CHECK FOR APT
1529 011600 001405          BEQ      1$          ;YES, BRANCH
1530 011602 012723 012000          MOV      #TRP200, (R3)+      ;ERROR TRAP SUB. FOR 0-200
1531 011606 012713 000340          MOV      #340, (R3)         ;PSW
1532 011612 000404          BR       2$          ;RETURN
1533 011614 012723 011544          1$:      MOV      #TRP0UT, (R3)+  ;RESTORE APT SUB TRAP
1534                                     ;IF ON APT
1535 011620 012713 000340          MOV      #340, (R3)         ;RESTORE PSW SAVE
1536 011624 062737 000004 002452 2$:      ADD      #4, VECLOC      ;UPDATE FOR NEXT VECTOR
1537 011632 000205          RTS       R5          ;RETURN
1538
1539                                     ;ROUTINE TO SET UP TRAP SUBROUTINE 0-200 IN STANDALONE MODE OR
1540                                     ;A COMMON ERROR VECTOR ROUTINE SETUP UNDER APT.
1541
1542 011634 010046          CAT200: MOV      R0, -(SP)          ;SAVE R0
1543 011636 017737 170276 002460 MOV      @SWR, SWRSV          ;SAVE LOCATION 176, SOFT SWR
1544 011644 022737: 000001 002210 CMP      #1, $ENV          ;CHECK FOR APT
1545 011652 001411          BEQ      APT200          ;YES, SET UP TRAPS FOR APT
1546 011654 005000          CLR      R0          ;START CATCHER SUB AT 0
1547 011656 012720 012000          1$:      MOV      #TRP200, (R0)+      ;STORE VECTOR TRAP SUBROUTINE
1548 011662 012720 000340          MOV      #340, (R0)+      ;PSW
1549 011666 022700 000204          CMP      #204, R0          ;FINISHED 0-202
1550 011672 001371          BNE      1$
1551 011674 000410          BR       END200          ;RETURN IN STAND ALONE MODE
1552 011676 005000          APT200: CLR      R0          ;START AT VECTOR 0
1553 011700 012720 011544          1$:      MOV      #TRP0UT, (R0)+      ;VECTOR ERROR ROUTINE
1554 011704 012720 000340          MOV      #340, (R0)+      ;PSW NEXT
1555 011710 022700 000204          CMP      #204, R0          ;DONE?
1556 011714 001371          BNE      1$          ;DO NEXT VECTOR
1557 011716 012600          END200: MOV      (SP)+, R0
1558 011720 012713 011734          MOV      #INTSR3, (R3)      ;STORE VECTOR ROUTINE
1559 011724 012763 000340 000002 MOV      #340, 2(R3)        ;STORE PSW
1560 011732 000205          RTS       R5          ;EXIT
1561
1562                                     ;INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 0-200, BYTE COUNT 0-3
1563                                     ;FOR BYTE COUNTS OF 1 TO 4 VECTORS IN VECTOR ADDRESS MEMORY.
1564

```

