

DRV11J

DRV11J DIAG TST PRT 1
CVDRCA0

AH-F758A-MC
COPYRIGHT 1980
FICHE 1 OF 1

JAN 1980
digital
MADE IN USA

The microfiche card contains 24 frames of technical data arranged in a 4x6 grid. Each frame displays a table with multiple columns and rows of text, likely representing test results or diagnostic data. The text is small and difficult to read due to the resolution of the scan. The frames are separated by a grid of small white lines. The overall appearance is that of a standard microfiche card used for data storage and retrieval.

IDENTIFICATION

PRODUCT CODE: AC-F756A-MC
PRODUCT NAME: CVDRCA0 DRV11J DIAG TST PRT1
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 OR PORTS 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 APSOLJTE 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 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 (VDRCA 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.

- 4. IF THE FOLLOWING ERROR OCCURS RIGHT AFTER STARTUP IT IS POSSIBLE THAT THE DRV11J LOOPBACK CABLE MAY NOT BE INSTALLED OR WAS NOT INSTALLED PROPERLY:

```
REG READ/WRITE ER
ERRPC  TSTNUM  BUSADR  EXPCT  RCVD
002224 000002 *164160 100700 000700
*BUSADR MAY BE DIFFERENT DEPENDING ON THE CSR OF THE DRV11J.
```

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

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

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

5.2 CONTROL

- 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.

TO CONSERVE MEMORY SPACE FOR THE 4K PROGRAM MEMORY REQUIREMENT, REGISTER 5 (R5) IS RESERVED FOR THE \$BDDAT LOCATION (1126).

EXAMPLE: CMP \$GDDAT,(R5) IS THE SAME AS CMP \$GDDAT,\$BDDAT.

6.1.1 ERROR DATA

ERROR TITLE HEADING

ERRPC	TSNUM	BUSADR	EXPCT	RCVD
XXXXXX	XXXXXX	XXXXXX	XXXXXX	XXXXXX

ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
TSNUM	TEST NUMBER WHERE THE ERROR OCCURRED
BUSADR	DRV11J BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED

6.2.2 ERROR TITLES

REG TIMEOUT ER	;REGISTER TIMEOUT ERROR
REG READ/WRITE ER	;REGISTER READ/WRITE ERROR
IRR REG ER	;INTERRUPT REQUEST REGISTER ERROR
ACR REG ER	;AUTOCLEAR REGISTER
IMR REG ER	;INTERRUPT MASK REGISTER ERROR
ISR REG ER	;INTERRUPT SERVICE REGISTER ERROR
CHIP STAT ER	;CHIP STATUS ERROR

7.0 MISCELLANEOUS

7.1 DRV11-J BUS ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' (ADDR: 1244) IF BASE BUS ADDRESS IS NOT 164160.

7.2 XXDP/APT NOTES

THIS DIAGNOSTIC DOES SUPPORT ACT AND APT ENVIRONMENTS.

7.3 POWER FAIL

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

7.4 MULTIPLE DRV11J INTERFACE TESTING

THIS PROGRAM DOES NOT "AUTO-SIZE" THE NUMBER OF DRV11J'S CONNECTED. THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 4 DRV11J INTERFACES WITH CONTIGUOUS BUS ADDRESSES. THIS IS ACCOMPLISHED BY THE USER SETTING UP LOCATION 'DEVM' (ADDR: 1246) WITH A BIT MAP INDICATING WHAT INTERFACES ARE TO BE TESTED.

I.E. BIT0 = 1 SAYS TEST 1ST DRV11J,
 BIT1 = 1 SAYS TEST 2ND DRV11J
 BIT2 = 1 SAYS TEST 3RD DRV11J
 BIT3 = 1 SAYS TEST 4TH DRV11J

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

7.5 RESTRICTIONS

8.0 EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT <5 SECONDS ON FIRST PASS WITH NO ITERATIONS TO <20 SECONDS WITH ITERATIONS ENABLED AFTER FIRST PASS, 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.

TESTING WITH THE DRV11J CABLE ALLOWS FOR TESTING OF PORT A WITH PORT C AND THE TESTING OF PORT B WITH PORT D.

9.1 TESTS T1-T22 REGISTER CHECKING

THESE TESTS WILL WRITE/READ PORTS A TO C AND WRITE/READ PORTS B TO D WITH FLOATING ONES AND FLOATING ZEROS, GROWING ONES AND GROWING ZEROS AND DATA PATTERNS. TESTS ARE PERFORMED TO INSURE INTERACTION WITH CONNECTED PORTS AND NO INTERACTION BETWEEN UNCONNECTED PORTS.

9.2 T22-T42 INTERRUPT CONTROL REGISTER CHECKING

TESTS ARE MADE TO THE INTERRUPT CONTROL CHIP REGISTERS IRR, ACR, IMR AND LIMITED TESTING OF ISR REGISTER UNTIL INTERRUPTS ARE PERFORMED. TESTS ARE PERFORMED ON THE REGISTERS WITH FLOATING ONES, FLOATING ZEROS, GROWING ONES AND GROWING ZEROS. CHIP RESET CAPABILITIES ARE TESTED AS WELL AS UNIQUENESS OF REGISTERS WITHIN A CHIP GROUP AND THE UNIQUENESS OF ONE GROUP'S REGISTERS TO ANOTHER GROUP'S REGISTERS.

9.3 TEST 43 - TEST STATUS BITS GINT, S2, S1, S0, GP1, GP2

EXERCISE THE STATUS BITS S2, S1, S0 AND GINT FOR EACH GROUP CHIP BY SETTING SINGLE IRR BITS TO CAUSE THE STATUS BITS TO GO FROM 120 TO 127. EACH IRR BIT IS THEN MASKED TO RETURN STATUS BACK TO ORIGINAL STATUS WITH NO IRR BITS SET.

9.4 T44-T47 POLLED MODE TESTING

THIS TEST WILL WRITE PATTERNS INTO DBRA WITH EITHER LOW TO HIGH OR HIGH TO LOW POLARITIES. AFTER PLACING ALL ONES IN DBRA AND THEN SELECTING ACTIVE LOW, CLEARING DBRA WILL NOW SET IRR BITS 0-7, GROUP 1 AND IRR BITS 0-3, GROUP 2. THE RPLY SIGNALS WILL SET IRR BITS 4-7 WHEN WRITING DBRA IN OUTPUT MODE, THIS WILL

SET THE RPLY FOR DBRC.(IRR6, GROUP 2)
WHEN READING DBRC IN INPUT MODE WILL SET
THE RPLY FOR DBRA.(IRR BIT 4, GROUP 2)
WHEN WRITING DBRB IN OUTPUT MODE, THIS WILL
SET THE RPLY FOR DBRD(IRR BIT 7, GROUP 2)
WHEN READING DBRD IN INPUT MODE, THIS WILL
SET THE RPLY FOR DBRB(IRR BIT 5, GROUP 2).
TEST ALL DATA PATTERNS TO SET IRR BITS
GROUP 1 AND GROUP 2 AND FOR THE SETTING OF RPLY
BITS IN THE GROUP 2 IRR REGISTER BY WRITING IN
OUTPUT MODE AND READING IN INPUT MODE.
TEST THAT NO RPLY BITS SET WHEN READING IN OUTPUT
MODE AND WHEN WRITING IN INPUT MODE.

4.5 T50-T51 CSR'S WITH RESET
SET UP CSR'S IN OUTPUT MODE AND CLEAR
DIRECTION BIT OUT OF EACH CSR EXCEPT FOR
CSRA WHICH WILL CLEAR DIR BIT AND I/E BIT.

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
400	PROGRAM START
402	INITIALIZE THE COMMON TAGS
492	T1 TEST THAT ALL REGISTERS ARE ADDRESSABLE
512	T2 TEST CSRA W/R DIR BIT,INT. CHIP RESET STATUS
553	T3 TEST CSRA INT. ENABLE BIT
578	T4 TEST CSRA I/E,DIR BIT
603	T5 TEST CSRB W/R DIR BIT,INT CHIP RESET STATUS
638	T6 TEST CSRC W/R DIR BIT,INT CHIP RESET STATUS
677	T7 TEST CSRD W/R DIR BIT,INT CHIP RESET STATUS
712	T10 TEST DBRA W/R IN OUTPUT MODE
735	T11 TEST DBRB W/R IN OUTPUT MODE
759	T12 TEST DBRC W/R IN OUTPUT MODE
783	T13 TEST DBRD W/R IN OUTPUT MODE
807	T14 TEST CSR UNIQUENESS,CSRS (A-B),(C-D)
846	T15 TEST CSR UNIQUENESS,CSRS (A-D),(C-B)
885	T16 TEST PORT A TO PORT C INTERACTION
944	T17 TEST PORT B TO PORT D INTERACTION
1000	T20 TEST PORT C TO PORT A INTERACTION
1058	T21 TEST PORT D TO PORT B INTERACTION
1113	T22 TEST GROUP 1,2 IMR,IRR,ACR WITH CHIP RESET
1147	T23 TEST GROUPS 1 AND 2 ACR UNIQUENESS
1195	T24 TEST GROUPS 1 AND 2 IMR UNIQUENESS
1242	T25 TEST GROUPS 1 AND 2 IRR UNIQUENESS
1290	T26 TEST GROUPS 1,2 ACR WITH PATTERNS
1319	T27 TEST GROUPS 1,2 IMR WITH PATTERNS
1348	T30 TEST GROUP 1,2 CLEAR IMR INSTR.
1371	T31 TEST GROUP 1,2 SET IMR INSTR.
1395	T32 TEST GROUP 1,2 CLEAR SINGLE IMR BIT INSTR.
1428	T33 TEST GROUP 1,2 SET SINGLE IMR BIT INSTR.
1460	T34 TEST GROUP 1,2 SET IRR INSTR.
1485	T35 TEST GROUP 1,2 CLEAR IRR INSTR.
1510	T36 TEST GROUP 1,2 CLEAR SINGLE IRR BIT INSTR.
1543	T37 TEST GROUP 1,2 SET SINGLE IRR BIT INSTR.
1576	T40 TEST GROUP 1,2 CLEAR IRR+IMR INSTR.
1607	T41 TEST GROUP 1,2 CLEAR SINGLE IRR+IMR BIT INSTR.
1646	T42 TEST GROUPS 1,2 FOR GROUP UNIQUENESS
1696	T43 TEST STATUS BITS GINT,S2,S1,S0,GP1,2
1759	T44 TEST POLLED MODE;CSRS A,B=OUT C,D=IN,ACTIVE LOW
1821	T45 TEST GROUPS 1,2 IN POLLED MODE,NO REPLY
1884	T46 TEST POLLED MODE;CSRS C,D=OUT A,B=IN,ACTIVE LOW
1946	T47 TEST IRR BITS WITH DATA PAT.,POLLED MODE,ACT. HIGH
1986	T50 TEST CSRA AND CSRB WITH RESET
2013	T51 TEST CSRC AND CSRD WITH RESET
2053	END OF PASS ROUTINE
2091	PROGRAM SUBROUTINES
2125	PATTERNS FOR REGISTER R/W
2230	SYSMAC ROUTINES

2232	TYPE ROUTINE
2311	APT COMMUNICATIONS ROUTINE
2368	BINARY TO OCTAL (ASCII) AND TYPE
2415	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2512	ERROR HANDLER ROUTINE
2567	ERROR MESSAGE TIMEOUT ROUTINE
2574	SCOPE HANDLER ROUTINE
2679	TTY INPUT ROUTINE
2818	POWER DOWN AND UP ROUTINES
2876	TRAP DECODER
2883	TRAP TABLE
2907	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

(01'00)

0000 1
0000 12
0000 15
0000 20
177776

1777 74
1777 72
1775 70
1775 70

(000 0)
(000 1)
(00 02)
(00 03)
(00 04)
(00 05)
(00 06)
(000 07)
(000 08)
(000 09)

```

.TITLE CVDRCA DRV11J DIAG TST PRT1
;*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 TYPEOUTS
;* 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
    
```

```
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
65
66          :*'SWITCH REGISTER' SWITCH DEFINITIONS
67          100000          SW15=   100000
68          040000          SW14=    40000
69          020000          SW13=    20000
70          010000          SW12=    10000
71          004000          SW11=     4000
72          002000          SW10=     2000
73          001000          SW09=     1000
74          000400          SW08=      400
75          000200          SW07=     200
76          000100          SW06=     100
77          000040          SW05=      40
78          000020          SW04=     20
79          000010          SW03=     10
80          000004          SW02=      4
81          000002          SW01=      2
82          000001          SW00=      1
83          .EQUIV SW09,SW9
84          .EQUIV SW08,SW8
85          .EQUIV SW07,SW7
86          .EQUIV SW06,SW6
87          .EQUIV SW05,SW5
88          .EQUIV SW04,SW4
89          .EQUIV SW03,SW3
90          .EQUIV SW02,SW2
91          .EQUIV SW01,SW1
92          .EQUIV SW00,SW0
93
94          :*DATA BIT DEFINITIONS (BIT00 TO BIT15)
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 TRIVEC= 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= 50 ;:TTY KEYBOARD VECTOR
133 000064 TPV= 64 ;:TTY PRINTER VECTOR
134 000240 ;:PROGRAM INTERRUPT REQUEST VECTOR
135 164160 ABASE= 164160 ;:BASE ADDRESS
136 000001 ADEVM= 1 ;:DEFAULT TO ONE DRV11J
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= 30C          ;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          000174          .=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 001402          JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
184          000100          .-100
185 000100 000104 000200 000002          .WORD 104,200,?          ;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 012746
000052 000052
000000 000106
001000 001000

0C1000
000024 000200
000044 001000
001000 000000
001002 001170
001004 000006
001006 000024
001010 000024
001012 000031

```
.SBTTL ACT11 HOOKS
:*****
:HOOKS REQUIRED BY ACT11
    $SVPC=.          ;SAVE PC
    .=46             ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
    $ENDAD           ;:2)SET LOC.52 TO ZERO .
    .-52             ;: RESTORE PC
    .WORD 0
    .-$SVPC
    .-1000
                                ;LONGEST TEST TIME
                                ;1ST PASS RUN TIME
                                ;ADDITIONAL RUN TIME

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

$APTHD:
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$TSTM:  .WORD 6      ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 20     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 20     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
                .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-FTABLE(WORDS)
```



```
223      .SBTTL  COMMON TAGS
224
225      ::*****
226      ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
227      ::*USED IN THE PROGRAM.
228
229      001100      .=1100
230      001100      SCMTAG:      ::START OF COMMON TAGS
231      001100      000000      .WORD      0
232      001102      000      $TSTNM: .BYTE      0      ::CONTAINS THE TEST NUMBER
233      001103      000      $ERFLG: .BYTE      0      ::CONTAINS ERROR FLAG
234      001104      000000      $ICNT:  .WORD      0      ::CONTAINS SUBTEST ITERATION COUNT
235      001106      000000      $LPADR: .WORD      0      ::CONTAINS SCOPE LOOP ADDRESS
236      001110      000000      $LPERR: .WORD      0      ::CONTAINS SCOPE RETURN FOR ERRORS
237      001112      000000      $ERTTL: .WORD      0      ::CONTAINS TOTAL ERRORS DETECTED
238      001114      000      $ITEMB: .BYTE      0      ::CONTAINS ITEM CONTROL BYTE
239      001115      001      $ERMAX: .BYTE      1      ::CONTAINS MAX. ERRORS PER TEST
240      001116      000000      $ERRPC: .WORD      0      ::CONTAINS PC OF LAST ERROR INSTRUCTION
241      001120      000000      $GDADR: .WORD      0      ::CONTAINS ADDRESS OF 'GOOD' DATA
242      001122      000000      $BDADR: .WORD      0      ::CONTAINS ADDRESS OF 'BAD' DATA
243      001124      000000      $GDDAT: .WORD      0      ::CONTAINS 'GOOD' DATA
244      001126      000000      $BDDAT: .WORD      0      ::CONTAINS 'BAD' DATA
245      001130      000000      .WORD      0      ::RESERVED--NOT TO BE USED
246      001132      000000      .WORD      0
247      001134      000      $AUTOB: .BYTE      0      ::AUTOMATIC MODE INDICATOR
248      001135      000      $INTAG: .BYTE      0      ::INTERRUPT MODE INDICATOR
249      001136      000000      .WORD      0
250      001140      177570      $WR:      .WORD      DSWR      ::ADDRESS OF SWITCH REGISTER
251      001142      177570      $DISPLAY: .WORD      DDISP      ::ADDRESS OF DISPLAY REGISTER
252      001144      177560      $TKS:      177560      ::TTY KBD STATUS
253      001146      177562      $TKB:      177562      ::TTY KBD BUFFER
254      001150      177564      $TPS:      177564      ::TTY PRINTER STATUS REG. ADDRESS
255      001152      177566      $TPB:      177566      ::TTY PRINTER BUFFER REG. ADDRESS
256      001154      000      $NULL:  .BYTE      0      ::CONTAINS NULL CHARACTER FOR FILLS
257      001155      002      $FILLS: .BYTE      2      ::CONTAINS # OF FILLER CHARACTERS REQUIRED
258      001156      012      $FILLC: .BYTE      12      ::INSERT FILL CHARS. AFTER A 'LINE FEED'
259      001157      000      $TPFLG: .BYTE      0      ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
260      001160      000000      $TIMES: 0      ::MAX. NUMBER OF ITERATIONS
261      001162      000000      $ESCAPE:0      ::ESCAPE ON ERROR ADDRESS
262      001164      077      $QUES:  .ASCII  '??'      ::QUESTION MARK
263      001165      015      $CRLF:  .ASCII  '<15>'      ::CARRIAGE RETURN
264      001166      000012      $LF:    .ASCII  '<12>'      ::LINE FEED
265      ::*****
266      .SBTTL  APT MAILBOX-ETABLE
267
268      ::*****
269      .EVEN
270      001170      $MAIL:      ::APT MAILBOX
271      001170      000000      $MSGTY: .WORD      AMSGTY      ::MESSAGE TYPE CODE
272      001172      000000      $FATAL: .WORD      AFATAL      ::FATAL ERROR NUMBER
273      001174      000000      $TESTN: .WORD      ATESTN      ::TEST NUMBER
274      001176      000000      $PASS:  .WORD      APASS      ::PASS COUNT
275      001200      000000      $DEVCT: .WORD      ADEVCT      ::DEVICE COUNT
276      001202      000000      $UNIT:  .WORD      AUNIT      ::I/O UNIT NUMBER
277      001204      000000      $MSGAD: .WORD      AMSGAD      ::MESSAGE ADDRESS
278      001206      000000      $MSGLG: .WORD      AMSLG      ::MESSAGE LENGTH
```