```

1565 011734 005237 002436      INTR3: INC      INTFLG      ;COUNT INTERRUPT
1566 011740 013737 002472 002124      MOV      BYNUM,$GDDAT ;STORE EXPECTED
1567 011746 013737 002436 002126      MOV      INTFLG,$BDDAT ;SAVE INT. COUNT
1568 011754 023737 002124 002126      CMP      $GDDAT,$BDDAT ;INTERRUPTS SHOULD NOT EXCEED BYTE COUNT
1569 011762 002000      BGE      '$'
1570 011764 004537 011204      JSR      R5,FRMBUF    ;RESTORE 0-202 BEFORE ERROR
1571 011770 104013      ERROR   13          ;MULTIPLE INTERRUPTS RECEIVED
1572                                ;MORE INTERRUPTS THAN BYTE COUNT
1573 011772 004537 011634      JSR      R5,CAT200    ;RESTORE VECTOR SUBROUTINES
1574 011776 000002      1$:     RTI          ;RETURN FROM INT.
1575
1576                                ;COMMON ILLEGAL VECTOR ERROR RETURN ROUTINE FOR 0-200
1577 012000 011637 002120      TRP200: MOV      (SP),$GDADR ;SAVE PC OF TEST
1578 012004 004537 011204      JSR      R5,FRMBUF    ;RETURN LOC 0-200
1579 012010 104014      ERROR   14          ;ILLEGAL VECTOR ADDR MEM ERROR
1580 012012 004537 011634      JSR      R5,CAT200    ;RESTORE VECTOR SUBROUTINES FOR PROCEED
1581 012016 000002      RTI          ;RETURN
1582
1583                                ;VECTOR SERVICE ROUTINE FOR VECTOR UNIQUENESS TEST
1584 012020 012524      SETVEC: MOV      (R5)+,(R4)+ ;STORE VALUE IN BUFFER
1585 012022 005726      TST      (SP)+        ;RESTORE STACK FOR RTI
1586 012024 005237 002436      INC      INTFLG      ;COUNT INTERRUPT
1587 012030 022737 000100 002436      CMP      #64.,INTFLG
1588 012036 002001      BGE      '$'          ;64 EXPECTED INTERRUPTS?
1589 012040 104013      ERROR   13          ;ERROR MORE THAN 64 INTERRUPTS
1590 012042 000002      1$:     RTI          ;RETURN FOR NEXT INTERRUPT
1591
1592                                ;INTERRUPT SERVICE ROUTINES THAT ARE STORED FROM 300-674
1593                                ;IN THE VECTOR UNIQUENESS TEST.
1594 012044 004537 012020      INTO:   JSR      R5,SETVEC ;STORE 300 INTO VBUF, TO BE
1595 012050 000300      300    ;CHECKED AFTER ALL INTERRUPTS.
1596 012052 004537 012020      INT1:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1597 012056 000304      304
1598 012060 004537 012020      INT2:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1599 012064 000310      310
1600 012066 004537 012020      INT3:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1601 012072 000314      314
1602 012074 004537 012020      INT4:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1603 012100 000320      320
1604 012102 004537 012020      INT5:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1605 012106 000324      324
1606 012110 004537 012020      INT6:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1607 012114 000330      330
1608 012116 004537 012020      INT7:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1609 012122 000334      334
1610 012124 004537 012020      INT8:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1611 012130 000340      340
1612 012132 004537 012020      INT9:   JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1613 012136 000344      344
1614 012140 004537 012020      INT10:  JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1615 012144 000350      350
1616 012146 004537 012020      INT11:  JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1617 012152 000354      354
1618 012154 004537 012020      INT12:  JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN
1619 012160 000360      360
1620 012162 004537 012020      INT13:  JSR      R5,SETVEC ;VECTOR UNIQUENESS TEST, INT. SERV RTN

```

1621	012166	000364			364		
1622	012170	004537	012020	INT14:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1623	012174	000370			370		
1624	012176	004537	012020	INT15:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1625	012202	000374			374		
1626	012204	004537	012020	INT16:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1627	012210	000400			400		
1628	012212	004537	012020	INT17:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1629	012216	000404			404		
1630	012220	004537	012020	INT18:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1631	012224	000410			410		
1632	012226	004537	012020	INT19:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1633	012232	000414			414		
1634	012234	004537	012020	INT20:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1635	012240	000420			420		
1636	012242	004537	012020	INT21:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1637	012246	000424			424		
1638	012250	004537	012020	INT22:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1639	012254	000430			430		
1640	012256	004537	012020	INT23:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1641	012262	000434			434		
1642	012264	004537	012020	INT24:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1643	012270	000440			440		
1644	012272	004537	012020	INT25:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1645	012276	000444			444		
1646	012300	004537	012020	INT26:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1647	012304	000450			450		
1648	012306	004537	012020	INT27:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1649	012312	000454			454		
1650	012314	004537	012020	INT28:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1651	012320	000460			460		
1652	012322	004537	012020	INT29:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1653	012326	000464			464		
1654	012330	004537	012020	INT30:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1655	012334	000470			470		
1656	012336	004537	012020	INT31:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1657	012342	000474			474		
1658	012344	004537	012020	INT32:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1659	012350	000500			500		
1660	012352	004537	012020	INT33:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1661	012356	000504			504		
1662	012360	004537	012020	INT34:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1663	012364	000510			510		
1664	012366	004537	012020	INT35:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1665	012372	000514			514		
1666	012374	004537	012020	INT36:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1667	012400	000520			520		
1668	012402	004537	012020	INT37:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1669	012406	000524			524		
1670	012410	004537	012020	INT38:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1671	012414	000530			530		
1672	012416	004537	012020	INT39:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1673	012422	000534			534		
1674	012424	004537	012020	INT40:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1675	012430	000540			540		
1676	012432	004537	012020	INT41:	JSR	R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN

1677	012436	000544		544		
1678	012440	004537	012020	INT42:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1679	012444	000550		550		
1680	012446	004537	012020	INT43:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1681	012452	000554		554		
1682	012454	004537	012020	INT44:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1683	012460	000560		560		
1684	012462	004537	012020	INT45:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1685	012466	000564		564		
1686	012470	004537	012020	INT46:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1687	012474	000570		570		
1688	012476	004537	012020	INT47:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1689	012502	000574		574		
1690	012504	004537	012020	INT48:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1691	012510	000600		600		
1692	012512	004537	012020	INT49:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1693	012516	000604		604		
1694	012520	004537	012020	INT50:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1695	012524	000610		610		
1696	012526	004537	012020	INT51:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1697	012532	000614		614		
1698	012534	004537	012020	INT52:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1699	012540	000620		620		
1700	012542	004537	012020	INT53:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1701	012546	000624		624		
1702	012550	004537	012020	INT54:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1703	012554	000630		630		
1704	012556	004537	012020	INT55:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1705	012562	000634		634		
1706	012564	004537	012020	INT56:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1707	012570	000640		640		
1708	012572	004537	012020	INT57:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1709	012576	000644		644		
1710	012600	004537	012020	INT58:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1711	012604	000650		650		
1712	012606	004537	012020	INT59:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1713	012612	000654		654		
1714	012614	004537	012020	INT60:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1715	012620	000660		660		
1716	012622	004537	012020	INT61:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1717	012626	000664		664		
1718	012630	004537	012020	INT62:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1719	012634	000670		670		
1720	012636	004537	012020	INT63:	JSR R5,SETVEC	;VECTOR UNIQUENESS TEST, INT. SERV RTN
1721	012642	000674		674		

1722						
1723				.SBITL	PATTERNS FOR REGISTER R/W	
1724				:		
1725				:	PATTERNS USED FOR LOADING/READING REGISTERS	
1726				:	ISR	INTERRUPT SERVICE REGISTER
1727				:	IRR	INTERRUPT REQUEST REGISTER
1728				:	IMR	INTERRUPT MASK REGISTER
1729				:		
1730	012644	001	376	BGCHP3:	.BYTE	1,376
1731	012646	002	375		.BYTE	2,375
1732	012650	004	373		.BYTE	4,373

CVDRDA DRV11J DIAG TST PRT2
CVDRDA.P11 21-OCT-79 13:19

MACY11 30A(1052) 21-OCT-79 13:38 ^{G 4} PAGE 36
PATTERNS FOR REGISTER R/W

PAGE: 0045

1733	012652	010	367	.BYTE	10,367
1734	012654	020	357	.BYTE	20,357
1735	012656	040	337	.BYTE	40,337
1736	012660	100	277	.BYTE	100,277
1737	012662	200	177	.BYTE	200,177
1738	012664	000000		EDCMP3:	000000
1739					
1740					

1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796

.SBTTL SYSMAC ROUTINES
.SBTTL TYPE ROUTINE

*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*

*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*

012666	105737	002157	\$TYPE:	TSTB	\$TPFLG	:: IS THERE A TERMINAL?
012672	100002			BFL	1\$:: BR IF YES
012674	000000			HALT		:: HALT HERE IF NO TERMINAL
012676	000430			BR	3\$:: LEAVE
012700	010046		1\$:	MOV	RO,-(SP)	:: SAVE RO
012702	017600	000002		MOV	@2(SP),RO	:: GET ADDRESS OF ASCIZ STRING
012706	122737	000001 002210		CMPB	#APTENV,\$ENV	:: RUNNING IN APT MODE
012714	001011			BNE	62\$:: NO,GO CHECK FOR APT CONSOLE
012716	132737	000100 002211		BITB	#APTPOOL,\$ENVM	:: SPOOL MESSAGE TO APT
012724	001405			BEQ	62\$:: NO,GO CHECK FOR CONSOLE
012726	010037	012736		MOV	RO,61\$:: SETUP MESSAGE ADDRESS FOR APT
012732	004737	013156		JSR	PC,\$ATY3	:: SPOOL MESSAGE TO APT
012736	000000		61\$:	.WORD	0	:: MESSAGE ADDRESS
012740	132737	000040 002211	62\$:	BITB	#APTCSUP,\$ENVM	:: APT CONSOLE SUPPRESSED
012746	001003			BNE	60\$:: YES,SKIP TYPE OUT
012750	112046		2\$:	MOVB	(RO)+,-(SP)	:: PUSH CHARACTER TO BE TYPED ONTO STACK
012752	001005			BNE	4\$:: BR IF IT ISN'T THE TERMINATOR
012754	005726			TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK
012756	012600		60\$:	MOV	(SP)+,RO	:: RESTORE RO
012760	062716	000002	3\$:	ADD	#2,(SP)	:: ADJUST RETURN PC
012764	000002			RTI		:: RETURN
012766	122716	000011	4\$:	CMPB	#HT,(SP)	:: BRANCH IF <HT>
012772	001430			BEQ	8\$	
012774	122716	000200		CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
013000	001006			BNE	5\$	
013002	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
013004	104401			TYPE		:: TYPE A CR AND LF
013006	002165			\$CRLF		
013010	105037	013144		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
013014	000755			BR	2\$:: GET NEXT CHARACTER
013016	004737	013100	5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
013022	123726	002156	6\$:	CMPB	\$FILLC,(SP)+	:: IS IT TIME FOR FILLER CHARS.?
013026	001350			BNE	2\$:: IF NO GO GET NEXT CHAR.
013030	013746	002154		MOV	\$NULL,-(SP)	:: GET # OF FILLER CHARS. NEEDED
						:: AND THE NULL CHAR.
013034	105366	000001	7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
013040	002770			BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK

```
1797 013042 004737 013100 JSR PC,$TYPEC ::GO TYPE A NULL
1798 013046 105337 013144 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
1799 013052 000770 BR 7$ ::LOOP
1800
1801 ;HORIZONTAL TAB PROCESSOR
1802
1803 013054 112716 000040 8$: MOVB #' (SP) ::REPLACE TAB WITH SPACE
1804 013060 004737 013100 9$: JSR PC,$TYPEC ::TYPE A SPACE
1805 013064 132737 000007 013144 BITB #7,$CHARCNT ::BRANCH IF NOT AT
1806 013072 001372 BNE 9$ ::TAB STOP
1807 013074 005726 TST (SP)+ ::POP SPACE OFF STACK
1808 013076 000724 BR 2$ ::GET NEXT CHARACTER
1809 013100 105777 167044 $TYPEC: TSTB @STPS ::WAIT UNTIL PRINTER IS READY
1810 013104 100375 BPL $TYPEC
1811 013106 116677 000002 167036 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
1812 013114 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
1813 013122 001003 BNE 1$ ::BRANCH IF NO
1814 013124 105037 013144 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT
1815 013130 000406 BR $TYPEX ::EXIT
1816 013132 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
1817 013140 001402 BEQ $TYPEX ::BRANCH IF YES
1818 013142 105227 INCB (PC)+ ::COUNT THE CHARACTER
1819 013144 000000 $CHARCNT: .WORD 0 ::CHARACTER COUNT STORAGE
1820 013146 000207 $TYPEX: RTS PC
```

.SBTTL APT COMMUNICATIONS ROUTINE

```
1821
1822
1823
1824 ;*****
1825 013150 112737 000001 013414 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
1826 013156 112737 000001 013412 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
1827 013164 000403 BR $ATYC
1828 013166 112737 000001 013414 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
1829 013174 $ATYC:
1830 013174 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
1831 013176 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
1832 013200 105737 013412 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
1833 013204 001450 BEQ 5$ ::IF NOT: BR
1834 013206 122737 000001 002210 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
1835 013214 001031 BNE 3$ ::IF NOT: BR
1836 013216 132737 000100 002211 BITB #ARTSPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
1837 013224 001425 BEQ 3$ ::IF NOT: BR
1838 013226 017600 000004 MOV @4(SP),R0 ::GET MESSAGE ADDR.
1839 013232 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
1840 013240 005737 002170 1$: TST $MSGTYPE ::SEE IF DONE W/ LAST XMISSION?
1841 013244 001375 BNE 1$ ::IF NOT: WAIT
1842 013246 010037 002204 MOV R0,$MSGAD ::PUT ADDR IN MAILBOX
1843 013252 105720 2$: TSTB (R0)+ ::FIND END OF MESSAGE
1844 013254 001376 BNE 2$
1845 013256 163700 002204 SUB $MSGAD,R0 ::SUB START OF MESSAGE
1846 013262 006200 ASR R0 ::GET MESSAGE LNTH IN WORDS
1847 013264 010037 002206 MOV R0,$MSGLGT ::PUT LENGTH IN MAILBOX
1848 013270 012737 000004 002170 MOV #4,$MSGTYPE ::TELL APT TO TAKE MSG.
1849 013276 000413 BR 5$
1850 013300 017637 000004 013324 3$: MOV @4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
1851 013306 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
1852 013314 013746 177776 MOV 177776,-(SP) ::PUSH 177776 ON STACK
```


1853 013320 004737 012666
1854 013324 000000
1855 013326
1856 013326 105737 013414
1857 013332 001416
1858 013334 005737 002210
1859 013340 001413
1860 013342 005737 002170
1861 013346 001375
1862 013350 017637 000004 002172
1863 013356 062766 000002 000004
1864 013364 005237 002170
1865 013370 105037 013414
1866 013374 105037 013413
1867 013400 105037 013412
1868 013404 012601
1869 013406 012600
1870 013410 000207
1871 013412 000
1872 013413 000
1873 013414 000
1874 013416
1875 000200
1876 000001
1877 000100
1878 000040
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904 013416 017646 000000
1905 013422 116637 000001 013641
1906 013430 112637 013643
1907 013434 062716 000002
1908 013440 000406

```
JSR PC,$TYPE      ;;CALL TYPE MACRO
4$: .WORD 0
5$:
10$: TSTB $FFLG      ;;SHOULD REPORT FATAL ERROR?
    BEQ 12$          ;;IF NOT: BR
    TST $ENV         ;;RUNNING UNDER APT?
    BEQ 12$          ;;IF NOT: BR
11$: TST $MSGTYPE    ;;FINISHED LAST MESSAGE?
    BNE 11$          ;;IF NOT: WAIT
    MOV @4(SP),$FATAL ;;GET ERROR #
    ADD #2,4(SP)     ;;BUMP RETURN ADDR.
    INC $MSGTYPE     ;;TELL APT TO TAKE ERROR
12$: CLRB $FFLG      ;;CLEAR FATAL FLAG
    CLRB $LFLG       ;;CLEAR LOG FLAG
    CLRB $MFLG       ;;CLEAR MESSAGE FLAG
    MOV (SP)+,R1     ;;POP STACK INTO R1
    MOV (SP)+,R0     ;;POP STACK INTO R0
    RTS PC           ;;RETURN
$MFLG: .BYTE 0      ;;MESSG. FLAG
$LFLG: .BYTE 0      ;;LOG FLAG
$FFLG: .BYTE 0      ;;FATAL FLAG
```

APTSIZE=200
APTENV=001
APTSPOOL=100
APTSUP=040
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV NUM,-(SP)    ;;NUMBER TO BE TYPED
*   TYPOS            ;;CALL FOR TYPEOUT
*   .BYTE N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE M          ;;M 1 OR 0
*                       ;;1=TYPE LEADING ZEROS
*                       ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV NUM,-(SP)    ;;NUMBER TO BE TYPED
*   TYPON            ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV NUM,-(SP)    ;;NUMBER TO BE TYPED
*   TYPOC            ;;CALL FOR TYPEOUT
$TYPOS: MOV @2(SP),-(SP) ;;PICKUP THE MODE
        MOV# 1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
        MOV# (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD #2,(SP)      ;;ADJUST RETURN ADDRESS
        BR $TYPON
```

```
1909 013442 112737 000001 013641 $TYROC: MOVB #1,$OFILL      ::SET THE ZERO FILL SWITCH
1910 013450 112737 000006 013643      MOVB #6,$OMODE+1    ::SET FOR SIX(6) DIGITS
1911 013456 112737 000005 013640 $TYPON: MOVB #5,$OCNT  ::SET THE ITERATION COUNT
1912 013464 010346      MOV R3,-(SP)        ::SAVE R3
1913 013466 010446      MOV R4,-(SP)        ::SAVE R4
1914 013470 010546      MOV R5,-(SP)        ::SAVE R5
1915 013472 113704 013643      MOVB $OMODE+1,R4    ::GET THE NUMBER OF DIGITS TO TYPE
1916 013476 005404      NEG R4              ::
1917 013500 062704 000006      ADD #6,R4           ::SUBTRACT IT FOR MAX. ALLOWED
1918 013504 110437 013642      MOVB R4,$OMODE      ::SAVE IT FOR USE
1919 013510 113704 013641      MOVB $OFILL,R4      ::GET THE ZERO FILL SWITCH
1920 013514 016605 000012      MOV 12(SP),R5       ::PICKUP THE INPUT NUMBER
1921 013520 005003      CLR R3              ::CLEAR THE OUTPUT WORD
1922 013522 006105      ROL R5              ::ROTATE MSB INTO 'C'
1923 013524 000404      BR 3$               ::GO DO MSB
1924 013526 006105      ROL R5              ::FORM THIS DIGIT
1925 013530 006105      ROL R5              ::
1926 013532 006105      ROL R5              ::
1927 013534 010503      MOV R5,R3           ::
1928 013536 006103      ROL R3              ::GET LSB OF THIS DIGIT
1929 013540 105337 013642      DECB $OMODE         ::TYPE THIS DIGIT?
1930 013544 100016      BPL 7$              ::BR IF NO
1931 013546 042703 177770      BIC #177770,R3     ::GET RID OF 'JUNK'
1932 013552 001002      BNE 4$              ::TEST FOR 0
1933 013554 005704      TST R4              ::SUPPRESS THIS 0?
1934 013556 001403      BEQ 5$              ::BR IF YES
1935 013560 005204      INC R4              ::DON'T SUPPRESS ANYMORE '0'S
1936 013562 052703 000060      BIS #0,R3           ::MAKE THIS DIGIT ASCII
1937 013566 052703 000040      BIS #1,R3           ::MAKE ASCII IF NOT ALREADY
1938 013572 110337 013636      MOVB R3,8$         ::SAVE FOR TYPING
1939 013576 104401 013636      TYPE 8$            ::GO TYPE THIS DIGIT
1940 013602 105337 013640      DECB $OCNT         ::COUNT BY 1
1941 013606 003347      BGT 2$              ::BR IF MORE TO DO
1942 013610 002402      BLT 6$              ::BR IF DONE
1943 013612 005204      INC R4              ::INSURE LAST DIGIT ISN'T A BLANK
1944 013614 000744      BR 2$              ::GO DO THE LAST DIGIT
1945 013616 012605      MOV (SP)+,R5        ::RESTORE R5
1946 013620 012604      MOV (SP)+,R4        ::RESTORE R4
1947 013622 012603      MOV (SP)+,R3        ::RESTORE R3
1948 013624 016666 000002 000004      IMOV 2(SP),4(SP)   ::SET THE STACK FOR RETURNING
1949 013632 012616      MOV (SP)+,(SP)     ::
1950 013634 000002      RTI                 ::RETURN
1951 013636 000      8$: .BYTE 0         ::STORAGE FOR ASCII DIGIT/
1952 013637 000      .BYTE 0           ::TERMINATOR FOR TYPE ROUTINE
1953 013640 000      $OCNT: .BYTE 0    ::OCTAL DIGIT COUNTER
1954 013641 000      $OFILL: .BYTE 0   ::ZERO FILL SWITCH
1955 013642 000000      $OMODE: .WORD 0   ::NUMBER OF DIGITS TO TYPE
1956      .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
```

```
1957
1958
1959 *****
1960 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBFR TO A 5-DIGIT
1961 *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1962 *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1963 *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1964 *REPLACED WITH SPACES.
1965 *CALL:
```

```

1965      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
1966      TYPDS      ;;GO TO THE ROUTINE
1967
1968      013644      STYPDS:
1969      013644      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
1970      013646      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
1971      013650      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
1972      013652      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
1973      013654      010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
1974      013656      012746      020200      MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
1975      013662      016605      000C20      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
1976      013666      100004      BPL      1$      ;;BR IF INPUT IS POS.
1977      013670      005405      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
1978      013672      112766      000055      000001      MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1979      013700      005000      1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
1980      013702      012703      014060      MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
1981      013706      112723      000040      MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
1982      013712      005002      2$:      CLR      R2      ;;CLEAR THE BCD NUMBER
1983      013714      016001      014050      MOV      $DTBL(R0),R1      ;;GET THE CONSTANT
1984      013720      160105      3$:      SUB      R1,R5      ;;FORM THIS BCD DIGIT
1985      013722      002402      BLT      4$      ;;BR IF DONE
1986      013724      005202      INC      R2      ;;INCREASE THE BCD DIGIT BY 1
1987      013726      000774      BR       3$
1988      013730      060105      4$:      ADD      R1,R5      ;;ADD BACK THE CONSTANT
1989      013732      005702      TST      R2      ;;CHECK IF BCD DIGIT=0
1990      013734      001002      BNE      5$      ;;FALL THROUGH IF 0
1991      013736      105716      TSTB     (SP)      ;;STILL DOING LEADING 0'S?
1992      013740      100407      BMI      7$      ;;BR IF YES
1993      013742      106316      5$:      ASLB     (SP)      ;;MSD?
1994      013744      103003      BCC      6$      ;;BR IF NO
1995      013746      116663      000001      177777      MOVB     1(SP),-1(R3)      ;;YES--SET THE SIGN
1996      013754      052702      000060      6$:      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
1997      013760      052702      000040      7$:      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1998      013764      110223      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1999      013766      005720      TST      (R0)+      ;;JUST INCREMENTING
2000      013770      020027      000010      CMP      R0,#10      ;;CHECK THE TABLE INDEX
2001      013774      002746      BLT      2$      ;;GO DO THE NEXT DIGIT
2002      013776      003002      BGT      8$      ;;GO TO EXIT
2003      014000      010502      MOV      R5,R?      ;;GET THE LSD
2004      014002      000764      BR       6$      ;;GO CHANGE TO ASCII
2005      014004      105726      8$:      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
2006      014006      100003      BPL      9$      ;;BR IF NO
2007      014010      116663      177777      177776      MOVB     -1(SP),-2(R3)      ;;YES--SET THE SIGN FOR TYPING
2008      014016      105013      9$:      CLRB     (R3)      ;;SET THE TERMINATOR
2009      014020      012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
2010      014022      012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
2011      014024      012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
2012      014026      012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
2013      014030      012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
2014      014032      104401      014060      TYPE     ,SDBLK      ;;NOW TYPE THE NUMBER
2015      014036      016666      000002      000004      MOV      2(SP),4(SP)      ;;ADJUST THE STACK
2016      014044      012616      MOV      (SP)+,(SP)
2017      014046      000002      RTI      ;;RETURN TO USER
2018      014050      023420      $DTBL:  10000.
2019      014052      001750      1000.
2020      014054      000144      100.
  