```
279 001210          SETABLE:          ;; APT ENVIRONMENT TABLE
280 001210          $ENV: .BYTE   AENV          ;; ENVIRONMENT BYTE
281 001211          $ENVM: .BYTE   AENVM         ;; ENVIRONMENT MODE BITS
282 001212          $SWREG: .WORD   ASWREG        ;; APT SWITCH REGISTER
283 001214          $USWR: .WORD   AUSWR         ;; USER SWITCHES
284 001216          $CPUOP: .WORD   ACPUOP        ;; CPU TYPE, OPTIONS
285                :          ;;
286                :          ;;          BITS 15-11=CPU TYPE
287                :          ;;          11/04=01,11/05=02,11/20=03,11/40=04,11/45-05
288                :          ;;          11/70=06,PDW=07,G=10
289                :          ;;          BIT 1J=REAL TIME CLOCK
290                :          ;;          BIT 9=FLOATING POINT PROCESSOR
291                :          ;;          BIT 8=MEMORY MANAGEMENT
291 001220          $MAMS1: .BYTE   AMAMS1        ;; HIGH ADDRESS, M.S. BYTE
292 001221          $MTYP1: .BYTE   AMTYP1        ;; MEM. TYPE, BLK#1
293                :          ;;          MEM. TYPE BYTE -- (HIGH BYTE)
294                :          ;;          900 NSEC COHE=001
295                :          ;;          300 NSEC BIPOLAR=002
296                :          ;;          500 NSEC MOS=003
297 001222          $MADR1: .WORD   AMADR1        ;; HIGH ADDRESS, BLK#1
298                :          ;;          MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
299 001224          $MAMS2: .BYTE   AMAMS2        ;; HIGH ADDRESS, M.S. BYTE
300 001225          $MTYP2: .BYTE   AMTYP2        ;; MEM. TYPE, BLK#2
301 001226          $MADR2: .WORD   AMADR2        ;; MEM. LAST ADDRESS, BLK#2
302 001230          $MAMS3: .BYTE   AMAMS3        ;; HIGH ADDRESS, M.S. BYTE
303 001231          $MTYP3: .BYTE   AMTYP3        ;; MEM. TYPE, BLK#3
304 001232          $MADR3: .WORD   AMADR3        ;; MEM. LAST ADDRESS, BLK#3
305 001234          $MAMS4: .BYTE   AMAMS4        ;; HIGH ADDRESS, M.S. BYTE
306 001235          $MTYP4: .BYTE   AMTYP4        ;; MEM. TYPE, BLK#4
307 001236          $MADR4: .WORD   AMADR4        ;; MEM. LAST ADDRESS, BLK#4
308 001240          $VECT1: .WORD   AVECT1        ;; INTERRUPT VECTOR#1, BUS PRIORITY#1
309 001242          $VECT2: .WORD   AVECT2        ;; INTERRUPT VECTOR#2, BUS PRIORITY#2
310 001244          $BASE: .WORD   ABASE         ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
311 001246          $DEVN: .WORD   ADEVN         ;; DEVICE MAP
312 001250          $CDW: .WORD   ACDW          ;; CONTROLLER DESCRIPTION WORD#1
313 001252          $ETEND:
314                .EXIT
```

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 \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
:*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
:*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

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

\$ERRTB:

:ERROR 1
EM1 :REG TIMEOUT ER
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT(R5)
0

:ERROR 2
EM2 :REG READ/WRITE ER
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT(R5)
0

:ERROR 3
EM3 :IRR REG ER
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT(R5)
0

:ERROR 4
EM4 :ACR REG ER
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT(R5)
0

:ERROR 5
EM5 :IMR REG ERROR
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT(R5)
0

:ERROR 6
EM6 :ISR REG ERROR
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT(R5)
0

:ERROR 7
EM7 :CHIP STAT ER
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT(R5)

001252
001252 016557
001254 016711
001256 017034
001260 000000
001262 016576
001264 016711
001266 017034
001270 000000
001272 016620
001274 016711
001276 017034
001300 000000
001302 016633
001304 016711
001306 017034
001310 000000
001312 016646
001314 016711
001316 017034
001320 000000
001322 016661
001324 016711
001326 017034
001330 000000
001332 016674
001334 016711
001336 017034

371 001340 000000

0

372

373

374

; BUS REGISTER ADDRESS POINTERS

375

376

377 001342 164160

DRCSA: ABASE

378 001344 164162

DRDBA: ABASE+2

379 001346 164164

DRCSB: ABASE+4

380 001350 164166

DRDBB: ABASE+6

381 001352 164170

DRDSC: ABASE+10

382 001354 164172

DRDBC: ABASE+12

383 001356 164174

DRDSD: ABASE+14

384 001360 164176

DRDBD: ABASE+16

385

386

387

; COMMON PROGRAM LOCATION(S)

388

389 001362 000000

TSTNUM: 0

; CONTAINS TEST NUMBER ON ERROR

390 001364 000001

DMAP: 1

391 001366 000000

INTFLG: .WORD 0

392 001370 000000

XXDP: .WORD 0

393 001372 000000

IMRLOC: .WORD 0

394 001374 000000

ISRLOC: .WORD 0

395 001376 000000

IRRLOC: .WORD 0

396 001400 000000

ARRLOC: .WORD 0

```

397          .SBTTL PROGRAM START
398 001402
399
400          START:
          .SBTTL INITIALIZE THE COMMON TAGS
          ;; CLEAR THE COMMON TAGS ($CMTAG) AREA
401 001402 012706 001100:  MOV    # $CMTAG,R6      ;; FIRST LOCATION TO BE CLEARED
402 001406 005026      CLR    (R6)+            ;; CLEAR MEMORY LOCATION
403 001410 022706 001140      CMP    # $SWR,R6      ;; DONE?
404 001414 001374      BNE    -6              ;; LOOP BACK IF NO
405 001416 012706 001100      MOV    #1'00,SP        ;; SETUP THE STACK POINTER
406          ;; INITIALIZE A FEW VECTORS
407 001422 012737 015142 000020  MOV    # $SCOPE,@IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
408 001430 012737 000340 000022  MOV    #340,@IOTVEC+2  ;; -LEVEL 7
409 001436 012737 014614 000030  MOV    # $ERROR,@EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
410 001444 012737 000340 000032  MOV    #340,@EMTVEC+2  ;; LEVEL 7
411 001452 012737 016402 000034  MOV    # $TRAP,@TRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
412 001460 012737 000340 000036  MOV    #340,@TRAPVEC+2 ;; LEVEL 7
413 001466 012737 016176 000024  MOV    # $SPWRDN,@PWRVEC ;; POWER FAILURE VECTOR
414 001474 012737 000340 000026  MOV    #340,@PWRVEC+2  ;; LEVEL 7
415 001502 005037 001160      CLR    $TIMES          ;; INITIALIZE NUMBER OF ITERATIONS
416 001506 005037 001162      CLR    $ESCAPE         ;; CLEAR THE ESCAPE ON ERROR ADDRESS
417 001512 112737 000001 001115  MOVB   #1,$ERMAX       ;; ALLOW ONE ERROR PER TEST
418 001520 012737 001520 001106  MOV    #.,$SLPADR      ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
419 001526 012737 001526 001110  MOV    #.,$LPERR       ;; SETUP THE ERROR LOOP ADDRESS
420          ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
421          ;; EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
422 001534 013746 000004      MOV    @ $ERRVEC,-(SP)  ;; SAVE ERROR VECTOR
423 001540 012737 001574 000004  MOV    #64,$ERRVEC     ;; SET UP ERROR VECTOR
424 001546 012737 177570 001140  MOV    # $DSWR,$SWR    ;; SETUP FOR A HARDWARE SWICH REGISTER
425 001554 012737 177570 001142  MOV    # $DDISP,$DISPLAY ;; AND A HARDWARE DISPLAY REGISTER
426 001562 022777 177777 177350  CMP    #-1,@ $SWR      ;; TRY TO REFERENCE HARDWARE SWR
427 001570 001012      BNE    66$            ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
428          ;; AND THE HARDWARE SWR IS NOT -1
429 001572 000403      BR    65$            ;; BRANCH IF NO TIMEOUT
430 001574 012716 001602 64$:  MOV    #65$,(SP)      ;; SET UP FOR TRAP RETURN
431 001600 000002      RTI
432 001602 012737 000176 001140 65$:  MOV    # $SWREG,$SWR   ;; POINT TO SOFTWARE SWR
433 001610 012737 000174 001142  MOV    # $DISPREG,$DISPLAY
434 001616 012637 000004 66$:  MOV    (SP)+,@ $ERRVEC ;; RESTORE ERROR VECTOR
435
436 001622 005037 001176      CLR    $PASS          ;; CLEAR PASS COUNT
437 001626 132737 000200 001211 BITB   # $APTSIZE,$ENVM ;; TEST USER SIZE UNDER APT
438 001634 001403      BEQ   67$            ;; YES, USE NON-APT SWITCH
439 001636 012737 001212 001140 67$:  MOV    # $SSWREG,$SWR  ;; NO, USE APT SWITCH REGISTER
440 001644
441 001644 005037 001172      CLR    $FATAL         ;; CLEAR ERROR NUMBER
442 001650 005037 001170      CLR    $MSGTYP        ;; CLEAR MESSAGE TYPE
443 001654 005037 001174      CLR    $TESTN         ;; CLEAR TEST NUMBER
444          ;; CHECK OPERATING ENVIRONMENT
445 001660 005737 000042      TST   @ $42           ;; ARE WE IN ACT/XXDP AUTO MODE
446 001664 001410      BEQ   1$             ;; BRANCH IF NO
447 001666 023737 000042 000046  CMP    @ $42,@ $46     ;; IS IT ACT AUTO MODE?
448 001674 001410      BEQ   2$             ;; BRANCH IF YES
449 001676 012737 177777 001370  MOV    #-1,$XXDP      ;; SET XXDP CHAIN MODE INDICATOR
450 001704 000404      BR    2$
451 001706 123727 001210 000001 1$:  CMPB   $ENV,#1        ;; ARE WE IN APT AUTO MODE
452 001714 001003      BNE   3$             ;; BRANCH IF NO

```

```

453 001716 112737 000001 001134 2$:   MOVB   #1,$AUTOB      ;SET AUTO MODE INDICATOR
454
455      ;PRINT TITLE IF NOT IN ACT OR APT AUTO MODE
456 001724 005227 177777 3$:   INC     #-1          ;FIRST TIME?
457 001730 001012          BNE     $$            ;SKIP TITLE IF NO
458 001732 005737 001134   TST     $AUTOB       ;ARE WE IN AUTO MODE?
459 001736 001403          BEQ     4$            ;BRANCH TO TITLE TYPEOUT IF NOT
460 001740 005737 001370   TST     XXDP         ;IS THE AUTO MODE UNDER XXDP?
461 001744 001404          BEQ     5$            ;SKIP TITLE IF NOT
462 001746 104401 016462 4$:   TYPE   ,TITLED      ;PRINT OUT THE TITLE
463 001752 104401 016530   TYPE   ,TLCABL      ;PRINT 'DRV1J CABLE REQ'D'
464
465      ;GET THE VALUE IN THE SOFTWARE SWITCH REGISTER
466 001756 005737 001134 5$:   TST     $AUTOB     ;ARE WE IN AUTOMATIC MODE?
467 001762 001001          BNE     START1      ;BRANCH IF YES
468 001764 104406          GTSWR          ;ASK FOR SWR INPUT FROM CONSOLE
469 001766 005037 001202   START1: CLR     $UNIT      ;CLEAR UNIT NUMBER
470 001772 013737 001246 001364   MOV     $DEVN,$DMAP  ;POSITION OF DRV11-J'S
471 002000 042737 177760 001364   BIC     #177760,$DMAP ;UP TO 4 DRV11-J'S ONLY
472 002006 013701 001244   MOV     $BASE,$R1    ;GET BASE ADDRESS
473 002012 010137 001342   MOV     R1,$DRCSA    ;MAKE BUS REG. POINTER = $BASE
474 002016 032737 000001 001364   BIT     #1,$DMAP     ;IS FIRST DRV11-J SELECTED?
475 002024 001002          BNE     NEXPAS      ;YES
476 002026 000137 012624   JMP     NXDEV1      ;ADVANCE BASE DRV11-J ADDRESS
477 002032 012700 001342   NEXPAS: MOV    #DRCSA,$R0 ;SET UP REGISTER ADDRESS POINTERS
478 002036 010120   NEXPA1: MOV    R1,($R0)+ ;LOAD EM,R1 = DRV11-J CSRA ADDRESS
479 002040 062701 000002   ADD     #2,$R1      ;DO POINTERS FOR ALL REGS, A THRU D
480 002044 022700 001362   CMP     #DRDBD+2,$R0 ;ALL DONE?
481 002050 001372          BNE     NEXPA1      ;BR IF NOT
482 002052 012706 001100   MOV     #STACK,$SP  ;ALWAYS RESET STACK
483 002056 012705 001126   MOV     #SBDDAT,$R5 ;INIT R5 WITH SBDDAT
484 002062 013737 001202 001200   MOV     $UNIT,$DEVCT ;LOAD APT COUNTER WITH UNIT NO.
485 002070 106427 000340   MTPS   #PR7        ;SET PRIORITY TO 7
486
487
488
489
490      ;*****
491      ;*TEST 1      TEST THAT ALL REGISTERS ARE ADDRESSABLE
492      ;*****
493 002074 000004   TST1:  SCOPE
494 002076 005037 001124   CLR     $GDDAT      ;NO DATA COMPARE
495 002102 005015   CLR     (R5)        ;NO DATA COMPARE
496 002104 012737 002140 000004   MOV     #2,$@ERRVEC ;SET UP TIMEOUT RETURN ADDR
497 002112 013700 001342   MOV     DRCSA,$R0   ;SET UP 1ST DRV11 BUS ADRS
498 002116 012701 000010   MOV     #8,$R1      ;SET UP REG COUNT
499 002122 010037 001122 1$:   MOV     R0,$BDADR   ;SET UP CURRENT DRV BUS ADRS
500 002126 005010   CLR     (R0)        ;SEE IF THERE
501 002130 005720   TST     (R0)+       ;BUMP TO NEXT
502 002132 005301   DEC     R1          ;COUNT 8 OF THEM
503 002134 001403   BEQ     3$          ;BR IF ALL DONE
504 002136 000771   BR     1$          ;TRY NEXT
505 002140 022626 2$:   CMP     (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
506 002142 104001   ERROR   1           ;BUS ADRS INDICATED DID NOT RESPOND
507 002144 012737 000006 000004 3$:   MOV     #ERRVEC+2,$@ERRVEC ;RESTORE LOC 4
508
;:*****

```

```
509          ;*TEST 2          TEST CSRA W/R DIR BIT,INT. CHIP RESET STAT
510          :*****
511 002152 000004          TST2: SCOPE
512 002154 013700 001342      MOV DRCSA,R0          ;GET CSR ADDRESS
513 002160 013701 001352      MOV DRCSA,R1          ;STORE CSRC ADDRESS
514 002164 005010          CLR (R0)              ;INIT CSRA
515 002166 005011          CLR (R1)              ;INIT CSRC
516 002170 010037 001122      MOV R0,$BDADR         ;STORE CSR ADDRESS
517          ;SET UP EXPECTED DATA
518 002174 012737 100700 001124      MOV #RDY.DIR.BIT7.BIT6,$GDDAT
519 002202 112760 000001 000001      MOVB #BIT0,1(R0)     ;SET DIRECTION BIT
520 002210 011015          MOV (R0),(R5)         ;GET CSR DATA
521 002212 042715 000007      BIC #7,(R5)          ;CLEAR UNDEFINED BITS
522 002216 023715 001124      CMP $GDDAT,(R5)
523 002222 001401          BEQ 100$
524 002224 104002          ERROR 2              ;CSRA ERROR
525 002226 010137 001122      MOV R1,$BDADR         ;STORE CSRC ADDRESS
526 002232 012737 000200 001124      MOV #BIT7,$GDDAT     ;STORE EXPECTED
527 002240 011115          MOV (R1),(R5)         ;STORE CSRC
528 002242 042715 000007      BIC #7,(R5)          ;CLEAR UNDEFINED BITS
529 002246 023715 001124      CMP $GDDAT,(R5)
530 002252 001401          BEQ 1$
531 002254 104002          ERROR 2              ;CSRC ERROR
532 002256 010037 001122      MOV R0,$BDADR         ;CSRA ADDRESS
533 002262 012737 100300 001124      MOV #RDY.BIT7.BIT6,$GDDAT
534 002270 105060 000001          CLRB 1(R0)           ;CLEAR CSRA DIR BIT
535 002274 011015          MOV (R0),(R5)         ;READ CSR
536 002276 042715 000007      BIC #7,(R5)          ;CLEAR UNDEFINED BITS
537 002302 023715 001124      CMP $GDDAT,(R5)
538 002306 001401          BEQ 2$
539 002310 104002          ERROR 2              ;BR IF EQUAL
540 002312 012737 000300 001124      MOV #BIT7.BIT6,$GDDAT
541 002320 112761 000001 000001      MOVB #BIT0,1(R1)     ;CSRC TO OUTPUT MODE
542 002326 011015          MOV (R0),(R5)         ;READ CSRA
543 002330 042715 000007      BIC #7,(R5)          ;CLEAR UNDEFINED BITS
544 002334 023715 001124      CMP $GDDAT,(R5)     ;CHECK IF RDY BIT CLEARED
545 002340 001401          BEQ TST3             ;BR IF EQUAL
546 002342 104002          ERROR 2              ;CSRA REG ERROR
```

```
547          :*****
548          ;*TEST 3          TEST CSRA INT. ENABLE BIT
549          :*****
550          TST3: SCOPE
551 002344 000004          JSR PC,CLRCSR         ;CLEAR CSR REGISTER
552 002346 004737 013002      MOV DRCSA,R0          ;GET CSRA ADDRESS
553 002352 013700 001342      MOV DRCSA,R1          ;GET CSRC ADDRESS
554 002356 013701 001352      MOV DRCSA,R1          ;GET CSRC ADDRESS
555 002362 112761 000001 000001      MOVB #BIT0,1(R1)     ;SET DIRECTION BIT (SRC)
556 002370 010037 001122      MOV R0,$BDADR         ;STORE CSRA ADDRESS
557 002374 012737 001300 001124      MOV #BIT9!BIT7.BIT6,$GDDAT
558 002402 012710 001000      MOV #IE,(R0)         ;SET INTERRUPT ENABLE BIT
559 002406 011015          MOV (R0),(R5)
560 002410 042715 000007      BIC #7,(R5)          ;CLEAR UNDEFINED BITS
561 002414 023715 001124      CMP $GDDAT,(R5)
562 002420 001401          BEQ 1$
563 002422 104002          ERROR 2              ;INT. ENABLE ERROR,CSRA
564 002424 012737 000300 001124      MOV #BIT7.BIT6,$GDDAT
```

```

565 002432 105060 000001 CLR# 1(R0) ;CLEAR I/E BIT
566 002436 011015 MOV (R0),(R5)
567 002440 042715 000007 BIC #7,(R5) ;CLEAR UNDEFINED BITS
568 002444 023715 001124 CMP $GDDAT,(R5)
569 002450 001401 BEQ TST4 ;BR IF EQUAL
570 002452 104002 ERROR 2 ;CSRA ERROR
571
572
573 ;*****
574 ;*TEST 4 TEST CSRA I/E,DIR BIT
575 ;*****
575 002454 000004 TST4: SCOPE
576 002456 004737 013002 JSR PC,CLRCSR ;CLEAR CSR REGISTERS
577 002462 013700 001342 MOV DRCSA,R0 ;GET CSRA ADDRESS
578 002466 010037 001122 MOV R0,$SDADR ;SAVE CSRA ADDRESS
579 002472 012737 101700 001124 MOV #101700,$GDDAT ;EXPECTED I/E,DIR
580 002500 112760 000003 000001 MOVB #BIT1!BIT0,1(R0) ;SET I/E AND DIR BIT
581
582 002506 011015 MOV (R0),(R5)
583 002510 042715 000007 BIC #7,(R5) ;CLEAR UNDEFINED BITS
584 002514 023715 001124 CMP $GDDAT,(R5)
585 002520 001401 BEQ 1$
586 002522 104002 ERROR 2 ;CSRA ERROR
587 002524 012737 100300 001124 1$: MOV #RDY!BIT7!BIT6,$GDDAT
588 002532 005010 CLR (R0)
589 002534 011015 MOV (R0),(R5)
590 002536 042715 000007 BIC #7,(R5) ;CLEAR UNDEFINED BITS
591 002542 023715 001124 CMP $GDDAT,(R5)
592 002546 001401 BEQ TST5 ;BR IF EQUAL
593 002550 104002 ERROR 2 ;CSRA ERROR
594
595
596 ;*****
597 ;*TEST 5 TEST CSRB W/R DIR BIT,IN CHIP RESET STATUS
598 ;*****
599 002552 000004 TST5: SCOPE
600 002554 004737 013002 JSR PC,CLRCSR ;CLEAR CSR REGISTERS
601 002560 013700 001346 MOV DRCSB,R0 ;GET CSR ADDRESS
602 002564 013701 001356 MOV DRCSB,P1 ;GET CSRD ADDRESS
603 002570 010037 001122 MOV R0,$SDADR ;STORE CSRD ADDRESS
604 002574 012737 100400 001124 MOV #RDY!DIR,$GDDAT ;SET UP EXPECTED DATA
605 002602 112760 000001 000001 MOVB #BIT0,1(R0) ;SET DIRECTION BIT
606 002610 011015 MOV (R0),(R5) ;GET CSRB DATA
607 002612 023715 001124 CMP $GDDAT,(R5)
608 002616 001401 BEQ 100$
609 002620 104002 ERROR 2 ;CSRB ERROR
610 002622 010137 001122 100$: MOV R1,$SDADR ;STORE CSRD ADDRESS
611 002626 005037 001124 CLR $GDDAT ;STORE EXPECTED
612 002632 011115 MOV (R1),(R5) ;READ CSRD
613 002634 023715 001124 CMP $GDDAT,(R5)
614 002640 001401 BEQ 1$
615 002642 104002 ERROR 2 ;CSRD ERROR
616 002644 010037 001122 1$: MOV R0,$SDADR ;STORE CSRB ADDRESS
617 002650 012737 100000 001124 MOV #RDY,$GDDAT ;STORE EXPECTED
618 002656 105060 000001 CLR# 1(R0) ;CLEAR CSRD DIR BIT
619 002662 011015 MOV (R0),(R5) ;READ CSRD
620 002664 023715 001124 CMP $GDDAT,(R5)

```



```

621 002670 001401 . BEQ 2$ ;:BR IF EQUAL
622 002672 104002 . ERROR 2
623 002674 005037 001124 2$: CLR $GDDAT ;STORE EXPECTED
624 002700 112761 000001 000001 MOVB #BIT0,1(R1) ;CSR0 TO OUTPUT MODE
625 002706 011015 . MOV (R0),(R5) ;READ CSRB
626 002710 013715 001124 . CMP $GDDAT,(R5) ;RDY BIT CLEARED
627 002714 001401 . BEQ TST6 ;:BR IF EQUAL
628 002716 104002 . ERROR 2 ;CSRB REG ERROR
629
630
631
632

```

 *TEST 6 TEST CSRC W/R DIR BIT,INT CHIP RESET STATUS

```

633 002720 003014 . TST6: SCOPE
634 002722 004737 013002 . JSR PC,CLRCSR ;CLEAR CSF REGISTERS
635 002726 013700 001352 . MOV DRCSA,R0 ;GET CSR ADDRESS
636 002732 013701 001342 . MOV DRCSA,R1 ;GET CSRA ADDRESS
637 002736 011017 001122 . MOV R0,$BDADR ;STORE CSR ADDRESS
638 002742 012737 100600 001124 . MOV #RDY!DIR!BIT7,$GDDAT
639 002750 112760 000001 000001 MOVB #BIT0,1(R0) ;SET DIRECTION BIT
640 002756 011015 . MOV (R0),(R5) ;GET CSR DATA
641 002760 012715 000007 . BIC #7,(R5) ;CLEAR UNDEFINED BITS
642 002764 013715 001124 . CMP $GDDAT,(R5)
643 002770 011401 . BEQ 100$
644 002772 104002 . ERROR 2
645 002774 010137 001122 100$: MOV R1,$BDADR ;STORE CSR ADDRESS
646 003000 012737 000300 001124 . MOV #BIT7,BIT6,$GDDAT
647 003006 011115 . MOV (R1),(R5) ;READ CSRA
648 003010 012715 000007 . BIC #7,(R5) ;CLEAR UNDEFINED BITS
649 003014 013715 001124 . CMP $GDDAT,(R5)
650 003020 001401 . BEQ $
651 003022 104002 . ERROR 2 ;CSRA REG ERROR
652 003024 011017 001122 $ . MOV R0,$BDADR ;STORE CSR ADDRESS
653 003030 012737 100600 001124 . MOV #RDY!BIT7,$GDDAT ;STORE EXPECTED DATA
654 003036 105000 000001 . CLR DIR BIT ;CLEAR CSR DIR BIT
655 003042 011015 . MOV (R0),(R5) ;READ CSRB
656 003044 012715 000007 . BIC #7,(R5) ;CLEAR UNDEFINED BITS
657 003050 013715 001124 . CMP $GDDAT,(R5)
658 003054 001401 . BEQ 2$ ;:BR IF EQUAL
659 003056 104002 . ERROR 2
660 003050 012737 000300 00 2$ 2$: MOV #BIT7,$GDDAT ;EXPECTED DATA
661 003066 112761 000001 00 101 . MOVB #BIT0,1(R1) ;CSR0 TO OUTPUT MODE
662 003074 011015 . MOV (R0),(R5) ;READ CSRB
663 003076 012715 000007 . BIC #7,(R5) ;CLEAR UNDEFINED BITS
664 003102 013715 001124 . CMP $GDDAT,(R5) ;RDY BIT CLEARED
665 003106 001401 . BEQ TST7 ;:BR IF EQUAL
666 003110 104002 . ERROR 2 ;CSRC REG ERROR
667
668
669
670

```

 *TEST 7 TEST CSRD W/R DIR BIT,INT CHIP RESET STATUS

```

671 003112 003014 . TST7: SCOPE
672 003114 004737 013002 . JSR PC,CLRCSR ;CLEAR CSF REGISTERS
673 003120 013700 001352 . MOV DRCSA,R0 ;GET CSR ADDRESS
674 003124 013701 001342 . MOV DRCSB,R1 ;CSR0 ADDRESS
675 003130 011017 001122 . MOV R0,$BDADR ;STORE CSR ADDRESS
676 003134 012737 100600 001124 . MOV #RDY.DIR,$GDDAT ;SET UP EXPECTED DATA

```

```

677 003142 112760 000001 000001      MOVB  #BIT0,1(R0)      ;SET DIRECTION BIT
678 003150 011015      MOV   (R0),(R5)       ;GET CSR DATA
679 003152 023715 001124      CMP   $GDDAT,(R5)
680 003156 001401      BEQ   100$
681 003160 104002      ERROR 2
682 003162 010137 001122      000: MOV   R1,$BDADR       ;STORE CSRB ADDRESS
683 003166 005037 001124      CLR   $GDDA          ;STORE EXPECTED
684 003172 011115      MOV   (R1),(R5)
685 003174 023715 00112-    CMP   $GDDAT,(R5)
686 003200 001401      BEQ   1$
687 003202 104002      ERROR 2 ;CSRB ERROR
688 003204 010037 001122      $:  MOV   R0,$E1DR     ;STORE CSRD ADDRESS
689 003210 012737 100000 00 124  $:  MOV   #RDY,$GDDA    ;STORE EXPECTED
690 003216 105060 000001      CLRB  1(R0)          ;CLEAR CSR DIR BIT
691 003222 011015      MOV   (R0),(R5)      ;READ CSR
692 003227 023715 00 124      CMP   $GDDAT,(R5)
693 003231 001401      BEQ   2$
694 003232 104002      ERROR 2 ;:BR IF EQUAL
695 003234 005037 001124      2$: CLR   $GDDAT        ;EXPECTED
696 003240 112751 000001 000001  $:  MOVB  #BIT0,1(R1)   ;CSRD TO OUTPUT MODE
697 003246 011015      MOV   (R0),(R5)      ;READ CSR
698 003250 023715 001124      CMP   $GDDAT,(R5)   ;RDY BIT CLEARED
699 003254 001401      BEQ   TST10         ;:BR IF EQUAL
700 003256 104002      ERROR 2 ;CSRD REG ERROR
    
```

 : *TEST 10 TEST DBRA W/R IN OUTPUT MODE
 : *****

```

705 003260 000004      TST10: SCOPE
706 003262 004737 013002      JSR   PC,CLRCR      ;CLEAR ALL CSRS
707 003266 012703 013110      MOV   #BEGPAT,R3   ;GET DATA PATTERN TABLE
708 003272 012737 003322 001110      MOV   #1$,$LPERR   ;SET UP SCOPE ADDRESS
709 003300 013700 001342      MOV   DRCSA,R0     ;GET CSRA
710 003304 013701 001344      MOV   DRDBA,R1     ;GET DBRA ADDRESS
711 003310 010137 001122      MOV   R1,$BDADR    ;STORE DBRA ADDRESS
712 003314 112760 000001 000001  $:  MOVB  #BIT0,1(R0)   ;SET CSRA IN OUTPUT MODE
713 003322 011337 001124      1$: MOV   (R3),$GDDAT ;SAVE EXPECTED DATA
714 003326 005011      CLR   (R1)         ;CLEAR DBRA
715 003330 051311      BIS   (R3),(R1)    ;WRITE INTO DBRA
716 003332 011115      MOV   (R1),(R5)    ;READ DBRA
717 003334 023715 001124      CMP   $GDDAT,(R5) ;CHECK W/R DBRA
718 003340 001401      BEQ   2$
719 003342 104002      ERROR 2 ;DBRA W/R ERROR
720 003344 005723      2$: TST   (R3)+      ;INC FOR NEXT PATTERN
721 003346 020327 013410      CMP   R3,#ENDDAT   ;CHECK FOR END
722 003352 001363      BNE   1$           ;DO NEXT PATTERN
    
```

 : *TEST 11 TEST DBRB W/R IN OUTPUT MODE
 : *****

```

727 003354 000004      TST11: SCOPE
728 003356 004737 013002      JSR   PC,CLRCR      ;CLEAR ALL CSRS
729 003362 012703 013110      MOV   #BEGPAT,R3   ;GET DATA PATTERN TABLE
730 003366 012737 003416 001110      MOV   #1$,$LPERR   ;SET UP SCOPE ADDRESS
731 003374 013700 001346      MOV   DRCSB,R0     ;GET CSRB
732 003400 013701 001350      MOV   DRDBB,R1     ;GET DBRB ADDRESS
    
```

```

733 003404 010137 001122      MOV      R1,$BDADR      ;STORE DBRB ADDRESS
734 003410 112760 000001 000001  MOVB     #BIT0,1(R0)    ;SET CSRB IN OUTPUT MODE
735 003416 011337 001124      MOV      (R3),$GDDAT   ;SAVE EXPECTED DATA
736 003422 005011      CLR      (R1)          ;CLEAR DBRB
737 003424 051311      BIS      (R3),(R1)     ;WRITE INTO DBRB
738 003426 011115      MOV      (R1),(R5)     ;READ DBRB
739 003430 023715 001124      CMP      $GDDAT,(R5)   ;CHECK W/R DBRB
740 003434 001401      BEQ      2$           ;NEXT PAT. IF EQUAL
741 003436 104002      ERROR    2            ;DBRB W/R ERROR
742 003440 005723      2$:      TST      (R3)+      ;INC FOR NEXT PATTERN
743 003442 020327 013410      CMP      R3,#ENDDAT   ;CHECK FOR END
744 003446 001363      BNE      1$           ;DO NEXT PATTERN
745
746
747
748
749

```

 *TEST 12 TEST DBRC W/R IN OUTPUT MODE

```

750 003450 000004      TST12:  SCOPE
751 003452 004737 013002      JSR      PC,CLRCSR    ;CLEAR ALL CSRS
752 003456 012703 013110      MOV      #BEGPAT,R3   ;GET DATA PATTERN TABLE
753 003462 012737 003512 001110      MOV      #1$,$LPERR   ;SET UP SCOPE ADDRESS
754 003470 013700 001352      MOV      DRCS,R0      ;GET CSRC
755 003474 013701 001354      MOV      DRDBC,R1     ;GET DBRC ADDRESS
756 003500 010137 001122      MOV      R1,$BDADR    ;STORE DBRC ADDRESS
757 003504 112760 000001 000001  MOVB     #BIT0,1(R0)    ;SET CSRC IN OUTPUT MODE
758 003512 011337 001124      1$:      MOV      (R3),$GDDAT   ;SAVE EXPECTED DATA
759 003516 005011      CLR      (R1)          ;CLEAR DBRC
760 003520 051311      BIS      (R3),(R1)     ;WRITE INTO DBRC
761 003522 011115      MOV      (R1),(R5)     ;READ DBRC
762 003524 023715 001124      CMP      $GDDAT,(R5)   ;CHECK W/R DBRC
763 003530 001401      BEQ      2$           ;NEXT PAT. IF EQUAL
764 003532 104002      ERROR    2            ;DBRC W/R ERROR
765 003534 005723      2$:      TST      (R3)+      ;INC FOR NEXT PATTERN
766 003536 020327 013410      CMP      R3,#ENDDAT   ;CHECK FOR END
767 003542 001363      BNE      1$           ;DO NEXT PATTERN
768
769
770
771
772

```

 *TEST 13 TEST DBRD W/R IN OUTPUT MODE

```

773 003544 000004      TST13:  SCOPE
774 003546 004737 013002      JSR      PC,CLRCSR    ;CLEAR ALL CSRS
775 003552 012703 013110      MOV      #BEGPAT,R3   ;GET DATA PATTERN TABLE
776 003556 012737 003606 001110      MOV      #1$,$LPERR   ;SET UP SCOPE ADDRESS
777 003564 013700 001356      MOV      DRCS,R0      ;GET CSRD
778 003570 013701 001360      MOV      DRDBD,R1     ;GET DBRD ADDRESS
779 003574 010137 001122      MOV      R1,$BDADR    ;STORE DBRD ADDRESS
780 003600 112760 000001 000001  MOVB     #BIT0,1(R0)    ;SET CSRD IN OUTPUT MODE
781 003606 011337 001124      1$:      MOV      (R3),$GDDAT   ;SAVE EXPECTED DATA
782 003612 005011      CLR      (R1)          ;CLEAR DBRD
783 003614 051311      BIS      (R3),(R1)     ;WRITE INTO DBRD
784 003616 011115      MOV      (R1),(R5)     ;READ DBRD
785 003620 023715 001124      CMP      $GDDAT,(R5)   ;CHECK W/R DBRD
786 003624 001401      BEQ      2$           ;NEXT PAT. IF EQUAL
787 003626 104002      ERROR    2            ;DBRD W/R ERROR
788 003630 005723      2$:      TST      (R3)+      ;INC FOR NEXT PATTERN

```

CVDRCA DRV11J DIAG TST PRT1
CVDRCA.P11 18-OCT-79 12:53

MACY11 30A(1052) 18-OCT-79 12:58 PAGE 18
T13 TEST DBRD W/R IN OUTPUT MODE

789 003632 020327 013410
790 003636 001363

CMP R3,#ENDDAT ;CHECK FOR END
BNE 18 ;DO NEXT PATTERN

791
792
793
794
795
796 003640 000004
797 003642 004737 013002
798
799
800
801
802 003646 112760 000003 000001
803 003654 112763 000001 000001
804 003662 010037 001122
805 003666 012737 101700 001124
806 003674 011015
807 003676 042715 000007
808 003702 023715 001124
809 003706 001401
810 003710 104002
811 003712 010137 001122
812 003716 005037 001124
813 003722 011115
814 003724 023715 001124
815 003730 001401
816 003732 104002
817 003734 010337 001122
818 003740 012737 100400 001124
819 003746 011315
820 003750 023715 001124
821 003754 001401
822 003756 104002
823 003760 010237 001122
824 003764 012737 000200 001124
825 003772 011215
826 003774 042715 000007
827 004000 023715 001124
828 004004 001401
829 004006 104002
830
831
832
833
834 004010 000004
835 004012 004737 013002
836
837
838
839
840 004016 112760 000003 000001
841 004024 112761 000001 000001
842 004032 010037 001122
843 004036 012737 101700 001124
844 004044 011015
845 004046 042715 000007
846 004052 023715 001124

```
*****  
*TEST 14 TEST CSR UNIQUENESS,CSRS (A-B),(C-D)  
*****  
TST14: SCOPE  
JSR PC,CLRCR :CLEAR ALL CSRS  
 :CSRA = R0  
 :CSRB = R1  
 :CSRC = R2  
 :CSRD = R3  
MOVB #3,1(R0) :CSRA OUTPUT MODE,I/E  
MOVB #BIT0,1(R3) :CSRD OUTPUT MODE  
MOV R0,$BDADR  
MOV #101700,$GDDAT :EXPECTED DATA  
MOV (R0),(R5)  
BIC #7,(R5) :CLEAR UNDEFINED BITS  
CMP $GDDAT,(R5)  
BEQ 1$  
ERROR 2 :CSRA ERROR  
1$: MOV R1,$BDADR :CHECK CSRB  
CLR $GDDAT  
MOV (R1),(R5)  
CMP $GDDAT,(R5)  
BEQ 2$  
ERROR 2 :CSRB ERROR  
2$: MOV R3,$BDADR  
MOV #100400,$GDDAT  
MOV (R3),(R5)  
CMP $GDDAT,(R5)  
BEQ 3$  
ERROR 2 :CSRD ERROR  
3$: MOV R2,$BDADR :CHECK CSRC  
MOV #BIT7,$GDDAT :EXPECTED  
MOV (R2),(R5)  
BIC #7,(R5)  
CMP $GDDAT,(R5)  
BEQ TST15 :;BR IF EQUAL  
ERROR 2 :CSRC ERROR
```