```

```

2021 014056 000012
2022 014060 000004
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036 014070
2037 014070 104407
2038 014072 104407
2039 014074 105237 002103
2040 014100 001775
2041 014102 013777 002102 166032
2042 014110 005237 002112
2043 014114 011637 002116
2044 014120 162737 000002 002116
2045 014126 117737 165764 002114
2046 014134 032777 020000 165776
2047 014142 001004
2048 014144 004737 014244
2049 014150 104401 002165
2050 014154
2051 014154 122737 000001 002210
2052 014162 001007
2053 014164 113737 002114 014176
2054 014172 004737 013166
2055 014176 000
2056 014177 000
2057 014200 000777
2058 014202 005777 165732
2059 014206 100002
2060 014210 000000
2061 014212 104407
2062 014214 032777 001000 165716
2063 014222 001402
2064 014224 013716 002110
2065 014230 005737 002162
2066 014234 001402
2067 014236 013716 002162
2068 014242
2069 014242 000002
2070
2071
2072
2073
2074 014244 113737 002102 002432
2075 014252 004737 014262
2076 014256 104407

10.
$DBLK: .BLKW 4
.SBTTL ERROR HANDLER ROUTINE

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO SWRCK ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1 HALT ON ERROR
*SW13=1 INHIBIT ERROR TYPEOUTS
*SW09=1 LOOP ON ERROR
*CALL
* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
CKSWR ;GO LOOK FOR SWR CHANGE
7$: INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
INC $ERTTL ;;INC THE ERROR COUNT
MOV (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,$SWRCK ;;GO TO USER ERROR ROUTINE
TYPE $,CRLF

20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
BNE 2$ ;;NO,SKIP APT ERROR REPORT
MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
JSR PC,$SATY4 ;;REPORT FATAL ERROR TO APT

21$: .BYTE 0
.BYTE 0
22$: BR 22$ ;;APT ERROR LOOP
2$: TST @SWR ;;HALT ON ERROR
BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
BEQ 4$ ;;BR IF NO
MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
BEQ 5$ ;;BR IF NONE
MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE

5$: RTI ;;RETURN

*****
;GO TYPE ERROR
;GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
*****
SWRCK: MOVB $TSTNM,$TSTNUM ;;SET UP TEST # ON ER
JSR PC,$ERRTYP ;;GO TYPE ERROR
CKSWR ;;GO LOOK FOR SWR CHANGE
    
```