```
*****  
*TEST 15 TEST CSR UNIQUENESS,CSRS (A-D),(C-B)  
*****  
TST15: SCOPE  
JSR PC,CLRCR :CLR ALL CSRS  
 :CSRA = R0  
 :CSRB = R1  
 :CSRC = R2  
 :CSRD = R3  
MOVB #3,1(R0) :CSRA OUTPUT,I/E  
MOVB #BIT0,1(R1) :CSRB OUTPUT MODE  
MOV R0,$BDADR :CSRA ADDRESS  
MOV #101700,$GDDAT  
MOV (R0),(R5)  
BIC #7,(R5) :CLEAR UNDEFINED BITS  
CMP $GDDAT,(R5)
```

```

847 004056 001400      BEQ      1$
848 004060 010137 001122      MOV      R1,$BDADR      :CHECK CSRB
849 004064 012737 100400 001124      MOV      #100400,$GDDAT
850 004072 011115      MOV      (R1),(R5)
851 004074 023715 001124      CMP      $GDDAT,(R5)
852 004100 001401      BEQ      2$
853 004102 104002      ERROR    2      :CSRB ERROR
854 004104 010237 001122      MOV      R2,$BDADR      :CHECK CSRC
855 004110 012737 000200 001124      MOV      #BIT7,$GDDAT
856 004116 011215      MOV      (R2),(R5)
857 004120 042715 000007      BIC      #7,(R5)
858 004124 023715 001124      CMP      $GDDAT,(R5)
859 004130 001401      BEQ      3$
860 004132 104002      ERROR    2      :CSRC ERROR
861 004134 010337 001122      MOV      R3,$BDADR
862 004140 005037 001124      CLR      $GDDAT
863 004144 011315      MOV      (R3),(R5)
864 004146 023715 001124      CMP      $GDDAT,(R5)
865 004152 001401      BEQ      TST16      :;BR IF EQUAL
866 004154 104002      ERROR    2      :CSRD ERROR

```

```

:*****
: *TEST 16      TEST PORT A TO PORT C INTERACTION
:*****

```

```

870
871
872 004156 000004      TST16:  SCOPE
873 004160 004737 013002      JSR      PC,CLRCSR      :CLEAR ALL CSRS
874 004164 012703 013110      MOV      #BEGPAT,R3      :GET DATA PATTERN TABLE
875 004170 012737 004220 001110      MOV      #1$,$LPERR      :SET UP SCOPE ADDRESS
876 004176 013700 001342      MOV      DRCSA,R0      :GET PORT A, CSRA ADDRESS
877 004202 013701 001352      MOV      DRCSB,R1      :GET PORT C, CSRC ADDRESS
878 004206 013702 001346      MOV      DRCSB,R2      :CSRB ADDRESS
879 004212 112762 000001 000001      MOV      #BIT0,1(R2)      :CSRB IN OUTPUT MODE
880 004220 010037 001122 1$:      MOV      R0,$BDADR      :STORE CSRA ADDRESS
881 004224 005062 000002      CLR      2(R2)      :CLEAR DBRB
882 004230 105061 000001      CLR      1(R1)      :PORT C, CSRC, INPUT MODE
883 004234 112760 000001 000001      MOV      #BIT0,1(R0)      :SET CSRA IN OUTPUT MODE
884 004242 011360 000002      MOV      (R3),2(R0)      :WRITE INTO DBRA
885 004246 016104 000002      MOV      2(R1),R4      :READ DBRC
886 004252 012737 100700 001124      MOV      #RDY!DIR!BIT7!BIT6,$GDDAT
887
888 004260 011015      MOV      (R0),(R5)      :CSRA DIR SHOULD STAY SET
889 004262 042715 000007      BIC      #7,(R5)      :READ CSRA
890 004266 023715 001124      CMP      $GDDAT,(R5)      :CLEAR UNDEFINED BITS
891 004272 001401      BEQ      100$
892 004274 104002      ERROR    2      :CSRA ERROR
893 004276 012737 000200 001124 100$:      MOV      #BIT7,$GDDAT
894 004304 010137 001122      MOV      R1,$BDADR      :CSRC ADDRESS
895 004310 011115      MOV      (R1),(R5)      :READ CSRC
896 004312 042715 000007      BIC      #7,(R5)      :CLEAR UNDEFINED BITS
897 004316 023715 001124      CMP      $GDDAT,(R5)
898 004322 001401      BEQ      101$
899 004324 104002      ERROR    2      :CSRC ERROR
900 004326 013737 001354 001122 101$:      MOV      DRDBC,$BDADR      :STORE DBRC ADDRESS
901 004334 011337 001124      MOV      (R3),$GDDAT      :SAVE EXPECTED
902 004340 010415      MOV      R4,(R5)      :DBRC CONTENTS

```

```

903 004342 023715 001124      CMP      $GDDAT,(R5)      ;CHECK PORT C,DBRC
904 004346 001401              BEQ      2$              ;BEQ TO NEXT SUBTEST
905 004350 104002              ERROR   ?              ;DBRC REG ERROR
906 004352 005137 001124      2$:     COM      $GDDAT      ;SET UP TO WRITE COM DATA FROM
907                                ;PORT C TO PORT A
908 004356 013737 001344 001122      MOV      DRDBA,$BDADR      ;STORE DBRA ADDRESS
909 004364 013761 001124 C00002 3$:     MOV      $GDDAT,2(R1)      ;WRITE PORT C,INPUT MODE
910 004372 105060 000001      CLRB    1(R0)            ;MAKE PORT A INPUT MODE
911 004376 112761 000001 000001      MOVB    #BIT0,1(R1)       ;MAKE PORT C OUTPUT MODE
912 004404 016015 000002      MOV      2(R0),(R5)       ;READ PORT A,DBRA
913 004410 023715 001124      CMP      $GDDAT,(R5)      ;PORT A =PORT C?
914 004414 001401              BEQ      4$              ;CHECK DBRB FOR NO DATA CHANGE
915 004416 104002              ERROR   2              ;PORT A,DBRA REG ERROR
916 004420 013737 001350 001122 4$:     MOV      DRDBB,$BDADR      ;STORE DBRB ADDRESS
917 004426 005037 001124      CLR      $GDDAT          ;CONTENTS SHOULD STAY ZERO
918 004432 016215 000002      MOV      2(R2),(R5)       ;READ DBRB FOR CLEAR
919 004436 023715 001124      JMP      $GDDAT,(R5)      ;CHECK FOR DBRB CLEAR
920 004442 001401              BEQ      5$              ;YES,CONTINUE
921 004444 104002              ERROR   2              ;DBRB INTERACTION ERROR
922 004446 005723              TST     (R3)+            ;INC FOR NEXT PATTERN
923 004450 020327 013410      CMP      R3,#ENDDAT      ;CHECK FOR END
924 004454 001261              BNE     1$              ;DO NEXT PATTERN

```

```

*****
:TEST 17 TEST PORT B TO PORT D INTERACTION
*****

```

```

925
926
927
928
929
930 004455 000004      TST17: SCOPE
931 004460 004737 013002      JSR     PC,CLRCR          ;CLEAR ALL CSRS
932 004464 012703 013110      MOV     #BEGPAT,R3       ;GET DATA PATTERN TABLE
933 004470 012737 004520 001110      MOV     #1$,$LPERR       ;SET UP SCOPE ADDRESS
934 004476 013700 001346      MOV     DRCSB,R0         ;GET PORT B, CSRB ADDRESS
935 004502 013701 001356      MOV     DRCSA,R1         ;GET PORT D, CSRD ADDRESS
936 004506 013702 001342      MOV     DRCSA,R2         ;STORE CSRA ADDRESS
937 004512 112762 000001 000001      MCVB   #BIT0,1(R2)       ;CSRA IN OUTPUT MODE
938 004520 010037 001122 5$:     MOV     R0,$BDADR        ;STORE CSRB ADDRESS
939 004524 005062 000002      CLR     2(R2)            ;CLEAR DBRA
940 004530 105061 000001      CLRB   1(R1)            ;PORT D,CSRD,INPUT MODE
941 004534 112760 000001 000001      MOVB   #BIT0,1(R0)       ;SET CSRB IN OUTPUT MODE
942 004542 011360 000002      MOV     (R3),2(R0)       ;WRITE INTO DBRB
943 004546 016104 000002      MOV     2(R1),R4         ;DBRD CONTENTS
944 004552 012737 100400 001124      MOV     #RDY.DIR,$GDDAT  ;SAVE EXPECTED
945 004560 011015      MOV     (R0),(R5)        ;READ CSRB
946 004562 023715 001124      CMP     $GDDAT,(R5)
947 004566 001401              BEQ     100$            ;CSRB ERROR
948 004570 104002              ERROR   2              ;SAVE EXPECTED
949 004572 005037 001124 100$:   CLR     $GDDAT          ;SAVE CSRD ADDRESS
950 004576 010137 001122      MOV     R1,$BDADR
951 004602 011115      MOV     (R1),(R5)
952 004604 023715 001124      CMP     $GDDAT,(R5)
953 004610 001401              BEQ     101$
954 004612 104002              ERROR   2              ;CSRD ERROR
955 004614 013737 001360 001122 101$:   MOV     DRDBD,$BDADR      ;SAVE DBRD ADDRESS
956 004622 011337 001124      MOV     (R3),$GDDAT      ;SAVE EXPECTED
957 004626 010415      MOV     R4,(R5)         ;DBRD CONTENTS
958 004630 023715 001124      CMP     $GDDAT,(R5)      ;CHECK PORT D,DBRD

```

```

959 004634 001401 BEQ 2$ :BEQ TO NEXT SUBTEST
960 004636 104002 ERROR 2 :DBRD REG ERROR
961 004640 005137 001124 2$: COM $GDDAT :SET UP TO WRITE COM DATA FROM
962 :PORT D TO PORT B
963 004644 013737 001350 001122 MOV DRDBB,$BDADR :STORE DBRB ADDRESS
964 004652 013761 001124 000002 3$: MOV $GDDAT,2(R1) :WRITE PORT D, INPUT MODE
965 004660 05060 000001 CLR 1(R0) :MAKE PORT B INPUT MODE
966 004664 112761 000001 000001 MOVB #BIT0,1(R1) :MAKE PORT D OUTPUT MODE
967 004672 016015 000002 MOV 2(R0),(R5) :READ PORT B, DBRB
968 004676 023715 001124 CMP $GDDAT,(R5) :PORT B =PORT D?
969 004702 001401 BEQ 4$ :CHECK DBRA FOR NO DATA CHANGE
970 004704 104002 ERROR 2 :PORT B, DBRB REG ERROR
971 004706 013737 001344 001122 4$: MOV DRDBA,$BDADR :STORE DBRA ADDRESS
972 004714 005037 001124 CLR $GDDAT :CONTENTS = 0
973 004720 016215 000002 MOV 2(R2),(R5) :READ DBRA FOR CLEAR
974 004724 023715 001124 CMP $GDDAT,(R5) :DBRA CLEAR?
975 004730 001401 BEQ 5$ :YES,CONTINUE
976 004732 104002 ERROR 2 :DBRA INTERACTION ERROR
977 004734 005723 013410 5$: TST (R3)+ :INC FOR NEXT PATTERN
978 004736 020327 CMP R3,#ENDDAT :CHECK FOR END
979 004742 001266 BNE $ :DO NEXT PATTERN

```

```

*****
: *TEST 20 TEST PORT C TO PORT A INTERACTION
*****

```

```

981
982
983
984
985 004744 000004 TST20: SCOPE
986 004746 004737 013002 JSR PC,CLRCR :CLEAR ALL CSRS
987 004752 012703 013110 MOV #BEGPAT,R3 :GET DATA PATTERN TABLE
988 004756 012737 005006 001110 MOV #1$, $LPERR :SET UP SCOPE ADDRESS
989 004764 013700 001352 MOV DRCSA,R0 :GET PORT C, CSRC ADDRESS
990 004770 013701 001342 MOV DRCSA,R1 :GET PORT A, CSRA ADDRESS
991 004774 013702 001356 MOV DRCSA,R2 :STORE CSRC ADDRESS
992 005000 112762 000001 000001 MOVB #BIT0,1(R2) :CSRD IN OUTPUT MODE
993 005006 010037 001122 1$: MOV R0,$BDADR :STORE CSRC ADDRESS
994 005012 005062 000002 CLR 2(R2) :CLEAR DBRD
995 005016 105061 000001 CLR 1(R1) :PORT A, CSRA, INPUT MODE
996 005022 112760 000001 000001 MOVB #BIT0,1(RC) :SET CSRC IN OUTPUT MODE
997 005030 011360 000002 MOV (R3),2(R0) :WRITE INTO DBRC
998 005034 016104 000002 MOV 2(R1),R4 :READ DBRA
999 005040 012737 100600 001124 MOV #RDY!DIR BIT7,$GDDAT :CSRC DIR SHOULD BE SET
1000 :READ CSRC
1001 005046 011015 MOV (R0),(R5) :CLEAR UNDEFINED BITS
1002 005050 042715 000007 BIC #7,(R5)
1003 005054 023715 001124 CMP $GDDAT,(R5)
1004 005060 001401 BEQ 100$
1005 005062 104002 ERROR 2 :CSRC ERROR
1006 005064 012737 000300 001124 100$: MOV #BIT7!BIT6,$GDDAT
1007 005072 010137 001122 MOV R1,$BDADR :CSRA ADDRESS
1008 005076 011115 MOV (R1),(R5) :READ CSRA
1009 005100 042715 000007 BIC #7,(R5) :CLEAR UNDEFINED BITS
1010 005104 023715 001124 CMP $GDDAT,(R5)
1011 005110 001401 BEQ 101$
1012 005112 104002 ERROR 2 :CSRA ERROR
1013 005114 013737 001344 001122 101$: MOV DRDBA,$BDADR :STORE DBRA ADDRESS
1014 005122 011337 001124 MOV (R3),$GDDAT :SAVE EXPECTED

```



```

1015 005126 010415          MOV      R4,(R5)          ;DBRA CONTENTS
1016 005130 023715 001124  CMP      $GDDAT,(R5)     ;CHECK PORT A,DBRA
1017 005134 001401          BEQ      2$              ;BEQ TO NEXT SUBTEST
1018 005136 104002          ERROR   2              ;DBRA REG ERROR
1019 005140 005137 001124  2$:     COM      $GDDAT     ;SET UP TO WRITE COM DATA FROM
1020                                ;PORT A TO PORT C
1021 005144 013737 001354 001122  MOV      DRDBC,$BDADR    ;STORE DBRC ADDRESS
1022 005152 013761 001124 000002  3$:     MOV      $GDDAT,2(R1)   ;WRITE PORT A,INPUT MODE
1023 005160 105060 000001          CLRB    1(R0)           ;MAKE PORT C INPUT MODE
1024 005164 112761 000001 000001  MOVB    #BIT0,1(R1)     ;MAKE PORT A OUTPUT MODE
1025 005172 016015 000002          MOV      2(R0),(R5)     ;READ PORT C,DBRC
1026 005176 023715 001124  CMP      $GDDAT,(R5)     ;PORT C =PORT A?
1027 005202 001401          BEQ      4$              ;CHECK DBRD FOR NO DATA CHANGE
1028 005204 104002          ERROR   2              ;PORT C,DBRC REG ERROR
1029 005206 013737 001360 001122  4$:     MOV      DRDBD,$BDADR    ;STORE DBRD ADDRESS
1030 005214 005037 001124          CLR      $GDDAT         ;DBRD = 0 EXPECTED
1031 005220 016215 000002          MOV      2(R2),(R5)     ;READ DBRD FOR CLEAR
1032 005224 023715 001124  CMP      $GDDAT,(R5)     ;IS DBRD CLEAR?
1033 005230 001401          BEQ      5$              ;YES,CONTINUE
1034 005232 104002          ERROR   2              ;DBRD INTERACTION ERROR
1035 005234 005723          TST     (R3)+           ;INC FOR NEXT PATTERN
1036 005236 020327 013410  CMP      R3,#ENDDAT     ;CHECK FOR END
1037 005242 001261          BNE     1$              ;DO NEXT PATTERN

```

```

1038
1039
1040 *****
1040 :TEST 21 TEST PORT D TO PORT B INTERACTION
1041 *****

```

```

1042 005244 000004          TST21: SCOPE
1043 005246 004737 013002  JSR      PC,CLRCR      ;CLEAR ALL CSRS
1044 005252 012703 013110  MOV      #BEGPAT,R3    ;GET DATA PATTERN TABLE
1045 005256 012737 005306 001110  MOV      #1$,$LPERR    ;SET UP SCOPE ADDRESS
1046 005264 013700 001356  MOV      DRCSA,R0      ;GET PORT D, CSRD ADDRESS
1047 005270 013701 001346  MOV      DRCSB,R1      ;GET PORT B, CSRB ADDRESS
1048 005274 013702 001352  MOV      DRCSA,R2      ;STORE CSRD ADDRESS
1049 005300 112762 000001 000001  MOVB    #BIT0,1(R2)    ;CSRD IN OUTPUT MODE
1050 005306 010037 001122  1$:     MOV      R0,$BDADR    ;STORE CSRD ADDRESS
1051 005312 005062 000002          CLR      2(R2)         ;CLEAR DBRC
1052 005316 105061 000001          CLRB    1(R1)         ;PORT B,CSRB,INPUT MODE
1053 005322 112760 000001 000001  MOVB    #BIT0,1(R0)    ;SET CSRD IN OUTPUT MODE
1054 005330 011360 000002          MOV      (R3),2(R0)    ;WRITE INTO DBRD
1055 005334 016104 000002          MOV      2(R1),R4      ;READ DBRB
1056 005340 012737 100400 001124  MOV      #RDY.DIR,$GDDAT ;SAVE EXPECTED
1057 005346 011015          MOV      (R0),(R5)     ;READ CSRD
1058 005350 023715 001124  CMP      $GDDAT,(R5)
1059 005354 001401          BEQ      100$          ;CSRD ERROR
1060 005356 104002          ERROR   2              ;SAVE EXPECTED
1061 005360 005037 001124  100$:  CLR      $GDDAT         ;SAVE CSRB ADDRESS
1062 005364 010137 001122          MOV      R1,$BDADR
1063 005370 011115          MOV      (R1),(R5)
1064 005372 023715 001124  CMP      $GDDAT,(R5)
1065 005376 001401          BEQ      101$          ;CSRB ERROR
1066 005400 104002          ERROR   2              ;SAVE DBRB ADDRESS
1067 005402 013737 001350 001122  101$:  MOV      DRDBB,$BDADR
1068 005410 011337 001124          MOV      (R3),$GDDAT
1069 005414 010415          MOV      R4,(R5)
1070 005416 023715 001124  CMP      $GDDAT,(R5)   ;CHECK PORT B,DBRB

```

```

1071 005422 001401          BEQ      2$          ;BEQ TO NEXT SUBTEST
1072 005424 104002          ERROR    2          ;DBRB REG ERROR
1073 005426 005137 001124  2$:      COM      $GDDAT ;SET UP TO WRITE COM DATA FROM
1074                                ;PORT B TO PORT D
1075 005432 013737 001360 001122  MOV     DRDBD,$BDADR ;STORE DBRD ADDRESS
1076 005440 013761 001124 000002 3$:      MOV     $GDDAT,2(R1) ;WRITE PORT B, INPUT MODE
1077 005446 105060 000001          CLR     1(R0)        ;MAKE PORT D INPUT MODE
1078 005452 112761 000001 000001  MOV     #BIT0,1(R1)   ;MAKE PORT B OUTPUT MODE
1079 005460 016015 000002          MOV     2(R0),(R5)   ;READ PORT D, DBRD
1080 005464 023715 001124          CMP     $GDDAT,(R5)  ;PORT B =PORT D?
1081 005470 001401          BEQ     4$          ;CHECK DBRC FOR NO DATA CHANGE
1082 005472 104002          ERROR    2          ;PORT D, DBRD REG ERROR
1083 005474 013737 001354 001122 4$:      MOV     DRDBC,$BDADR ;STORE DBRC ADDRESS
1084 005502 005037 001124          CLR     $GDDAT      ;EXPECT DBRC = 0
1085 005506 016215 000002          MOV     2(R2),(R5)  ;READ DBRC
1086 005512 023715 001124          CMP     $GDDAT,(R5) ;IS DBRC = 0
1087 005516 001401          BEQ     5$          ;YES, CONTINUE
1088 005520 104002          ERROR    2          ;DBRC INTERACTION ERROR
1089 005522 005723          TST     (R3)+       ;INC FOR NEXT PATTERN
1090 005524 020327 013410          CMP     R3,#ENDDAT ;CHECK FOR END
1091 005530 001266          BNE     1$          ;DO NEXT PATTERN
  
```

 *TEST 22 TEST GROUP 1,2 IMR,IRR,ACR WITH CHIP RESET

```

1096 005532 000004          TST22: SCOPE
1097 005534 004737 013002  JSR     PC,CLRCR    ;CLEAR ALL CSRS
1098 005540 012702 000002  MOV     #2,R2       ;TWO GROUPS TO TEST
1099 005544 013700 001342  MOV     DRCSA,R0    ;STORE CSRA
1100 005550 013701 001346  MOV     DRCSB,R1    ;STORE CSRB
1101 005554 004737 013052  JSR     PC,CLRIRR   ;CLEAR IRR REGISTERS
1102 005560 010137 001122 11$:    MOV     R1,$BDADR   ;STORE ADDRESS
1103 005564 012737 000377 001124  MOV     #377,$GDDAT ;STORE EXPECTED DATA
1104 005572 112710 000244  MOV     #IMR,(R0)   ;LOAD MODE BITS FOR IMR
1105 005576 111115          MOV     (R1),(R5)   ;READ IMR REGISTER
1106 005600 023715 001124          CMP     $GDDAT,(R5)
1107 005604 001401          BEQ     1$          ;IMR REG ERROR
1108 005606 104005          ERROR    5          ;STORE EXPECTED
1109 005610 005037 001124 1$:      CLR     $GDDAT      ;LOAD MODE BITS FOR IRR
1110 005614 112710 000250          MOV     #IRR,(R0)   ;READ IRR REGISTER
1111 005620 111115          MOV     (R1),(R5)
1112 005622 023715 001124          CMP     $GDDAT,(R5)
1113 005626 001401          BEQ     2$          ;IRR REG ERROR
1114 005630 104003          ERROR    3          ;LOAD MODE BITS FOR ACR
1115 005632 112710 000254 2$:      MOV     #MACR,(R0)  ;READ ACR REGISTER
1116 005636 111115          MOV     (R1),(R5)
1117 005640 023715 001124          CMP     $GDDAT,(R5)
1118 005644 001401          BEQ     3$          ;ACR REG ERROR
1119 005646 104004          ERROR    4          ;FINISHED BOTH GROUPS?
1120 005650 005302          DEC     R2          ;BR IF EQUAL
1121 005652 001405          BEQ     TST23
1122 005654 013700 001352  MOV     DRCSA,R0
1123 005660 013701 001356  MOV     DRCSB,R1
1124 005664 000735          BR      111$
  
```

```

1127      : *TEST 23      TEST GROUPS 1 AND 2 ACR UNIQUENESS
1128      : *****
1129      TST23:  SCOPE
1130      JSR      PC,CLRCR      ;CLEAR ALL CSRS
1131      MOV      DRCSA,R0      ;CSRA ADDRESS
1132      MOV      DRCSB,R1      ;CSRB ADDRESS
1133      MOV      #2,R3        ;COUNTER FOR TESTING TWO GROUPS
1134      JSR      PC,CLRIRR     ;CLEAR IRR REGS WITH CHIP RESET
1135      MOV      R1,$BDADR     ;STORE ADDRESS
1136      MOV      #252,$GDDAT   ;STORE EXPECTED
1137      MOV      #MACR,(R0)    ;LOAD MODE BITS FOR ACR
1138      MOV      #PACR,(R0)    ;PRESELECT ACR FOR WRITING
1139      MOV      $GDDAT,(R1)   ;WRITE INTO DATA PORT
1140      MOV      (R1),(R5)     ;STORE DATA FOR COMPARE
1141      CMP      $GDDAT,(R5)   ;CHECK ACR RESULTS
1142      BEQ      1$           ;
1143      ERROR    4             ;ACR ERROR
1144      MOV      #MIMR,(R0)    ;CHANGE TO IMR REGISTER
1145      MOV      #377,$GDDAT   ;STORE EXPECTED
1146      MOV      (R1),(R5)     ;READ IMR
1147      CMP      $GDDAT,(R5)   ;SHOULD STILL BE ALL 1'S
1148      BEQ      2$           ;
1149      ERROR    5             ;IMR REG ERROR
1150      MOV      #MISR,(R0)    ;LOAD MODE BITS FOR ISR
1151      CLR      $GDDAT        ;STORE EXPECTED
1152      MOV      (R1),(R5)     ;READ ISR
1153      CMP      $GDDAT,(R5)   ;ISR SHOULD BE CLEARED
1154      BEQ      3$           ;
1155      ERROR    6             ;ISR REG ERROR
1156      MOV      #MIRR,(R0)    ;LOAD MODE BITS FOR IRR
1157      MOV      (R1),(R5)     ;READ IRR
1158      CMP      $GDDAT,(R5)   ;IRR SHOULD BE CLEARED
1159      BEQ      4$           ;
1160      ERROR    3             ;IRR REG ERROR
1161      CLRB    (R0)          ;CHIP RESET GROUP 1
1162      MOV      #MACR,(R0)    ;LOAD MODE BITS FOR ACR
1163      MOV      (R1),(R5)     ;READ ACR
1164      CMP      $GDDAT,(R5)   ;ACR SHOULD BE CLEARED
1165      BEQ      5$           ;
1166      ERROR    4             ;ACR REG ERROR
1167      DEC     R3            ;TEST GROUPS 1 AND 2
1168      BEQ     TST24        ;BR IF BOTH GROUPS TESTED
1169      MOV     DRCSA,R0      ;GROUP 2 CONTROL PORT
1170      MOV     DRCSB,R1      ;GROUP 2 DATA PORT
1171      BR     111$          ;TEST GROUP 2 ACR UNIQUENESS

```

```

1172      : *****
1173      : *TEST 24      TEST GROUPS 1 AND 2 IMR UNIQUENESS
1174      : *****
1175      TST24:  SCOPE
1176      JSR      PC,CLRCR      ;CLEAR ALL CSRS
1177      MOV      DRCSA,R0      ;CSRA ADDRESS
1178      MOV      DRCSB,R1      ;CSRB ADDRESS
1179      MOV      #2,R3        ;COUNTER FOR TWO GROUP TESTING
1180      JSR      PC,CLRIRR     ;CLEAR IRR REGS WITH CHIP RESET
1181      MOV      R1,$BDADR     ;STORE ADDRESS
1182      MOV      #252,$GDDAT   ;STORE EXPECTED

```

```

1183 006132 112710 000244      MOVB  #MIMR,(R0)      ;LOAD MODE BITS FOR IMR
1184 006136 112710 000260      MOVB  #PIMR,(R0)      ;PRESELECT IMR FOR WRITING
1185 006142 113711 001124      MOVB  $GDDAT,(R1)     ;WRITE INTO DATA PORT
1186 006146 111115                MOVB  (R1),(R5)       ;STORE DATA FOR COMPARE
1187 006150 023715 001124      CMP   $GDDAT,(R5)     ;CHECK IMR RESULTS
1188 006154 001401                BEQ   1$              ;
1189 006156 104005                ERROR  5              ;IMR ERROR
1190 006160 112710 000254      1$:  MOVB  #MACR,(R0)     ;CHANGE TO ACR REGISTER
1191 006164 005037 001124      CLR   $GDDAT          ;STORE EXPECTED
1192 006170 111115                MOVB  (R1),(R5)       ;READ ACR
1193 006172 023715 001124      CMP   $GDDAT,(R5)     ;SHOULD STILL BE CLEARED
1194 006176 001401                BEQ   2$              ;
1195 006200 104004                ERROR  4              ;ACR REG ERROR
1196 006202 112710 000240      2$:  MOVB  #MISR,(R0)     ;LOAD MODE BITS FOR ISR
1197 006206 111115                MOVB  (R1),(R5)       ;READ ISR
1198 006210 023715 001124      CMP   $GDDAT,(R5)     ;ISR SHOULD BE CLEARED
1199 006214 001401                BEQ   3$              ;
1200 006216 104006                ERROR  6              ;SR REG ERROR
1201 006220 112710 000250      3$:  MOVB  #MIRR,(R0)     ;LOAD MODE BITS FOR IRR
1202 006224 111115                MOVB  (R1),(R5)       ;READ IRR
1203 006226 023715 001124      CMP   $GDDAT,(R5)     ;IRR SHOULD BE CLEARED
1204 006232 001401                BEQ   4$              ;
1205 006234 104003                ERROR  3              ;IRR REG ERROR
1206 006236 105010                CLRB  (R0)            ;HIP RESET GROUP 1
1207 006240 112710 000244      4$:  MOVB  #MIMR,(R0)     ;LOAD MODE BITS FOR IMR
1208 006244 012737 000377 001124  MOV   #377,$GDDAT     ;STORE EXPECTED
1209 006252 111115                MOVB  (R1),(R5)       ;READ IMR
1210 006254 023715 001124      CMP   $GDDAT,(R5)     ;IMR SHOULD BE ALL ONES
1211 006260 001401                BEQ   5$              ;DO NEXT GROUP
1212 006262 104005                ERROR  5              ;IMR REG ERROR
1213 006264 005303                DEC   R3              ;DO GROUPS 1 AND GROUP 2
1214 006266 001405                BEQ   TST25          ;ERR IF BOTH GROUPS TESTED
1215 006270 013700 001352      MOV   DRCSA,R0        ;SET UP TO TEST GROUP 2
1216 006274 013701 001356      MOV   DRCSB,R1        ;GROUP 2 DATA PORT
1217 006300 000707                BR    111$           ;DO GROUP 2 IMR UNIQUENESS
  
```

 : TEST 25 TEST GROUPS 1 AND 2 IRR UNIQUENESS

```

1222 006302 000004      TST25: SCOPE
1223 006304 004737 013002      JSR   PC,CLRCSR      ;CLEAR ALL CSRS
1224 006310 013700 001342      MOV   DRCSA,R0       ;CSRA ADDRESS
1225 006314 013701 001346      MOV   DRCSB,R1       ;CSRB ADDRESS
1226 006320 012703 000002      MOV   #2,R3          ;COUNTER FOR TESTING TWO GROUPS
1227 006324 004737 013052      JSR   PC,CLRIRR      ;CLEAR IRR REGS WITH CHIP RESET
1228 006330 010137 001122      11$: MOV   R1,$BDADR     ;STORE ADDRESS
1229 006334 012737 000200 001124  MOV   #200,$GDDAT     ;STORE EXPECTED
1230 006342 112710 000250      MOVB  #MIRR,(R0)     ;LOAD MODE BITS FOR IRR
1231 006346 112710 000137      MOVB  #137,(R0)      ;SET SINGLE IRR BIT 7
1232 006352 111115                MOVB  (R1),(R5)       ;STORE DATA FOR COMPARE
1233 006354 023715 001124      CMP   $GDDAT,(R5)     ;CHECK IRR RESULTS
1234 006360 001401                BEQ   1$              ;
1235 006362 104003                ERROR  3              ;IRR ERROR
1236 006364 112710 000244      1$:  MOVB  #MIMR,(R0)     ;CHANGE TO IMR REGISTER
1237 006370 012737 000377 001124  MOV   #377,$GDDAT     ;STORE EXPECTED
1238 006376 111115                MOVB  (R1),(R5)       ;READ IMR
  
```

```

1239 006400 023715 001124      CMP      $GDDAT,(R5)      ;SHOULD STILL BE ALL 1'S
1240 006404 001401      BEQ      2$
1241 006406 104005      ERROR   5                  ;IMR REG ERROR
1242 006410 112710 000240      2$:     MOVB   #MISR,(R0)    ;LOAD MODE BITS FOR ISR
1243 006414 005037 001124      CLR      $GDDAT          ;STORE EXPECTED
1244 006420 111115      MOVB   (R1),(R5)        ;READ ISR
1245 006422 023715 001124      CMP      $GDDAT,(R5)    ;ISR SHOULD BE CLEAR
1246 006426 001401      BEQ      3$
1247 006430 104006      ERROR   6                  ;ISR REG ERROR
1248 006432 112710 000254      3$:     MOVB   #MACR,(R0)    ;LOAD MODE BITS FOR ACR
1249 006436 111115      MOVB   (R1),(R5)        ;READ ACR
1250 006440 023715 001124      CMP      $GDDAT,(R5)    ;ACR SHOULD BE CLEARED
1251 006444 001401      BEQ      4$
1252 006446 104004      ERROR   4                  ;ACR REG ERROR
1253 006450 105010      CLRB   (R0)             ;CHIP RESULT GROUP 1
1254 006452 112710 000250      4$:     MOVB   #MIRR,(R0)    ;LOAD MODE BITS FOR IRR
1255 006456 111115      MOVB   (R1),(R5)        ;READ IRR
1256 006460 023715 001124      CMP      $GDDAT,(R5)    ;IRR SHOULD BE CLEARED
1257 006464 001401      BEQ      5$
1258 006466 104003      ERROR   3                  ;IRR REG ERROR
1259 006470 005303      5$:     DEC     R3              ;TEST GROUPS 1 AND 2
1260 006472 001405      BEQ     TST26            ;BR IF BOTH GROUPS TESTED
1261 006474 013700 001352      MOV     DRCSC,R0        ;GROUP 2 CONTROL PORT
1262 006476 013701 001356      MOV     DRCSD,R1        ;GROUP 2 DATA PORT
1263 006478 000711      BR      111$           ;TEST GROUP 2 ACR UNIQUENESS
    
```

```

1264
1265
1266 *****
1267 *TEST 26 TEST GROUPS 1,2 ACR WITH PATTERNS
1268 *****
1269 TST26: SCOPE
1270 JSR PC,CLRCR          ;CLEAR ALL CSRS
1271 MOV #2,R2             ;TEST TWO GROUPS
1272 MOV #1,$LPERR        ;SET UP LOOP RETURN
1273 MOV DRCSA,R0         ;STORE CSR1 ADDRESS
1274 MOV DRCSD,R1        ;STORE CSR2 ADDRESS
1275 MOV #BEGPAT,R3      ;SET UP PATTERN TABLE
1276 MOV R1,$BDADR       ;STORE ADDRESS
1277 1$:     MOVB   #MACR,(R0) ;LOAD MODE BITS FOR ACR
1278 MOVB   #PACR,(R0)   ;PRESELECT ACR FOR WRITING
1279 MOVB   (R3),$GDDAT  ;STORE EXPECTED
1280 MOVB   (R3),(R1)    ;WRITE INTO ACR
1281 MOVB   (R1),(R5)    ;READ OUT OF ACR
1282 CMP   $GDDAT,(R5)
1283 BEQ   2$
1284 ERROR 4              ;ACR REGISTER ERROR
1285 2$:     TST   (R3)+    ;INC FOR NEXT PATTERN
1286 CMP   R3,#ENDDAT    ;CHECK FOR TABLE END
1287 BNE   '$           ;W/R NEXT PATTERN
1288 DEC   R2            ;FINISHED BOTH GROUPS
1289 BEQ   TST27        ;BR IF EQLA
1290 MOV   DRCSC,R0
1291 MOV   DRCSD,R1
1292 BR   111$          ;DO NEXT GROUP
1293
1294 *****
    
```

1295
1296
1297 006624 010004
1298 006625 0104737 013002
1299 006626 0112702 000002
1300 006627 0112737 000600 001110
1301 006628 0113700 001112
1302 006629 0113701 001116
1303 006630 0113703 011310
1304 006631 0113710 001122
1305 006632 0113710 001244
1306 006633 0113710 001260
1307 006634 0113717 001244
1308 006635 0113717 001244
1309 006636 0113717 001244
1310 006637 0113715 001244
1311 006638 0113715 001244
1312 006639 0113712 001400
1313 006640 0113712 001400
1314 006641 0113716 013410
1315 006642 0113722 001350
1316 006643 0113724 001350
1317 006644 0113726 001405
1318 006645 0113700 001352
1319 006646 0113701 001356
1320 006647 000745
1321
1322
1323
1324
1325 006742 000004
1326 006744 004737 013002
1327 006750 012704 000002
1328 006754 013700 001342
1329 006760 013701 001346
1330 006764 010137 001122
1331 006770 005037 001124
1332 006774 112710 000244
1333 007000 112710 000040
1334 007004 111115
1335 007006 023715 001124
1336 007012 001401
1337 007014 104005
1338 007016 005304
1339 007020 001405
1340 007022 013700 001352
1341 007026 013701 001356
1342 007032 000754
1343
1344
1345
1346
1347 007034 000004
1348 007036 004737 013002
1349 007042 012704 000002
1350 007046 013700 001342