2077 014260 000207
2078
2079
2080
2081
2082
2083
2084
2085 014262
2086 014262 104401 002165
2087 014266 010046
2088 014270 005000
2089 014272 153700 002114
2090 014276 001004
2091
2092 014300 013746 002116
2093
2094 014304 104402
2095 014306 000426
2096 014310 005300
2097 014312 006300
2098 014314 006300
2099 014316 006300
2100 014320 062700 002252
2101 014324 012037 014334
2102 014330 001404
2103 014332 104401
2104 014334 000000
2105 014336 104401 002165
2106 014342 012037 014352
2107 014346 001404
2108 014350 104401
2109 014352 000000
2110 014354 104401 002165
2111 014360 011000
2112 014362 001004
2113 014364 012600
2114 014366 104401 002165
2115 014372 000207
2116 014374
2117 014374 013046
2118 014376 104402
2119 014400 005710
2120 014402 001770
2121 014404 104401 014412
2122 014410 000771
2123 014412 020040 000
2124 014416
2125
2126
2127
2128
2129
2130
2131
2132

```
RTS PC RETURN TO ERROR HANDLER
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*****
```

```
2133      ;*SW11=1      INHIBIT ITERATIONS
2134      ;*SW09=1      LOOP ON ERROR
2135      ;*SW08=1      LOOP ON TEST IN SWR<7:0>
2136      ;*CALL
2137      ;*          SCOPE          ;;SCOPE=10T
2138
2139      $SCOPE:
2140      014416 104407      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
2141      014420 032777 040000 165512 1$: BIT #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
2142      014426 001114      BNE $OVER          ;;YES IF SW14=1
2143      ;*****START OF CODE FOR THE XOR TESTER*****
2144      014430 000416      $XTSTR: BR 6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
2145      ;THIS INSTRUCTION TO A 'NOP' (NOP=240)
2146      014432 013746 000004      MOV @ERRVEC,-(SP)      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2147      014436 012737 014456 000004      MOV #5,$ERRVEC        ;;SET FOR TIMEOUT
2148      014444 005737 177060      TST @177060           ;;TIME OUT ON XOR?
2149      014450 012637 000004      MOV (SP)+,@ERRVEC     ;;RESTORE THE ERROR VECTOR
2150      014454 000463      BR $SVLAD             ;;GO TO THE NEXT TEST
2151      014456 022626      5$: CMP (SP)+,(SP)+    ;;CLEAR THE STACK AFTER A TIME OUT
2152      014460 012637 000004      MOV (SP)+,@ERRVEC     ;;RESTORE THE ERROR VECTOR
2153      014464 000423      BR 7$                 ;;LOOP ON THE PRESENT TEST
2154      014466      6$:;*****END OF CODE FOR THE XOR TESTER*****
2155      014466 032777 000400 165444      BIT #BIT08,@SWR       ;;LOOP ON SPEC. TEST?
2156      014474 001404      BEQ 2$                ;;BR IF NO
2157      014476 127737 165436 002102      CMPB @SWR,$STNM       ;;ON THE RIGHT TEST? SWR<7:0>
2158      014504 001465      BEQ $OVER             ;;BR IF YES
2159      014506 105737 002103      2$: TSTB $ERFLG        ;;HAS AN ERROR OCCURRED?
2160      014512 001421      BEQ 3$                ;;BR IF NO
2161      014514 123737 002115 002103      CMPB $ERMAX,$ERFLG    ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2162      014522 101015      BHI 3$                ;;BR IF NO
2163      014524 032777 001000 165406      BIT #BIT09,@SWR       ;;LOOP ON ERROR?
2164      014532 001404      BEQ 4$                ;;BR IF NO
2165      014534 013737 002110 002106      7$: MOV $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
2166      014542 000446      BR $OVER
2167      014544 105037 002103      4$: CLRB $ERFLG         ;;ZERO THE ERROR FLAG
2168      014550 005037 002160      CLR $TIMES            ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2169      014554 000415      BR 1$                 ;;ESCAPE TO THE NEXT TEST
2170      014556 032777 004000 165354      3$: BIT #BIT11,@SWR     ;;INHIBIT ITERATIONS?
2171      014564 001011      BNE 1$                ;;BR IF YES
2172      014566 005737 002176      TST $PASS            ;;IF FIRST PASS OF PROGRAM
2173      014572 001406      BEQ 1$                ;;INHIBIT ITERATIONS
2174      014574 005237 002104      INC $ICNT            ;;INCREMENT ITERATION COUNT
2175      014600 023737 002160 002104      CMP $TIMES,$ICNT     ;;CHECK THE NUMBER OF ITERATIONS MADE
2176      014606 002024      BGE $OVER            ;;BR IF MORE ITERATION REQUIRED
2177      014610 012737 000001 002104      1$: MOV #1,$ICNT       ;;REINITIALIZE THE ITERATION COUNTER
2178      014616 013737 014674 002160      MOV $MXCNT,$TIMES    ;;SET NUMBER OF ITERATIONS TO DO
2179      014624 105237 002102      $SVLAD: INCB $STNM     ;;COUNT TEST NUMBERS
2180      014630 113737 002102 002174      : MOVB $STNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
2181      014636 011637 002106      : MOV (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
2182      014642 011637 002110      : MOV (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
2183      014646 005037 002162      CLR $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
2184      014652 112737 000001 002115      : MOVB #1,$ERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2185      014660 013777 002102 165254      $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
2186      014666 013716 002106      : MOV $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
2187      014672 000002      RTI                  ;;FIXES PS
2188      014674 000001      $MXCNT: 1.           ;;MAX. NUMBER OF ITERATIONS
```

C 5

```
2189 .SBTTL TTY INPUT ROUTINE
2190
2191 ;:*****
2192 .ENABL LSB
2193
2194 ;:*****
2195 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2196 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2197 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2198 ;*WHEN OPERATING IN TTY FLAG MODE.
2199 014676 022737 000176 002140 $CKSWR: CMP #SWREG,SWR ;:IS THE SOFT-SWR SELECTED?
2200 014704 001074 BNE 15$ ;:BRANCH IF NO
2201 014706 105777 165232 TSTB @STKS ;:CHAR THERE?
2202 014712 100071 BPL 15$ ;:IF NO, DON'T WAIT AROUND
2203 014714 117746 165226 MOVB @STKB,-(SP) ;:SAVE THE CHAR
2204 014720 042716 177600 BIC #^C177,(SP) ;:STRIP-OFF THE ASCII
2205 014724 022726 000007 CMP #7,(SP)+ ;:IS IT A CONTROL G?
2206 014730 001062 BNE 15$ ;:NO, RETURN TO USER
2207 014732 123727 002134 000001 CMPB $AUTOB,#1 ;:ARE WE RUNNING IN AUTO-MODE?
2208 014740 001456 BEQ 15$ ;:BRANCH IF YES
2209
2210 014742 104401 015423 $GTSWR: TYPE ,SCNTLG ;:ECHO THE CONTROL-G (^G)
2211 014746 104401 015430 TYPE ,SMSWR ;:TYPE CURRENT CONTENTS
2212 014752 013746 000176 MOV SWREG,-(SP) ;:SAVE SWREG FOR TYPEOUT
2213 014756 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2214 014760 104401 015441 TYPE ,SMNEW ;:PROMPT FOR NEW SWR
2215 014764 005046 19$: CLR -(SP) ;:CLEAR COUNTER
2216 014766 005046 CLR -(SP) ;:THE NEW SWR
2217 014770 105777 165150 7$: TSTB @STKS ;:CHAR THERE?
2218 014774 100375 BPL 7$ ;:IF NOT TRY AGAIN
2219
2220 014776 117746 165144 MOVB @STKB,-(SP) ;:PICK UP CHAR
2221 015002 042716 177600 BIC #^C177,(SP) ;:MAKE IT 7-BIT ASCII
2222
2223
2224
2225 015006 021627 000025 9$: CMP (SP),#25 ;:IS IT A CONTROL-U?
2226 015012 001005 BNE 10$ ;:BRANCH IF NOT
2227 015014 104401 015416 TYPE ,SCNTLU ;:YES, ECHO CONTROL-U (^U)
2228 015020 062706 000006 20$: ADD #6,SP ;:IGNORE PREVIOUS INPUT
2229 015024 000757 BR 19$ ;:LET'S TRY IT AGAIN
2230
2231
2232 015026 021627 000015 10$: CMP (SP),#15 ;:IS IT A <CR>?
2233 015032 001022 BNE 16$ ;:BRANCH IF NO
2234 015034 005766 000004 TST 4(SP) ;:YES, IS IT THE FIRST CHAR?
2235 015040 001403 BEQ 11$ ;:BRANCH IF YES
2236 015042 016677 000002 165070 MOV 2(SP),@SWR ;:SAVE NEW SWR
2237 015050 062706 000006 11$: ADD #6,SP ;:CLEAR UP STACK
2238 015054 104401 002165 14$: TYPE ,SCRLF ;:ECHO <CR> AND <LF>
2239 015060 123727 002135 000001 CMPB $INTAG,#1 ;:RE-ENABLE TTY KBD INTERRUPTS?
2240 015066 001003 BNE 15$ ;:BRANCH IF NOT
2241 015070 012777 000100 165046 MOV #100,@STKS ;:RE-ENABLE TTY KBD INTERRUPTS
2242 015076 000002 15$: RTI ;:RETURN
2243 015100 004737 013100 16$: JSR PC,$TYPEC ;:ECHO CHAR
2244 015104 021627 000060 CMP (SP),#60 ;:CHAR < 0?
```