```
*****
:*TEST 27 TEST GROUPS 1,2 IMR WITH PATTERNS
*****
TST27: SCOPE
        JSR PC,CLRCR      ;CLEAR ALL CSRS
        MOV #2,R4        ;COUNTER FOR TWO GROUPS
        MOV DRCSA,R0     ;CSRA ADDRESS
        MOV DRCSB,R1     ;CSRB ADDRESS
111$:   MOV R1,$BDADR     ;STORE ADDRESS
        MOVB #MIMR,(R0)  ;LOAD MODE BITS FOR IMR
        MOVB #PIMR,(R0)  ;PRESELECT IMR FOR WRITING
        MOVB (R3),%GDDAT ;STORE EXPECTED
        MOVB (R3),(R1)   ;WRITE INTO IMR
        MOVB (R1),(R5)   ;READ OUT OF IMR
        CMP $GDDAT,(R5)
        B.O 5
        ERROR 5
2$:     TST (R3)+        ;IMR REGISTER ERROR
        CMP R3,#ENDDAT  ;INC FOR NEXT PATTERN
        BNE 1$          ;CHECK FOR TABLE END
        DEC R2           ;W/R NEXT PATTERN
        BEQ TST30       ;FINISHED BOTH GROUPS?
        MOV DRCSA,R0    ;;BR IF EQUAL
        MOV DRCSB,R1
        BR 111$         ;DO NEXT GROUP
*****
```

```
*****
:*TEST 30 TEST GROUP 1,2 CLEAR IMR INSTR.
*****
TST30: SCOPE
        JSR PC,CLRCR      ;CLEAR ALL CSRS
        MOV #2,R4        ;COUNTER FOR TWO GROUPS
        MOV DRCSA,R0     ;CSRA ADDRESS
        MOV DRCSB,R1     ;CSRB ADDRESS
111$:   MOV R1,$BDADR     ;STORE ADDRESS
        CLR $GDDAT       ;EXPECTED DATA
        MOVB #MIMR,(R0)  ;LOAD MODE BITS TO READ IMR
        MOVB #CIMR,(R0)  ;CLEAR IMR COMMAND
        MOVB (R1),(R5)   ;READ DATA PORT
        CMP $GDDAT,(R5)
        BEQ 1$
        ERROR 5         ;ERROR ,IMR SHOULD BE CLEAFED
1$:     DEC R4           ;FINISHED BOTH GRO,PS?
        BEQ TST31       ;;BR IF BOTH GROUPS TESTED
        MOV DRCSA,R0    ;SETUP FOR GROUP 2
        MOV DRCSB,R1
        BR 111$         ;DO IMR TEST WITH GROUP 2
*****
```

```
*****
:*TEST 31 TEST GROUP 1,2 SET IMR INSTR.
*****
TST31: SCOPE
        JSR PC,CLRCR      ;CLEAR ALL CSRS
        MOV #2,R4        ;COUNTER FOR TWO GROUPS
        MOV DRCSA,R0     ;CSRA ADDRESS
*****
```

```

1351 007052 013701 001346      MOV      DRCSB,R1      ;CSRB ADDRESS
1352 007056 010137 001122      MOV      R1,$BDADR    ;STORE ADDRESS
1353 007062 012737 000377 001124  MOV      #377,$GDDAT  ;EXPECTED DATA
1354 007070 112710 000244      MOV      #MIMR,(R0)   ;LOAD MODE BITS TO READ IMR
1355 007074 112710 000040      MOV      #CIMR,(R0)   ;CLEAR IMR
1356 007100 112710 000060      MOV      #SIMR,(R0)   ;SET IMR COMMAND
1357 007104 111115      MOV      (R1),(R5)    ;READ DATA PORT
1358 007106 023715 001124      CMP      $GDDAT,(R5)
1359 007112 001401      BEQ      1$
1360 007117 104005      ERROR   5             ;ERROR IMR SHOULD BE SET
1361 007117 005304      DEC      R4           ;FINISHED BOTH GROUPS?
1362 007120 001405      BEQ      TST32        ;BR IF BOTH GROUPS TESTED
1363 007122 013700 001352      MOV      DRCSB,R0     ;SETUP FOR GROUP 2
1364 007125 013701 001356      MOV      DRCSB,R1
1365 007128 000751      BR       111$        ;DO IMR TEST WITH GROUP 2
  
```

 *TEST 32 TEST GROUP 1,2 CLEAR SINGLE IMR BIT INSTR.

```

TST32: SCOPE
1370 007134 000004      JSR      PC,CLRCSR    ;CLEAR ALL CSRS
1371 007136 004737 013002      MOV      #2,R4       ;GROUP COUNTER
1372 007142 012704 000002      MOV      #1,$SLPERR  ;SCOPE RETURN ADDRESS
1373 007146 012737 007204 001110  MOV      DRCSA,R0     ;CSRA ADDRESS
1374 007154 013700 001342      MOV      DRCSB,R1     ;CSRB ADDRESS
1375 007160 013701 001346      MOV      R1,$BDADR    ;STORE ADDRESS
1376 007164 010137 001122      MOV      #BGCHP4,R3   ;GOOD DATA PATTERN TABLE
1377 007170 012703 013254      MOV      #MIMR,(R0)   ;LOAD MODE BITS FOR IMR
1378 007174 112710 000244      MOV      #CSIMR,R2    ;CLEAR SINGLE IMR BIT VALUE
1379 007200 012702 000050      MOV      (R3),$GDDAT  ;STORE EXPECTED
1380 007204 111337 001124      MOV      #SIMR,(R0)   ;SET ALL IMR BITS
1381 007210 112710 000060      MOV      R2,(R0)     ;CLEAR SINGLE IMR BIT
1382 007214 110210      MOV      (R1),(R5)    ;READ DATA PORT
1383 007216 111115      CMP      $GDDAT,(R5)
1384 007220 023715 001124      BEQ      2$
1385 007224 001401      ERROR   5             ;IMR REG ERROR
1386 007226 104005      TST      (R3)+        ;INC EXPECTED DATA TABLE
1387 007230 005723      CMP      R3,#EDCHP4   ;CHECK FOR END
1388 007232 020327 013274      BEQ      3$
1389 007236 001402      INC      R2           ;SET UP TO CLEAR NEXT IMR BIT
1390 007240 005202      BR       1$          ;CLEAR NEXT IMR BIT
1391 007242 000760      DEC      R4           ;DO GROUPS 1 AND 2
1392 007244 005304      BEQ      TST33        ;BR IF BOTH GROUPS TESTED
1393 007246 001405      MOV      DRCSB,R0     ;CSRB ADDRESS
1394 007250 013700 001352      MOV      DRCSB,R1     ;CSRB ADDRESS
1395 007254 013701 001356      BR       111$        ;DO GROUP 2
1396 007260 000741
  
```

 *TEST 33 TEST GROUP 1,2 SET SINGLE IMR BIT INSTR.

```

TST33: SCOPE
1402 007262 000004      JSR      PC,CLRCSR    ;CLEAR ALL CSRS
1403 007264 004737 013002      MOV      #2,R4       ;GROUP COUNTER
1404 007270 012704 000002      MOV      #1,$SLPERR  ;SCOPE RETURN ADDRESS
1405 007274 012737 007332 001110  MOV      DRCSA,R0     ;CSRA ADDRESS
1406 007302 013700 001342
  
```

```
1407 007306 013701 001346      MOV      DRC5B,R1      ;CSR5 ADDRESS
1408 007312 010137 001122      111$:  MOV      R1,$5DADR  ;STORE ADDRESS
1409 007316 012703 013212      MOV      #5GCHP3,R3  ;GOOD DATA PATTERN TABLE
1410 007322 112710 000244      MOV8     #MIMR,(R0)  ;LOAD MODE BITS FOR IMR
1411 007326 012702 000070      MOV      #SSIMR,R2  ;SET SINGLE IMR BIT VALUE
1412 007332 111337 001124      1$:    MOV8     (R3),$GDDAT ;STORE EXPECTED
1413 007336 112710 000040      MOV8     #CIMR,(R0)  ;CLEAR ALL IMR BITS
1414 007342 110210      MOV8     R2,(R0)     ;SET SINGLE IMR BIT
1415 007344 111115      MOV8     (R1),(R5)   ;READ DATA PORT
1416 007346 023715 001124      CMP      $GDDAT,(R5)
1417 007352 001401      BEQ      2$
1418 007354 104005      ERROR   5           ;IMR REG ERROR
1419 007356 005723      2$:    TST      (R3)+    ;INC EXPECTED DATA TABLE
1420 007360 020327 013232      CMP      R3,#EDCHP3 ;CHECK FOR END
1421 007364 001402      BEQ      3$
1422 007366 005202      INC     R2           ;SET UP TO SET NEXT IMR BIT
1423 007370 000760      BR      1$          ;SET NEXT IMR BIT
1424 007372 005304      3$:    DEC     R4           ;DO GROUPS 1 AND 2
1425 007374 001405      BEQ     TST34       ;BR IF BOTH GROUPS TESTED
1426 007376 013700 001352      MOV     DRC5C,R0    ;CSRC ADDRESS
1427 007402 013701 001356      MOV     DRC5D,R1    ;CSR5 ADDRESS
1428 007406 000741      BR     111$        ;DO GROUP 2
```

```
::*****
:*TEST 34      TEST GROUP 1,2 SET IRR INSTR.
:*****
```

```
1432 TST34:  SCOPE
1433 007410 000004      JSR     PC,CLRCSR   ;CLEAR ALL CSRS
1434 007412 004737 013002      MOV     #2,R4       ;COUNTER FOR TWO GROUPS
1435 007416 012704 000002      MOV     DRC5A,R0    ;CSRA ADDRESS
1436 007422 013700 001342      MOV     DRC5B,R1    ;CSR5 ADDRESS
1437 007426 013701 001346      JSR     PC,CLRIRR   ;CLEAR IRR REGS WITH CHIP RESET
1438 007432 004737 013052      111$:  MOV     R1,$5DADR  ;STORE ADDRESS
1439 007436 010137 001122      MOV     #377,$GDDAT ;EXPECTED DATA
1440 007442 012737 000377 001124      MOV8    #MIRR,(R0)  ;LOAD MODE BITS TO READ IRR
1441 007450 112710 000250      MOV8    #SIRR,(R0)  ;SET IRR COMMAND
1442 007454 112710 000120      MOV8    (R1),(R5)   ;READ DATA PORT
1443 007460 111115      CMP     $GDDAT,(R5)
1444 007462 023715 001124      BEQ     1$
1445 007466 001401      ERROR   3           ;ERROR IRR SHOULD BE SET
1446 007470 104003      1$:    DEC     R4       ;FINISHED BOTH GROUPS?
1447 007472 005304      BEQ     TST35       ;BR IF BOTH GROUPS TESTED
1448 007474 001405      MOV     DRC5C,R0    ;SETUP FOR GROUP 2
1449 007476 013700 001352      MOV     DRC5D,R1
1450 007502 013701 001356      BR     111$        ;DO IRR TEST WITH GROUP 2
1451 007506 000753
```

```
::*****
:*TEST 35      TEST GROUP 1,2 CLEAR IRR INSTR.
:*****
```

```
1456 TST35:  SCOPE
1457 007510 000004      JSR     PC,CLRCSR   ;CLEAR ALL CSRS
1458 007512 004737 013002      MOV     #2,R4       ;COUNTER FOR TWO GROUPS
1459 007516 012704 000002      MOV     DRC5A,R0    ;CSRA ADDRESS
1460 007522 013700 001342      MOV     DRC5B,R1    ;CSR5 ADDRESS
1461 007526 013701 001346      JSR     PC,CLRIRR   ;CLEAR IRR REGS WITH CHIP RESET
1462 007532 004737 013052
```



```

1463 007536 010137 001122      111$: MOV R1,$BDADR      ;STORE ADDRESS
1464 007542 005037 001124      CLR $GDDAT      ;EXPECTED DATA
1465 007546 112710 000250      MOVB #MIRR,(R0) ;LOAD MODE BITS TO READ IRR
1466 007552 112710 000120      MOVB #SIRR,(R0) ;SET IRR BITS
1467 007556 112710 000100      MOVB #CIRR,(R0) ;CLEAR IRR COMMAND
1468 007562 111115      MOVB (R1),(R5)  ;READ DATA PORT
1469 007564 023715 001124      CMP $GDDAT,(R5)
1470 007570 001401      BEQ 1$
1471 007572 104003      ERROR 3        ;ERROR ,IRR SHOULD BE CLEARED
1472 007574 005304      1$: DEC R4        ;FINISHED BOTH GROUPS?
1473 007576 001405      BEQ TST36      ;BR IF BOTH GROUPS TESTED
1474 007600 013700 001352      MOV DRCSA,R0   ;SETUP FOR GROUP 2
1475 007604 013701 001356      MOV DRCSB,R1
1476 007610 000752      BR 111$       ;DO IRR TEST WITH GROUP 2
1477
1478
1479
1480

```

```

:*****
:*TEST 36 TEST GROUP 1,2 CLEAR SINGLE IRR BIT INSTR.
:*****

```

```

1481 007612 000004      TST36: SCOPE
1482 007614 004737 013002      JSR PC,CLRCSR ;CLEAR ALL CSRS
1483 007620 012704 000002      MOV #2,R4     ;GROUP COUNTER
1484 007624 012737 007666 0011 0  MOV #1$,$LPERR ;SCOPE RETURN ADDRESS
1485 007632 013700 001342      MOV DRCSA,R0 ;CSRA ADDRESS
1486 007636 013701 001346      MOV DRCSB,R1 ;CSRB ADDRESS
1487 007642 004737 013052      JSR PC,CLRIRR ;CLEAR IRR REGS WITH CHIP RESET
1488 007646 010137 001122      111$: MOV R1,$BDADR ;STORE ADDRESS
1489 007652 012703 013254      MOV #BGCHP4,R3 ;GOOD DATA PATTERN TABLE
1490 007656 112710 000250      MOVB #MIRR,(R0) ;LOAD MODE BITS FOR IRR
1491 007662 012702 000110      MOV #CSIRR,R2 ;CLEAR SINGLE IRR BIT VALUE
1492 007666 111337 001124      1$: MOVB (R3),$GDDAT ;STORE EXPECTED
1493 007672 112710 000120      MOVB #SIRR,(R0) ;SET ALL IRR BITS
1494 007676 110210      MOVB R2,(R0)   ;CLEAR SINGLE IRR BIT
1495 007700 111115      MOVB (R1),(R5) ;READ DATA PORT
1496 007702 023715 001124      CMP $GDDAT,(R5)
1497 007706 001401      BEQ 2$
1498 007710 104003      ERROR 3        ;IRR REG ERROR
1499 007712 005723      2$: TST (R3)+    ;INC EXPECTED DATA TABLE
1500 007714 020327 013274      CMP R3,#EDCHP4 ;CHECK FOR END
1501 007720 001402      BEQ 3$
1502 007722 005202      INC R2        ;SET UP TO CLEAR NEXT IRR BIT
1503 007724 000760      BR 1$        ;CLEAR NEXT IRR BIT
1504 007726 005304      3$: DEC R4        ;DO GROUPS 1 AND 2
1505 007730 001405      BEQ TST37    ;BR IF BCTH GROUPS TESTED
1506 007732 013700 001352      MOV DRCSA,R0 ;CSRA ADDRESS
1507 007736 013701 001356      MOV DRCSB,R1 ;CSRB ADDRESS
1508 007742 000741      BR 111$     ;DO GROUP 2
1509
1510
1511
1512

```

```

:*****
:*TEST 37 TEST GROUP 1,2 SET SINGLE IRR BIT INSTR.
:*****

```

```

1513 007744 000004      TST37: SCOPE
1514 007746 004737 013002      JSR PC,CLRCSR ;CLEAR ALL CSRS
1515 007752 012704 000002      MOV #2,R4     ;GROUP COUNTER
1516 007756 012737 010020 001110  MOV #1$,$LPERR ;SCOPE RETURN ADDRESS
1517 007764 013700 001342      MOV DRCSA,R0 ;CSRA ADDRESS
1518 007770 013701 001346      MOV DRCSB,R1 ;CSRB ADDRESS

```

```
1519 007774 004737 013052 JSR PC,CLRIRR ;CLEAR IRR REGS WITH CHIP RE ET
1520 010000 010137 001122 111$: MOV R1,$BDADR ;STORE ADDRESS
1521 010004 012703 013212 MOV #BGCHP3,R3 ;GOOD DATA PATTERN TABLE
1522 010010 112710 000250 MOVB #MIRR,(R0) ;LOAD MODE BITS FOR IRR
1523 010014 012702 000130 MOV #SSIRR,R2 ;SET SINGLE IRR BIT VALUE
1524 010020 111337 001124 1$: MOVB (R3),$GDDAT ;STORE EXPECTED
1525 010024 112710 000100 MOVB #CIRR,(R0) ;CLEAR ALL IRR BITS
1526 010030 110210 MOVB R2,(R0) ;SET SINGLE IRR BIT
1527 010032 111115 MOVB (R1),(R5) ;READ DATA PORT
1528 010034 023715 001124 CMP $GDDAT,(R5)
1529 010040 001401 BEQ 2$
1530 010042 104003 ERROR 3 ;IRR REG ERROR
1531 010044 005723 2$: TST (R3)+ ;INC EXPECTED DATA TABLE
1532 010046 020327 013232 CMP R3,#EDCHP3 ;CHECK FOR END
1533 010052 001402 BEQ 3$
1534 010054 005202 INC R2 ;SET UP TO SET NEXT IRR BIT
1535 010056 000760 BR 1$ ;SET NEXT IRR BIT
1536 010060 005304 3$: DEC R4 ;DO GROUPS 1 AND 2
1537 010062 001405 BEQ TST40 ;BR IF BOTH GROUPS TESTED
1538 010064 013700 001352 MOV DRCSA,R0 ;CSRA ADDRESS
1539 010070 013701 001356 MOV DRCSB,R1 ;CSRB ADDRESS
1540 010074 000741 BR 111$ ;DO GROUP 2
```

```
1541
1542
1543 :*****
1544 :*TEST 40 TEST GROUP 1,2 CLEAR IRR+IMR INSTR.
1545 :*****
```

```
TST40: SCOPE
1546 010100 004737 013002 JSR PC,CLRCR ;CLEAR ALL CSRS
1547 010104 012704 000002 MOV #2,R4 ;COUNTER FOR TWO GROUPS
1548 010110 013700 001342 MOV DRCSA,R0 ;CSRA ADDRESS
1549 010114 013701 001346 MOV DRCSB,R1 ;CSRB ADDRESS
1550 010120 004737 013052 JSR PC,CLRIRR ;CLEAR IRR REGS WITH CHIP RESET
1551 010124 010137 001122 111$: MOV R1,$BDADR ;STORE ADDRESS
1552 010130 005037 001124 CLR $GDDAT ;EXPECTED DATA
1553 010134 112710 000250 MOVB #MIRR,(R0) ;LOAD MODE BITS TO READ IRR
1554 010140 112710 000120 MOVB #SIRR,(R0) ;SET IRR BITS
1555 010144 112710 000060 MOVB #SIMR,(R0) ;SET IMR BITS
1556 010150 112710 000020 MOVB #CIRMR,(R0) ;CLEAR IRR+IMR COMMAND
1557 010154 111115 MOVB (R1),(R5) ;READ DATA PORT FOR IRR
1558 010156 023715 001124 CMP $GDDAT,(R5)
1559 010162 001401 BEQ 1$
1560 010164 104003 ERROR 3 ;ERROR ,IRR SHOULD BE CLEARED
1561 010166 112710 000244 1$: MOVB #MIMR,(R0) ;LOAD MODE BITS TO READ IMR
1562 010172 111115 MOVB (R1),(R5) ;READ IMR REG
1563 010174 023715 001124 CMP $GDDAT,(R5)
1564 010200 001401 BEQ 2$
1565 010202 104005 ERROR 5 ;IRR+IMR COMMAND DID NOT CLEAR IMR
1566 010204 005304 2$: DEC R4 ;FINISHED BOTH GROUPS?
1567 010206 001405 BEQ TST41 ;BR IF BOTH GROUPS TESTED
1568 010210 013700 001352 MOV DRCSA,R0 ;SETUP FOR GROUP 2
1569 010214 013701 001356 MOV DRCSB,R1
1570 010220 000741 BR 111$ ;DO IRR+IMR TEST WITH GROUP 2
```

```
1571
1572 :*****
1573 :*TEST 41 TEST GROUP 1,2 CLEAR SINGLE IRR+IMR BIT INSTR.
1574 :*****
```

```

1575 010222 000004          TST41: SCOPE
1576 010224 004737 013002      JSR    PC,CLRCR      :CLEAR ALL CSRS
1577 010230 012704 000002      MOV    #2,R4        :GROUP COUNTER
1578 010234 012737 010272 001110  MOV    #1$,SLPERR    :SCOPE RETURN ADDRESS
1579 010242 013700 001342      MOV    DRCSA,R0     :CSRA ADDRESS
1580 010246 013701 001346      MOV    DRCSB,R1     :CSRB ADDRESS
1581 010252 004737 013052      JSR    PC,CLRIRR    :CLEAR IRR REGS WITH CHIP RESET
1582 010256 010137 001122 111$:  MOV    R1,$BDADR     :STORE ADDRESS
1583 010262 012703 013254      MOV    #BGCHP4,R3   :GOOD DATA PATTERN TABLE
1584 010266 012702 000030      MOV    #CSIMR,R2    :CLEAR SINGLE IRR+IMR BIT VALUE
1585 010272 111337 001124 1$:    MOVB   (R3),$GDDA    :STORE EXPECTED
1586 010276 112710 000250      MOVB   #MIRR,(R0)   :LOAD MODE BITS TO READ IRR
1587 010302 112710 000120      MOVB   #SIRR,(R0)   :SET ALL IRR BITS
1588 010306 112710 000060      MOVB   #SIMR,(R0)   :SET ALL IMR BITS
1589 010312 110210          MOVB   R2,(R0)      :CLEAR SINGLE IRR+IMR BIT
1590 010314 111115          MOVB   (R1),(R5)    :READ DATA PORT FOR IRR
1591 010316 023715 001124      CMP    $GDDAT,(R5)
1592 010322 001401          BEQ    2$
1593 010324 104003          ERROR  3           :IRR REG ERROR
1594 010326 112710 000244 2$:    MOVB   #MIMR,(R0)   :SET UP TO READ IMR
1595 010332 111115          MOVB   (R1),(R5)    :READ IMR REGISTER
1596 010334 023715 001124      CMP    $GDDAT,(R5)
1597 010340 001401          BEQ    3$
1598 010342 104005          ERROR  5           :IMR REG ERROR
1599 010344 005723 013274 3$:    TST    (R3)+        :INC EXPECTED DATA TABLE
1600 010346 020327          CMP    R3,#EDCHP4   :CHECK FOR END
1601 010352 001402          BEQ    4$
1602 010354 005202          INC    R2           :SET UP TO CLEAR NEXT IRR+IMR BIT
1603 010356 000745          BR     1$          :CLEAR NEXT IRR+IMR BIT
1604 010360 005304          DEC    R4           :DO GROUPS 1 AND 2
1605 010362 001405          BEQ    TST42       :BR IF BOTH GROUPS TESTED
1606 010364 013700 001352      MOV    DRCSA,R0     :CSRA ADDRESS
1607 010370 013701 001356      MOV    DRCSB,R1     :CSRB ADDRESS
1608 010374 000730          BR     111$       :DO GROUP 2

```

```

*****
*TEST 42      TEST GROUPS 1,2 FOR GROUP UNIQUENESS
*****

```

```

1613 010376 000004          TST42: SCOPE
1614 010400 004737 013002      JSR    PC,CLRCR      :CLEAR ALL CSRS
1615          :CSRA = R0
1616          :CSRB = R1
1617          :CSRC = R2
1618          :CSRD = R3
1619 010404 012704 000002      MOV    #2,R4        :COUNTER FOR TESTING TWO GROUPS
1620 010410 004737 013052      JSR    PC,CLRIRR    :CLEAR IRR REGS WITH CHIP RESET
1621 010414 010337 001122 111$:  MOV    R3,$BDADR     :STORE ADDRESS
1622 010420 112710 000300      MOVB   #PACR,(R0)   :PRESELECT ACR FOR WRITING
1623 010424 112711 000377      MOVB   #377,(R1)    :WRITE INTO DATA PORT FOR ACR
1624 010430 112710 000120      MOVB   #SIRR,(R0)   :SET IRR TO ALL 1'S
1625 010434 112710 000040      MOVB   #CIMR,(R0)   :CLEAR IMR
1626 010440 112712 000254      MOVB   #MACR,(R2)   :LOAD MODE TO READ ACR
1627 010444 005037 001124      CLR    $GDDAT      :EXPECTED
1628 010450 111315          MOVB   (R3),(R5)    :STORE DATA FOR COMPARE
1629 010452 023715 001124      CMP    $GDDAT,(R5) :CHECK ACR RESULTS
1630 010456 001401          BEQ    1$

```

```

1631 010460 104004          ERROR 4          ;ERROR,OTHER GROUP ACR SHOULD BE CLEARED
1632 010462 112712 000244 1$: MOVB #MIMR,(R2) ;CHANGE TO IMR REGISTER
1633 010466 012737 000377 001124 MOV #377,$GDDAT ;STORE EXPECTED
1634 010474 111315          MOVB (R3),(R5) ;READ IMR
1635 010476 023715 001124 CMP $GDDAT,(R5) ;SHOULD BE ALL 1'S
1636 010502 001401          BEQ 2$
1637 010504 104005          ERROR 5          ;ERROR,OTHER GROUP IMR SHOULD BE SET
1638 010506 112712 000240 2$: MOVB #MISR,(R2) ;LOAD MODE BITS FOR ISR
1639 010512 005037 001124 CLR $GDDAT ;STORE EXPECTED
1640 010516 111315          MOVB (R3),(R5) ;READ ISR
1641 010520 023715 001124 CMP $GDDAT,(R5) ;ISR SHOULD BE CLEARED
1642 010524 001401          BEQ 3$
1643 010526 104006          ERROR 6          ;ISR REG ERROR
1644 010530 112712 000250 3$: MOVB #MIRR,R2 ;LOAD MODE BITS FOR IRR
1645 010534 111315          MOVB (R3),(R5) ;READ IRR
1646 010536 023715 001124 CMP $GDDAT,(R5) ;IRR SHOULD BE CLEARED
1647 010542 001401          BEQ 4$
1648 010544 104003          ERROR 3          ;ERROR,OTHER GROUP IRR SHOULD BE CLEARED
1649 010546 005304 4$: DEC R4 ;TEST GROUPS 1 AND 2
1650 010550 001415          BEQ TST43 ;BR IF BOTH GROUPS TESTED
1651 010552 004737 013002 JSR PC,CLRCSR ;CLEAR ALL CSRS
1652 010556 013700 001352 MOV DRCSA,R0 ;GROUP 2 CONTROL PORT
1653 010562 013701 001356 MOV DRCSB,R1 ;GROUP 2 DATA PORT
1654 010566 013702 001342 MOV DRCSA,R2 ;GROUP 1 CONTROL PORT
1655 010572 013703 001346 MOV DRCSB,R3 ;GROUP 1 DATA PORT
1656 010576 004737 013052 JSR PC,CLRIFR ;CLEAR IRR RECS WITH CHIP RESET
1657 010602 000704          BR 111$ ;TEST GROUP 2 ACR UNIQUENESS

```

```

*****
*TEST 43 TEST STATUS BITS GIN1,S2,S1,S0,GP1,2
*****

```

```

1662 010604 000004          TST43: SCOPE
1663 010606 004737 013002 JSR PC,CLRCSR ;CLEAR ALL CSRS
1664 010612 013700 001342 MOV DRCSA,R0
1665 010616 013701 001346 MOV DRCSB,R1
1666 010622 012703 000002 MOV #2,R3 ;DO TWO GROUPS
1667 010626 012704 000120 111$: MOV #120,R4 ;EXPECTED STATUS BITS
1668 010632 005077 170504 CLR @DRCSA ;INIT CSRA
1669 010636 005077 170510 CLR @DRCSB ;INIT CSRB
1670 010642 012702 013112 MOV #BGPAT1,R2 ;EXPECTED IRR PATTERN
1671 010646 112760 000001 000001 MOVB #BIT0,1(R0) ;CSR TO OUTPUT MODE
1672 010654 112777 000204 170460 MOVB #204,@DRCSA ;POLLED MODE FOR CSRA,GROUP 1
1673 010662 112777 000204 170462 MOVB #204,@DRCSB ;POLLED MODE FOR CSRB,GROUP 2
1674 010670 112710 000020 MOVB #CIMR,(R0) ;CLEAR IMR + IRR
1675 010674 012737 000070 001372 MOV #SSIMR,IMRLOC ;STORE CODE FOR SINGLE IMR
1676 010702 012737 000130 001376 MOV #SSIRR,IRRLOC ;STORE CODE FOR SINGLE IRR
1677 010710 010037 001122 112$: MOV R0,$BDADR ;CSR CHIP COMMAND ADDRESS
1678 010714 010437 001124 MOV R4,$GDDAT ;EXPECTED DATA
1679 010720 113710 001376 MOVB IRRLOC,(R0) ;SET SINGLE IRR BIT
1680 010724 111015          MOVB (R0),(R5) ;CHECK STATUS
1681 010726 023715 001124 CMP $GDDAT,(R5)
1682 010732 001401          BEQ 1$
1683 010734 104007          ERROR 7          ;CHIP STATUS ERROR
1684 010736 005037 001124 1$: CLR $GDDAT
1685 010742 010137 001122 MOV R1,$BDADR ;CSR CHIP DATA ADDRESS
1686 010746 111237 001124 MOVB (R2),$GDDAT

```

```

1687 010752 112710 000250      MOVB #MIRR,(R0)      ;READ IRR
1688 010756 111115      MOVB (R1),(R5)
1689 010760 023715 001124      CMP $GDDAT,(R5)     ;CHECK IRR
1690 010764 001401      BEQ 2$
1691 010766 104003      ERROR 3             ;IRR ERROR
1692 010770 010037 001122      2$: MOV RC,$BDADR
1693 010774 113710 001372      MOVB IMRLOC,(R0)    ;SET IMR BIT
1694 011000 012737 000320 001124      MOV #320,$GDDAT     ;CSR EXPECTED DATA
1695 011006 111015      MOVB (R0),(R5)
1696 011010 042715 000007      BIC #7,(R5)         ;CLEAR UNDEFINED BITS
1697 011014 023715 001124      CMP $GDDAT,(R5)
1698 011020 001401      BEQ 3$
1699 011022 104007      ERROR 7             ;CHIP STATUS ERROR
1700 011024 010137 001122      3$: MOV R1,$BDADR
1701 011030 011237 001124      MOV (R2),$GDDAT     ;EXPECTED DATA
1702 011034 112710 000244      MOVB #MIMR,(R0)     ;READ IMR BITS
1703 011040 111115      MOVB (R1),(R5)       ;SAVE IMR READ
1704 011042 023715 001124      CMP $GDDAT,(R5)
1705 011046 001401      BEQ 4$
1706 011050 104005      ERROR 5             ;IMR ERROR
1707 011052 005722      4$: TST (R2)+         ;NEXT EXPECTED FOR IMR + IRR
1708 011054 020227 013132      CMP R2,#EDCHP1     ;CHECK FOR END
1709 011060 001407      BEQ 5$
1710 011062 005237 001376      INC IRRLOC          ;NEXT IRR BIT
1711 011066 005237 001372      INC IMRLOC          ;NEXT IMR BIT
1712 011072 005204      INC R4              ;INDEX EXPECTED STATUS
1713 011074 000137 010710      JMP 112$            ;DO NEXT STATUS CHECK
1714 011100 005303      5$: DEC R3             ;FINISHED BOTH GROUPS?
1715 011102 001406      BEQ TST44           ;:BR IF EQUAL
1716 011104 013700 001352      MOV DRCSO,R0
1717 011110 013701 001356      MOV DRCSO,R1
1718 011114 000137 010626      JMP 111$            ;DO NEXT GROUP
1719
1720
1721
1722
1723
1724 011120 000004      *****
1725 011122 004737 013002      *TEST 44 TEST POLLED MODE:CSRS A,B OUT C,D IN,ACTIVE LOW
1726
1727
1728
1729
1730 011126 012737 011170 001110      TST44: SCOPE
1731 011134 012704 013110      JSR PC,CLRCSR       ;CLEAR ALL CSRS
1732 011140 112760 000001 000001      ;R0 = CSRA-GROUP 1 CONTROL
1733 011146 112761 000001 000001      ;R1 = CSRB-GROUP 1 DATA
1734
1735 011154 105010      ;R2 = CSRC-GROUP 2 CONTROL
1736 011156 105012      ;R3 = CSRD-GROUP 2 DATA
1737 011160 112710 000204      MOV #1$,$LPERR      ;SET FOR SCOPE RETURN
1738 011164 112712 000204      MOV #BEGPAT,R4      ;START OF PATTERN TABLE
1739 011170 011460 000002      MOVB #BIT0,1(R0)    ;SET CSRA TO OUTPUT MODE
1740 011174 112710 000020      MOVB #BIT0,1(R1)    ;SET CSRB TO OUTPUT MODE
1741 011200 112712 000020      CLRB (R0)           ;CHIP RESET GROUP 1 CSRA
1742 011204 005060 000002      CLRB (R2)           ;CHIP RESET GROUP 2 CSRC
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
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
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
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

```

```

1743 011210 010137 001122 2$: MOV R1,$BDADR ; GROUP 1 DATA PORT
1744 011214 112710 000250 MOVB #MIRR,(R0) ;LOAD BITS TO READ IRR
1745 011220 111437 001124 MOVB (R4),$GDDAT
1746 011224 111115 MOVB (R1),(R5) ;READ IRR,GROUP 1
1747 011226 023715 001124 CMP $GDDAT,(R5)
1748 011232 001401 BEQ 3$
1749 011234 104003 ERROR 3 ;IRR ERROR,GROUP 1
1750 011236 010337 001122 3$: MOV R3,$BDADR ;GROUP 2 DATA PORT
1751 011242 112712 000250 MOVB #MIRR,(R2) ;LOAD MODE BITS TO READ IRR GROUP2
1752 011246 116437 000001 001124 MOVB 1(R4),$GDDAT ;BUILD EXPECTED DATA
1753 011254 042737 000360 001124 BIC #360,$GDDAT ;SAVE BITS 0-3
1754 011262 052737 000100 001124 BIS #100,$GDDAT ;EXPECTED 0-3,URPLY 6(C)
1755 011270 111315 MOVB (R3),(R5) ;READ IRR BITS,GROUP 2
1756 011272 023715 001124 CMP $GDDAT,(R5)
1757 011276 001401 BEQ 4$
1758 011300 104003 ERROR 3 ;IRR ERROR,GROUP 2
1759 011302 005762 000002 4$: TST 2(R2) ;READ DBRC FOR RPLY 4(A)
1760 011306 052737 000020 001124 BIS #20,$GDDAT ;STORE EXPECTED
1761 011314 111315 MOVB (R3),(R5) ;READ IRR BITS,GROUP 2
1762 011316 023715 001124 CMP $GDDAT,(R5)
1763 011322 001401 BEQ 5$
1764 011324 104003 ERROR 3 ;IRR ERROR,GROUP 2
1765 011326 005061 000002 5$: CLR 2(R1) ;CLEAR DBRB FOR RPY 7(D)
1766 011332 052737 000200 001124 BIS #200,$GDDAT ;EXPECTED
1767 011340 111315 MOVB (R3),(R5) ;READ IRR BITS GROUP 2
1768 011342 023715 001124 CMP $GDDAT,(R5)
1769 011346 001401 BEQ 6$
1770 011350 104003 ERROR 3 ;IRR ERROR,GROUP 2
1771 011352 005763 000002 6$: TST 2(R3) ;READ DBRD FOR RPLY 5(R)
1772 011356 052737 000040 001124 BIS #40,$GDDAT
1773 011364 111315 MOVB (R3),(R5) ;READ IRR BITS GROUP2
1774 011366 023715 001124 CMP $GDDAT,(R5)
1775 011372 001401 BEQ 7$
1776 011374 104003 ERROR 3 ;GET NEXT PATTERN
1777 011376 005724 001124 7$: TST (R4)+ ;IRR ERROR,GROUP 2
1778 011400 020427 013410 CMP R4,#ENDDAT ;INDEX DATA TABLE
1779 011404 001271 BNE 1$ ;CHECK FOR END
1780 ;DO NEXT PATTERN

```

```

*****
*TEST 45 TEST GROUPS 1,2 IN POLLED MODE,NO REPLY
*****

```

```

1785 011406 000004 013002 TST45: SCOPE
1786 011410 004737 JSR PC,CLRCR ;CLEAR ALL CSRS
1787 ;R0 = CSRA-GROUP 1 CONTROL
1788 ;R1 = CSRB-GROUP 1 DATA
1789 ;R2 = CSRC-GROUP 2 CONTROL
1790 ;R3 = CSRD-GROUP 2 DATA
1791 011414 012704 000002 MOV #2,R4 ;TWO PASSES
1792 ;FIRST PASS CSRA,CSRB = OUTPUT
1793 ; CSRC,CSRD = INPUT
1794 ;SEC PASS CSRC,CSRD = OUTPUT
1795 ; CSRA,CSRB = INPUT
1796 011420 112760 000001 000001 111$: MOVB #BIT0,1(R0) ;SET CSR TO OUTPUT MODE
1797 011426 112761 000001 000001 MOVB #BIT0,1(R1) ;SET CSR TO OUTPUT MODE
1798 011434 012760 177777 000002 MOV #-1,2(R0) ;SET ALL ONES DBR FOR H TO

```