2245	015110	002420			BLT	18\$::BRANCH IF YES
2246	015112	021627	000067		CMP	(SP),#67	::CHAR > 7?
2247	015116	003015			BGT	18\$::BRANCH IF YES
2248	015120	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
2249	015124	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
2250	015130	001403			BEQ	17\$::BRANCH IF YES
2251	015132	006316			ASL	(SP)	::NO, SHIFT PRESENT
2252	015134	006316			ASL	(SP)	::CHAR OVER TO MAKE
2253	015136	006316			ASL	(SP)	::ROOM FOR NEW ONE.
2254	015140	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
2255	015144	056616	177776		BIS	-2(SP), (SP)	::SET IN NEW CHAR
2256	015150	000707			BR	7\$::GET THE NEXT ONE
2257	015152	104401	002164	18\$:	TYPE	\$QUES	::TYPE ?<CR><LF>
2258	015156	000720			BR	20\$::SIMULATE CONTROL-U
2259					.DSABL	LSB	

2260
2261
2262
2263
2264
2265
2266
2267
2268
2269

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

2270	015160	011646			SRDCHR: MOV	(SP),-(SP)	::PUSH DOWN THE PC	
2271	015162	016666	000004	000002	MOV	4(SP),2(SP)	::SAVE THE PS	
2272	015170	105777	164750	1\$:	TSTB	@STKS	::WAIT FOR	
2273	015174	100375			BPL	1\$::A CHARACTER	
2274	015176	117766	164744	000004	MOVB	@STKB,4(SP)	::READ THE TTY	
2275	015204	042766	177600	000004	BIC	#^(<177>,4(SP)	::GET RID OF JUNK IF ANY	
2276	015212	026627	000004	000023	CMP	4(SP),#23	::IS IT A CONTROL-S?	
2277	015220	001013			BNE	3\$::BRANCH IF NO	
2278	015222	105777	164716	2\$:	TSTB	@STKS	::WAIT FOR A CHARACTER	
2279	015226	100375			BPL	2\$::LOOP UNTIL ITS THERE	
2280	015230	117746	164712		MOVB	@STKB,-(SP)	::GET CHARACTER	
2281	015234	042716	177600		BIC	#^C177,(SP)	::MAKE IT 7-BIT ASCII	
2282	015240	022627	000021		CMP	(SP)+,#21	::IS IT A CONTROL-Q?	
2283	015244	001366			BNE	2\$::IF NOT DISCARD IT	
2284	015246	000750			BR	1\$::YES, RESUME	
2285	015250	026627	000004	000140	3\$:	CMP	4(SP),#140	::IS IT UPPER CASE?
2286	015256	002407			BLT	4\$::BRANCH IF YES	
2287	015260	026627	000004	000175	CMP	4(SP),#175	::IS IT A SPECIAL CHAR?	
2288	015266	003003			BGT	4\$::BRANCH IF YES	
2289	015270	042766	000040	000004	BIC	#40,4(SP)	::MAKE IT UPPER CASE	
2290	015276	000002			4\$:	RTI	::GO BACK TO USER	

2291
2292
2293
2294
2295
2296
2297

```
*****  
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY  
*CALL:  
*RDLIN  
*RETURN HERE  
*::INPUT A STRING FROM THE TTY  
*::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK  
*::TERMINATOR WILL BE A BYTE OF ALL 0'S
```

2298	015300	010346			SRDLIN: MOV	R3,-(SP)	::SAVE R3
2299	015302	012703	015406	1\$:	MOV	#TTYIN,R3	::GET ADDRESS
2300	015306	022703	015416	2\$:	CMP	#TTYIN+8,R3	::BUFFER FULL


```

2301 015312 101405      BLOS 4$      ;;BR IF YES
2302 015314 104410      RDCHR      ;;GO READ ONE CHARACTER FROM THE TTY
2303 015316 112613      MOVB (SP)+,(R3) ;;GET CHARACTER
2304 015320 122713 000177 10$: CMPB #177,(R3) ;;IS IT A RUBOUT
2305 015324 001003      BNE 3$      ;;SKIP IF NOT
2306 015326 104401 002164 4$: TYPE ,SQUES ;;TYPE A '?'
2307 015332 000763      BR 1$      ;;CLEAR THE BUFFER AND LOOP
2308 015334 111337 015404 3$: MOVB (R3),9$ ;;ECHO THE CHARACTER
2309 015340 104401 015404      TYPE ,9$
2310 015344 122723 000015      CMPB #15,(R3)+ ;;CHECK FOR RETURN
2311 015350 001356      BNE 2$      ;;LOOP IF NOT RETURN
2312 015352 105063 177777      CLRB -1(R3) ;;CLEAR RETURN (THE 15)
2313 015356 104401 002166      TYPE ,SLF ;;TYPE A LINE FEED
2314 015362 012603      MOV (SP)+,R3 ;;RESTORE R3
2315 015364 011646      MOV (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2316 015366 016666 000004 000002      MOV 4(SP),2(SP) ;;FIRST ASCII CHARACTER ON I
2317 015374 012766 015406 000004      MOV #TTYI?,4(SP)
2318 015402 000002      RTI ;;RETURN
2319 015404 000 9$: .BYTE 0 ;;STORAGE FOR ASCII CHAR. TO TYPE
2320 015405 000 .BYTE 0 ;;TERMINATOR
2321 015406 000010      $TTYIN: .BLKB 8. ;;RESERVE 8 BYTES FOR TTY INPUT
2322 015416 052536 005015 000 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
2323 015423 136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
2324 015430 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR - /
2325 015436 020075 000
2326 015441 040 047040 Q53505 $MNEW: .ASCIZ / NEW - /
2327 015446 036440 000040
    
```

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
2331
2332 015452 012737 015616 000024 $PWRDN: MOV #SILLUP,@PWRVEC ;;SET FOR FAST UP
2333 015460 012737 000340 000026      MOV #340,@PWRVEC+2 ;;PRIO:7
2334 015466 010046      MOV R0,-(SP) ;;PUSH R0 ON STACK
2335 015470 010146      MOV R1,-(SP) ;;PUSH R1 ON STACK
2336 015472 010246      MOV R2,-(SP) ;;PUSH R2 ON STACK
2337 015474 010346      MOV R3,-(SP) ;;PUSH R3 ON STACK
2338 015476 010446      MOV R4,-(SP) ;;PUSH R4 ON STACK
2339 015500 010546      MOV R5,-(SP) ;;PUSH R5 ON STACK
2340 015502 017746 164432      MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
2341 015506 010637 015622      MOV SP,$SAVR6 ;;SAVE SP
2342 015512 012737 015524 000024      MOV #SPWRUP,@PWRVEC ;;SET UP VECTOR
2343 015520 000000      HALT
2344 015522 000776      BR -2 ;;HANG UP
    
```

```

*****
:POWER UP ROUTINE
2347
2348 015524 012737 015616 000024 $PWRUP: MOV #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
2349 015532 013706 015622      MOV $SAVR6,SP ;;GET SP
2350 015536 005037 015622      CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
2351 015542 005237 015622 1$: INC $SAVR6 ;;WAIT FOR THE INC
2352 015546 001375      BNE 1$      ;;OF WORD
2353 015550 012677 164364      MOV (SP)+,@SWR ;;POP STACK INTO @SWR
2354 015554 012605      MOV (SP)+,R5 ;;POP STACK INTO R5
2355 015556 012604      MOV (SP)+,R4 ;;POP STACK INTO R4
2356 015560 012603      MOV (SP)+,R3 ;;POP STACK INTO R3
    
```

```
2357 015562 012602      MOV      (SP)+,R2      :: POP STACK INTO R2
2358 015564 012601      MOV      (SP)+,R1      :: POP STACK INTO R1
2359 015566 012600      MOV      (SP)+,R0      :: POP STACK INTO R0
2360 015570 012737 015452 000024      MOV      #SPWRDN,@PWRVEC :: SET UP THE POWER DOWN VECTOR
2361 015576 012737 000340 000026      MOV      #340,@PWRVEC+2 :: PRIO:7
2362 015604 104401      TYPE      :: REPORT THE POWER FAILURE
2363 015606 015624      $PWRMG: .WORD  PWRMSG      :: POWER FAIL MESSAGE POINTER
2364 015610 012716      MOV      (PC)+,(SP)    :: RESTART AT START1
2365 015612 003066      $PWRAD: .WORD  START1    :: RESTART ADDRESS
2366 015614 000002      RTI
2367 015616 000000      $ILLUP: HALT          :: THE POWER UP SEQUENCE WAS STARTED
2368 015620 000776      BR      .-2           :: BEFORE THE POWER DOWN WAS COMPLETE
2369 015622 000000      $$SAVR6: 0           :: PUT THE SP HERE
2370 015624 005015 042522 052123      PWRMSG: .ASCIZ <-15><12>/RESTARTED FROM PWR FAIL/
2371 015632 051101 042524 020104
2372 015640 051106 046517 050040
2373 015646 051127 043040 044501
2374 015654 000114
2375
2376      .SBTTL TRAP DECODER
2377
2378      ::*****
2379      ::*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
2380      ::*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2381      ::*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2382      ::*GO TO THAT ROUTINE.
2383
2384 015656 010046      $TRAP: MOV      R0,-(SP)      :: SAVE R0
2385 015660 016600 000002      MOV      2(SP),R0        :: GET TRAP ADDRESS
2386 015664 005740      TST     -(R0)           :: BACKUP BY 2
2387 015666 111000      MOV     (R0),R0        :: GET RIGHT BYTE OF TRAP
2388 015670 006300      ASL    R0              :: POSITION FOR INDEXING
2389 015672 016000 015712      MOV     $TRPAD(R0),R0   :: INDEX TO TABLE
2390 015676 000200      RTS     R0             :: GO TO ROUTINE
2391
2392
2393      :: THIS IS USE TO HANDLE THE 'GETPRI' MACRO
2394
2395 015700 011646      $TRAP2: MOV     (SP),-(SP)  :: MOVE THE PC DOWN
2396 015702 016666 000004 000002      MOV     4(SP),2(SP)     :: MOVE THE PSW DOWN
2397 015710 003002      RTI                    :: RESTORE THE PSW
2398
2399      .SBTTL TRAP TABLE
2400
2401      :: THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2402      :: BY THE 'TRAP' INSTRUCTION.
2403
2404      ROUTINE
2405      -----
2406 015712 015700      $TRPAD: .WORD  $TRAP2
2407 015714 012666      $TYPE  :: CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
2408 015716 013442      $TYPOC :: CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2409 015720 013416      $TYPOS :: CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
2410 015722 013456      $TYPON :: CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
2411 015724 013644      $TYPDS :: CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
2412
```

2413	015726	014746			\$GTSWR	::CALL=GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING
2414								
2415	015730	014676			\$CKSWR	::CALL=EKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR
2416	015732	015160			\$RDCHR	::CALL=RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE
2417	015734	015300			\$RDLIN	::CALL=RDLIN	TRAP+11(104411)	TTY TYPEIN STRING ROUTINE
2418								
2419					.SBTTL	ASCII MESSAGES		
2420	015736	005015	053103	051104	TITLED:	.ASCIZ	<15><12>/CVDRDA	DRV11J DIAG TEST PART 2/<15><12>
2421	015744	040504	020040	051104				
2422	015752	030526	045061	042040				
2423	015760	040511	020107	042524				
2424	015766	052123	050040	051101				
2425	015774	020124	006462	000012				
2426	016002	005015	051104	030526	TLCABL:	.ASCIZ	15><12>/DRV11J	CABLE REQ'D/<15><12>
2427	016010	045061	041440	041101				
2428	016016	042514	051040	050505				
2429	016024	042047	005015	000				
2430								
2431	016031	122	043505	052040	EM1:	.ASCIZ	/REG TIMEOUT ER/	
2432	016036	046511	047505	052125				
2433	016044	042440	000122					
2434	016050	042522	020107	042522	EM2:	.ASCIZ	'REG READ/WRITE ER'	
2435	016056	042101	053457	044522				
2436	016064	042524	042440	000122				
2437	016072	051111	020122	042522	EM3:	.ASCIZ	/IRR REG ER/	
2438	016100	020107	051105	000				
2439	016105	101	051103	051040	EM4:	.ASCIZ	/ACR REG ER/	
2440	016112	043505	042440	000122				
2441	016120	046511	020122	042522	EM5:	.ASCIZ	/IMR REG ER/	
2442	016126	020107	051105	000				
2443	016133	111	051123	051040	EM6:	.ASCIZ	/ISR REG ER/	
2444	016140	043505	042440	000122				
2445	016146	046111	042514	040507	EM7:	.ASCIZ	/ILLEGAL VECTOR MEM ADDR ER/	
2446	016154	020114	042526	052103				
2447	016162	051117	046440	046505				
2448	016170	040440	042104	020122				
2449	016176	051105	000					
2450	016201	103	044510	020120	EM10:	.ASCIZ	/CHIP STAT ER/	
2451	016206	052123	052101	042440				
2452	016214	000122						
2453	016216	046111	042514	040507	EM11:	.ASCIZ	/ILLEGAL INTERRUPT RECEIVED/	
2454	016224	020114	047111	042524				
2455	016232	051122	050125	020124				
2456	016240	042522	042503	053111				
2457	016246	042105	000					
2458	016251	111	052116	051105	EM12:	.ASCIZ	/INTERRUPT TEST ERROR/	
2459	016256	052522	052120	052040				
2460	016264	051505	020124	051105				
2461	016272	047522	000122					
2462	016276	052515	052114	050111	EM13:	.ASCIZ	/MULTIPLE INTERRUPTS RECEIVED/	
2463	016304	042514	044440	052116				
2464	016312	051105	052522	052120				
2465	016320	020123	042522	042503				
2466	016326	053111	042105	000				
2467	016333	126	041505	020124	EM14:	.ASCIZ	/VECT ADDR MEM ER/	
2468	016340	042101	051104	046440				

2469	016346	046505	042440	000122					
2470	016354	051105	050122	020103	DM1:	.ASCIZ	/ERRPC	TSTNUM	BUSADR EXPCT RCVD/
2471	016362	020040	051524	047124					
2472	016370	046525	020040	052502					
2473	016376	040523	051104	020040					
2474	016404	054105	041520	020124					
2475	016412	020040	041522	042126					
2476	016420	000							
2477	016421	105	051122	041520	DM2:	.ASCIZ	/ERRPC	TSTNUM	BUSADR VAM ADDR EXPCT RCVD/
2478	016426	020040	052040	052123					
2479	016434	052516	020115	041040					
2480	016442	051525	042101	020122					
2481	016450	053040	046501	020040					
2482	016456	020040	040440	042104					
2483	016464	020122	020040	042440					
2484	016472	050130	052103	020040					
2485	016500	051040	053103	000104					
2486	016506	051105	050122	020103	DM3:	.ASCIZ	/ERRPC	TSTNUM	BUSADR VAM ADDR/
2487	016514	020040	051524	047124					
2488	016522	046525	020040	052502					
2489	016530	040523	051104	020040					
2490	016536	040526	020115	020040					
2491	016544	020040	042101	051104					
2492	016552	000							
2493	016553	105	051122	041520	DM4:	.ASCIZ	/ERRPC	TSTNUM	TESTPC BUSADR VAM ADDR/
2494	016560	052011	052123	052516					
2495	016566	004515	042524	052123					
2496	016574	041520	041011	051525					
2497	016602	042101	020122	053040					
2498	016610	046501	020040	020040					
2499	016616	040440	042104	000122					
2500									
2501						.EVEN			
2502	016624	002116	002432	002122	DT1:	\$ERRPC,	TSTNUM,	\$BDADR,	\$GDDAT,\$BDDAT,0
2503	016632	002124	002126	000000					
2504	016640	002116	002432	002122	DT2:	\$ERRPC,	TSTNUM,	\$BDADR,	VECVAR,VECLOC,\$GDDAT,\$BDDAT,0
2505	016646	002456	002452	002124					
2506	016654	002126	000000						
2507	016660	002116	002432	002122	DT3:	\$ERRPC,	TSTNUM,	\$BDADR,	VECVAR,VECLOC,0
2508	016666	002456	002452	000000					
2509	016674	002116	002432	002120	DT4:	\$ERRPC,	TSTNUM,	\$GDDADR,	\$BDADR,VECVAR,VECLOC,0
2510	016702	002122	002456	002452					
2511	016710	000000							
2512									
2513									
2514									
2515									
2516									
2517									
2518									
2519	016712	000102			DBUF:	.BLKW	66.		:1ST ADRS OF DATA STORAGE AREA
2520									
2521									
2522									
2523									
2524									

```
*****  
: 'DBUF' IS THE STORAGE AREA FOR SYSMAC LOCATIONS 0-202  
: PRESERVED IN ORDER TO TEST DRV11J VECTORING TO  
: VECTOR SPACE 0-200.  
:*****  
DBUF: .BLKW 66. :1ST ADRS OF DATA STORAGE AREA  
:*****  
: VECTOR BUFFER USED FOR VECTOR UNIQUENESS TEST  
: FOR 64 INTERRUPT VECTORS.  
:*****
```

CVDRDA DRV11J DIAG TST PRT2
CVDRDA.P11 21-OCT-79 13:19

MACY11 30A(1052) 21-OCT-79 13:38 I 5
ASCII MESSAGES

PAGE: 0060

2525 017116 000100
2526 017316
2527 000001

VBUF: .BLKW 64.
ENDBUF: .END

17

INT17	012212	1628#		
INT18	012220	1630#		
INT19	012226	1632#		
INT2	012060	1598#		
INT20	012234	1634#		
INT21	012242	1636#		
INT22	012250	1638#		
INT23	012256	1640#		
INT24	012264	1642#		
INT25	012272	1644#		
INT26	012300	1646#		
INT27	012306	1648#		
INT28	012314	1650#		
INT29	012322	1652#		
INT3	012066	1600#		
INT30	012330	1654#		
INT31	012336	1656#		
INT32	012344	1658#		
INT33	012352	1660#		
INT34	012360	1662#		
INT35	012366	1664#		
INT36	012374	1666#		
INT37	012402	1668#		
INT38	012410	1670#		
INT39	012416	1672#		
INT4	012074	1602#		
INT40	012424	1674#		
INT41	012432	1676#		
INT42	012440	1678#		
INT43	012446	1680#		
INT44	012454	1682#		
INT45	012462	1684#		
INT46	012470	1686#		
INT47	012476	1688#		
INT48	012504	1690#		
INT49	012512	1692#		
INT5	012102	1604#		
INT50	012520	1694#		
INT51	012526	1696#		
INT52	012534	1698#		
INT53	012542	1700#		
INT54	012550	1702#		
INT55	012556	1704#		
INT56	012564	1706#		
INT57	012572	1708#		
INT58	012600	1710#		
INT59	012606	1712#		
INT6	012110	1606#		
INT60	012614	1714#		
INT61	012622	1716#		
INT62	012630	1718#		
INT63	012636	1720#		
INT7	012116	1608#		
INT8	012124	1610#		
INT9	012132	1612#		
IOTVEC-	000020	128#	446*	447*

IRRLOC	002446	424#	750*	783	839*	873*	908	979*						
ISRLOC	002444	423#	751*	827	840*									
LF	= 000012	34#	1816	1822										
LMD04	= 000200	159#												
LMD57	= 000240	160#												
LVL CNT	002466	432#	748*	842*	871*	981*	1381*	1388*						
LVLSAV	002474	435#	1152*	1184*	1187*	1198*								
MACR	= 000254	166#	606	717	821	956	1086	1110	1233	1255				
MIMR	= 000244	164#	594	705	809	940	1075	1099	1223	1244				
MIRR	= 000250	165#	600	711	815	948	1080	1105	1228	1250				
MISR	= 000240	163#	588	613	699	803	828	932	965	1070	1094	1218	1239	
NEXPAS	003132	515	517#	1301										
NEXPA1	003136	518#	521											
NEXPA2	003202	526	528#											
NXDEV	010466	1293#												
NXDEV1	010472	516	1294#	1300										
PACR	= 000300	169#	665	1022	1034									
PIMR	= 000260	170#												
PIRQ	= 177772	40#												
PIRQVE	= 000240	134#												
PRO	= 000000	57#	554*	569*	673*	685*	777*	789*	900*	916*	1026*	1037*	1045*	1156*
		1166*	1174*	1189*										
PR1	- 000040	58#												
PR2	- 000100	59#												
PR3	= 000140	60#												
PR4	= 000200	61#												
PR5	- 000240	62#												
PR6	000300	63#												
PR7	= 000340	64#	528*	564*	576*	680*	695*	784*	798*	909*	927*	1040*	1049*	1169*
		1178*	1192*	1348*										
PS	- 177776	37#	38											
PSW	= 177776	38#												
PVMA	= 000340	171#	548	667	746	869	1349							
PWRMSG	015624	2363	2370#											
PWRVEC	= 000024	129#	452*	453*	2332*	2333*	2342*	2348*	2360*	2361*				
RDCHR	= 1C4410	2302	2416#											
RDLIN	= 104411	2417#												
RDY	= 100000	137#												
RESTRP	011242	833	1442#											
RESVEC	= 000010	124#												
RES200	011572	972	1528#											
SETVEC	012020	1584#	1594	1596	1598	1600	1602	1604	1606	1608	1610	1612	1614	1616
		1618	1620	1622	1624	1626	1628	1630	1632	1634	1636	1638	1640	1642
		1644	1646	1648	1650	1652	1654	1656	1658	1660	1662	1664	1666	1668
		1670	1672	1674	1676	1678	1680	1682	1684	1686	1688	1690	1692	1694
		1696	1698	1700	1702	1704	1706	1708	1710	1712	1714	1716	1718	1720
SIMR	= 000060	147#	551	670	775	898								
SIRR	= 000120	152#	563	679	1027	1038	1157	1167	1183	1197				
SSIMR	= 000070	148#												
SSIRR	= 000130	153#	750	873										
STACK	- 001100	28#												
START	002476	183	437#											
START1	003066	507	509#	1336	2365									
STK LMT	= 177774	39#												
STRVEC	011110	1010	1140	1400#										
SWR	002140	250#	442	463*	465	471*	478*	1436*	1444	1469	1543	2046	2058	2062

.SDIV	1#	
.SEOP	1#	1303
.SERRO	1#	2023
.SERRT	1#	2078
.SMULT	1#	
.SPOWF	1#	2328
.SRAND	1#	
.SRDDE	1#	
.SRDOC	1#	
.SREAD	1#	2189
.SR2AZ	1#	
.SSAVE	1#	
.SSB2D	1#	
.SSB2O	1#	
.SSCOP	1#	2125
.SSIZE	1#	
.SSUPR	1#	
.STRAP	1#	2376
.STYPB	1#	
.STYPD	1#	1956
.STYPE	1#	1743
.STYPO	1#	1879
.S4OCA	1#	
.117C	1#	

. ABS. 017316 000

ERRORS DETECTED: 0

DSKZ:CVDRDA,DSKZ:CVDRDA/CRF/SOL-DSKZ:SYSMAC.SML,DSKZ:CVDRDA.P11
RUN-TIME: 41 41 3 SECONDS
RUN-TIME RATIO: 145/86=1.6
CORE USED: 33K (65 PAGES)