```
1799 011442 105010 CLRB (R0) ;CHIP RESET GROUP CSR
1800 011444 105012 CLRB (R2) ;CHIP RESET GROUP CSR
1801 011446 112710 000204 MOVB #204,(R0) ;LOAD MODE BITS FOR POLLED MODE
1802 011452 112712 000204 MOVB #204,(R2) ;LOAD MODE BITS FOR POLLED MODE
1803 011456 112710 000020 MOVB #CIRMR,(R0) ;CLEAR IMR+IRR
1804 011462 112712 000020 MOVB #CIRMR,(R2) ;CLEAR IMR+IRR
1805 011466 005760 000002 TST 2(R0) ;READ DBR IN OUTPUT MODE,NO REPLY
1806 011472 012737 000320 001124 MOV #320,$GDDAT ;STORE EXPECTED
1807 011500 010037 001122 MOV R0,$BDADR ;STORE ADDRESS
1808 011504 111015 MOVB (R0),(R5) ;READ STATUS
1809 011506 042715 000007 BIC #7,(R5) ;CLEAR UNDEFINED BITS
1810 011512 023715 001124 CMP $GDDAT,(R5) ;CHECK STATUS
1811 011516 001401 BEQ 1$
1812 011520 104007 ERROR 7 ;CHIP STATUS ERROR
1813 011522 010237 001122 1$: MOV R2,$BDADR ;STORE ADDRESS
1814 011526 111215 MOVB (R2),(R5) ;READ STATUS
1815 011530 042715 000007 BIC #7,(R5) ;CLEAR UNDEFINED BITS
1816 011534 023715 001124 CMP $GDDAT,(R5)
1817 011540 001401 BEQ 2$
1818 011542 104007 ERROR 7 ;CHIP STATUS ERROR
1819 011544 012762 000000 000002 2$: MOV #0,2(R2) ;WRITE ONLY, DBR INPUT MODE FOR NO REPLY
1820 011552 005761 000002 TST 2(R1) ;READ DBR OUTPUT MODE,NO REPLY
1821 011556 012763 000000 000002 MOV #0,2(R3) ;WRITE ONLY, DBR INPUT MODE,NO REPLY
1822 011564 010137 001122 MOV R1,$BDADR ;CHIP DATA PORT
1823 011570 112710 000250 MOVB #MIRR,(R0) ;LOAD BITS TO READ IRR
1824 011574 005037 001124 CLR $GDDAT ;IRR SHOULD BE ZERO
1825 011600 111115 MOVB (R1),(R5) ;READ IRR
1826 011602 023715 001124 CMP $GDDAT,(R5)
1827 011606 001401 BEQ 3$
1828 011610 104003 ERROR 3 ;IRR ERROR
1829 011612 010337 001122 3$: MOV R3,$BDADR ;CHIP DATA PORT
1830 011616 112712 000250 MOVB #MIRR,(R2) ;LOAD MODE BITS TO READ IRR
1831 011622 111315 MOVB (R3),(R5) ;READ IRR BITS
1832 011624 023715 001124 CMP $GDDAT,(R5)
1833 011630 001401 BEQ 4$
1834 011632 104003 ERROR 3 ;IRR ERROR
1835 011634 005304 4$: DEC R4 ;FINISHED TWO PASSES.
1836 011636 001413 BEQ TST46 ;BR IF EQUAL
1837 011640 004737 013002 JSR PC,CLRCR ;CLEAR ALL CSRS
1838 011644 013700 001352 MOV DRCSA,R0 ;SET UP FOR NEXT PASS
1839 011650 013701 001356 MOV DRCSB,R1
1840 011654 013702 001342 MOV DRCSA,R2
1841 011660 013703 001346 MOV DRCSB,R3
1842 011664 000655 BR 111$ ;DO NEXT PASS
1843
1844
1845
1846
1847 011666 000004 TST46: SCOPE
1848 011670 004737 013002 ISR PC,CLRCR ;CLEAR ALL CSRS
1849 ;R0 = CSRA-GROUP 1 CONTROL
1850 ;R1 = CSRB-GROUP 1 DATA
1851 ;R2 = CSRC-GROUP 2 CONTROL
1852 ;R3 = CSRD-GROUP 2 DATA
1853 011674 012737 011736 001110 MOV #1,$LPEHR ;SCOPE LOOP RETURN
1854 011702 012704 013110 MOV #BEGPAT,R4 ;PATTERN TABLE
```

```

1855 011706 112762 000001 000001      MOVB  #BIT0,1(R2)      ;SET CSRC TO OUTPUT MODE
1856 011714 112763 000001 000001      MOVB  #BIT0,1(R3)      ;SET CSRD TO OUTPUT MODE
1857 011722 105010                (LRB  (R0)              ;CHIP RESET GROUP 1 CSRA
1858 011724 105012                (LRB  (R2)              ;CHIP RESET GROUP 2 CSRC
1859 011726 112710 000204      MOVB  #204,(R0)        ;LOAD MODE-BITS FOR POLLED MODE,GR 1
1860 011732 112712 000204      MOVB  #204,(R2)        ;LOAD MODE BITS FOR POLLED MODE,GR2
1861 011736 011462 000002      1$: MOV  (R4),2(R2)      ;SET PATTERN DBRC FOR H TO L
1862 011742 112710 000020      MOVB  #CIRMR,(R0)     ;CLEAR IMR+IRR GROUP 1
1863 011746 112712 000020      MOVB  #CIRMR,(R2)     ;CLEAR IMR+IRR GROUP 2
1864 011752 005062 000002      CLR  2(R2)            ;CLEAR DBRC,ACTIVE LOW WILL SET RPLY A
1865 011756 010137 001122      2$: MOV  R1,$BDADR     ;GROUP 1 DATA PORT
1866 011762 112710 000250      MOVB  #MIRR,(R0)      ;LOAD BITS TO READ IRR
1867 011766 111437 001124      MOVB  (R4),$GDDAT
1868 011772 111115                MOVB  (R1),(R5)        ;READ IRR,GROUP 1
1869 011774 023715 001124      CMP  $GDDAT,(R5)
1870 012000 001401                BEQ  3$
1871 012002 104003                ERROR 3                ;IRR ERROR,GROUP 1
1872 012004 010337 001122      3$: MOV  R3,$BDADR     ;GROUP 2 DATA PORT
1873 012010 112712 000250      MOVB  #MIRR,(R2)      ;LOAD MODE BITS TO READ IRR GROUP2
1874 012014 116437 000001 001124      MOVB  1(R4),$GDDAT
1875 012022 042737 000360 001124      BIC  #360,$GDDAT     ;SAVE BITS 0-3
1876 012030 052737 000020 001124      BIS  #20,$GDDAT     ;EXPECTED 0-3,URPLY 4(A)
1877 012036 111315                MOVB  (R3),(R5)        ;READ IRR BITS,GROUP 2
1878 012040 023715 001124      (MP  $GDDAT,(R5)
1879 012044 001401                BEQ  4$
1880 012046 104003                ERROR 3                ;IRR ERROR,GROUP 2
1881 012050 005760 000002      4$: TST  2(R0)          ;READ DBRA FOR RPLY 6(C)
1882 012054 052737 000100 001124      BIS  #100,$GDDAT     ;STORE EXPECTED
1883 012062 111315                MOVB  (R3),(R5)        ;READ IRR BITS,GROUP 2
1884 012064 023715 001124      (MP  $GDDAT,(R5)
1885 012070 001401                BEQ  5$
1886 012072 104003                ERROR 3                ;IRR ERROR,GROUP 2
1887 012074 005063 000002      5$: CLR  2(R3)          ;CLEAR DBRD FOR RPY 5(B)
1888 012100 052737 000040 001124      BIS  #40,$GDDAT     ;EXPECTED
1889 012106 111315                MOVB  (R3),(R5)        ;READ IRR BITS GROUP 2
1890 012110 023715 001124      (MP  $GDDAT,(R5)
1891 012114 001401                BEQ  6$
1892 012116 104003                ERROR 3                ;IRR ERROR,GROUP 2
1893 012120 005761 000002      6$: TST  2(R1)          ;READ DBRB FOR RPLY 7(D)
1894 012124 052737 000200 001124      BIS  #200,$GDDAT
1895 012132 111315                MOVB  (R3),(R5)        ;READ IRR BITS GROUP2
1896 012134 023715 001124      (MP  $GDDAT,(R5)
1897 012140 001401                BEQ  7$
1898 012142 104003                ERROR 3                ;IRR ERROR,GROUP 2
1899 012144 005724      7$: TST  (R4)+          ;INDEX DATA TABLE
1900 012146 020427 013410      (MP  R4,#ENDDAT     ;CHECK FOR END
1901 012152 001271                BNE  1$                ;DO NEXT PATTERN

```

```

*****
: *TEST 47      TEST IRR BITS WITH DATA PAT.,POLLED MODE,ACT. HIGH
*****

```

```

1908 012154 000004      TST47: SCOPE
1909 012156 004737 013002      JSR  PC,CLRCSR      ;CLEAR ALL CSRS
1910 012162 012703 013110      MOV  #BEGPAT,R3    ;GET DATA PATTERN TABLE

```



```

1911 012166 012737 000001 001110      MOV      #1,$LPERR      ;SET UP SCOPE ADDRESS
1912 012174 013700 001342      MOV      DRCSA,R0
1913 012200 013701 001346      MOV      DRCSB,R1
1914 012204 013702 001356      MOV      DRCSB,R1
1915 012210 112760 000001 000001      MOVB    #BIT0,1(R0)    ;CSRA OUTPUT MODE
1916 012216 112761 000001 000001      MOVB    #BIT0,1(R1)    ;CSRB OUTPUT MODE
1917 012224 112710 000224      MOVB    #224,(R0)      ;LOAD MODE BITS FOR POLLED MODE,GP1
1918 012230 112760 000224 000010      MOVB    #224,10(R0)    ;LOAD MODE BITS FOR POLLED MODE,GP2
1919 012236 005060 000002      1$: CLR    2(R0)          ;CLEAR DBRA
1920 012242 112710 000020      MOVB    #CIRMR,(R0)    ;CLEAR IMR +IRR GROUP 1
1921 012246 112760 000020 000010      MOVB    #CIRMR,10(R0)  ;CLEAR IMR + IRR GROUP 2
1922 012254 011360 000002      MOV      (R3),2(R0)    ;STORE PATTERN INTO DBRA
1923 012260 010137 001122      MOV      R1,$BDADR     ;CSRB ADDRESS
1924 012264 112710 000250      MOVB    #MIRR,(R0)     ;LOAD BITS TO READ IRR1
1925 012270 111337 001124      MOVB    (R3),$GDDAT
1926 012274 111115      MOVB    (R1),(R5)     ;READ IRR,GP1
1927 012276 023715 001124      CMP     $GDDAT,(R5)
1928 012302 001401      BEQ     2$
1929 012304 104003      ERROR   3             ;IRR ERROR,GP1
1930 012306 010237 001122      2$: MOV    R2,$BDADR     ;CSRD ADDRESS
1931 012312 112760 000250 000010      MOVB    #MIRR,10(R0)   ;SET MODE TO READ IRR BITS GROUP 2
1932 012320 116337 000001 001124      MOVB    1(R3),$GDDAT   ;BUILD EXPECTED DATA
1933 012326 042737 000360 001124      BIC     #360,$GDDAT    ;BITS 0-3 EXPECTED
1934 012334 052737 000100 001124      BIS     #100,$GDDAT   ;'A' REPLY EXPECTED
1935 012342 111215      MOVB    (R2),(R5)     ;READ IRR2
1936 012344 023715 001124      CMP     $GDDAT,(R5)
1937 012350 001401      BEQ     3$
1938 012352 104003      ERROR   3             ;IRR GROUP 2
1939 012354 005723      3$: TST   (R3)+         ;INDEX DATA TABLE
1940 012356 020327 013410      CMP     R3,#ENDDAT    ;CHECK FOR PATTERN END
1941 012362 001325      JNE     1$           ;DO NEXT PATTERN

```

```

1942
1943
1944
1945
1946
1947 012364 000004      ;:*****
1948 012366 012737 000001 001160      ;:*TEST 50      TEST CSRA AND CSRB WITH RESET
1949 012374 004737 013002      ;:*****
1950 012400 013700 001342      TST50: SCOPE
1951 012404 013701 001346      MOV      #1,$TIMES    ;:DO 1 ITERATION
1952 012410 112760 001001 000001      JSR      PC,CLRCR     ;:CLEAR ALL CSRS
1953 012416 112761 000001 000001      MOV      DRCSA,R0
1954 012424 010037 001122      MOV      DRCSB,R1
1955 012430 012737 100300 001124      MOVB    #IE!BIT0,1(R0) ;:CSRA TO OUTPUT MODE
1956 012436 000005      MOVB    #BIT0,1(R1)   ;:SET CSRB TO OUTPUT MODE
1957 012440 011015      MOV      R0,$BDADR    ;:STORE CSRA ADDRESS
1958 012442 042715 000007      MOV      #100300,$GDDAT ;:RDY + CHIP STATUS EXP'D
1959 012446 023715 001124      RESET
1960 012452 001401      MOV      (R0),(R5)    ;:INITIALIZE
1961 012454 104002      BIC     #7,(R5)       ;:STORE REC'D
1962 012456 012737 100000 001124 1$: BIC     #7,(R5)       ;:CLEAR UNDEFINED BITS
1963 012464 010137 001122      CMP     $GDDAT,(R5)  ;:MAKE COMPARE
1964 012470 011115      BEQ     1$
1965 012472 023715 001124      ERROR   2             ;:CSRA ERROR ON RESET
1966 012476 001401      MOV      #100000,$GDDAT ;:STORE EXPECTED
1967      MOV      R1,$BDADR   ;:CHECK CSRB
1968      MOV      (R1),(R5) ;:STORE IN $BDADR
1969      CMP     $GDDAT,(R5) ;:MAKE COMPARE
1970      BEQ     TST51     ;:BR IF EQUAL

```

1967 012500 104002 ERROR 2 ;CSRB ERROR WITH RESET

1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995

012502 000004
012504 012737 000001 001160
012512 004737 013002
012516 013702 001352
012522 013703 001356
012526 112762 000001 000001
012534 112763 000001 000001
012542 010237 001122
012546 012737 100200 001124
012554 000005
012556 011215
012560 042715 000007
012564 023715 001124
012570 001401
012572 104002
012574 010337 001122 1\$:
012600 012737 100000 001124
012606 011315
012610 023715 001124
012614 001401
012616 104002
012620 2\$:

```
*****  
*TEST 51 TEST CSRC AND CSRD WITH RESET  
*****  
TST51: SCOPE  
MOV #1,$TIMES ;DO 1 ITERATION  
JSR PC,CLRCR  
MOV DRCSC,R2  
MOV DRCSD,R3  
MOVB #BIT0,1(R2) ;SET CSRC TO OUTPUT MODF  
MOVB #BIT0,1(R3) ;SET CSRD TO OUTPUT MODE  
MOV R2,$BDADR ;STORE ADDRESS  
MOV #100200,$GDDAT ;RDY + CHIP STATUS  
RESET ;INITIALIZE  
MOV (R2),(R5) ;STORE CSRC  
BIC #7,(R5) ;CLEAR UNDEFINED BITS  
CMP $GDDAT,(R5) ;MAKE COMPARE  
BEQ 1$  
ERROR 2 ;CSRC ERROR WITH RESET  
MOV R3,$BDADR ;CHECK CSRD  
MOV #100000,$GDDAT ;STORE EXPECTED  
MOV (R3),(R5) ;READ CSRD  
CMP $GDDAT,(R5) ;MAKE COMPARE  
BEQ 2$  
ERROR 2 ;CSRD ERROR ON RESULT
```

```

1996
1997
1998
1999
2000 012620 013701 001342 ;DON'T REPORT 'EOP' UNTIL ALL SELECTED DRV11-J'S HAVE BEEN TESTED.
NXDEV: MOV DR(CSA,R1 ;INIT TO SETUP NEXT DRV11J ADDRESSES
2001 012624 000241 NXDEV1: CLC ;CLEAR CARRY FOR DEVICE MAP
2002 012626 006037 001364 ROR DMAP ;LOOK FOR NEXT DRV11J
2003 012632 001412 BEQ $EOP ;BR IF ALL TESTED
2004 012634 162701 000020 SUB #20,R1 ;NEXT DRV11-J STARTS -20
2005 012640 005237 001202 INC $UNIT ;UPDATE UNIT NUMBER
2006 012644 032737 000001 001364 BIT #1,DMAP ;IS UNIT SELECTED?
2007 012652 001764 BEQ NXDEV1 ;NEXT,IF NOT SELECTED
2008 012654 000137 002032 JMP NEXPAS ;TEST NEXT DRV11-J
2009
2010 .SBTTL END OF PASS ROUTINE
2011
2012 ;*****
2013 ;*INCREMENT THE PASS NUMBER ($PASS)
2014 ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
2015 ;*IF THERES A MONITOR GO TO IT
2016 ;*IF THERE ISN'T JUMP TO START1
2017
2018 012660 $EOP:
2019 012662 000240 NOP
2020 012666 005037 001102 CLR $STNM ;:ZERO THE TEST NUMBER
2021 012672 005037 001160 CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
2022 012676 042737 100000 001176 INC $PASS ;:INCREMENT THE PASS NUMBER
2023 012704 005327 BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
2024 012706 000001 DEC (PC)+ ;:LOOP?
2025 012710 003022 $EOPCT: .WORD 1
2026 012712 012737 BGT $DOAGN ;:YES
2027 012714 000001 MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
2028 012716 012706 $ENDCT: .WORD 1
2029 012720 104401 012765 TYPE $ENDMG ;:TYPE 'END PASS #'
2030 012724 013746 001176 MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
2031 012730 104405 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
2032 012732 104401 012762 TYPE $ENULL ;:TYPE A NULL CHARACTER
2033 012736 013700 000042 $GET42: MOV @#42,R0 ;:GET MONITOR ADDRESS
2034 012742 001405 BEQ $DOAGN ;:BRANCH IF NO MONITOR
2035 012744 000005 RESET ;:CLEAR THE WORLD
2036 012746 004710 $ENDAD: JSR PC,(R0) ;:GO TO MONITOR
2037 012750 000240 NOP ;:SAVE ROOM
2038 012752 000240 NOP ;:FOR
2039 012754 000240 NOP ;:ACT11
2040
2041 012756 000137 $DOAGN:
2042 012760 001766 JMP @(PC)+ ;:RETURN
2043 012762 377 377 000 $RTNAD: .WORD START1
2044 012765 015 042412 042116 $ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
2045 012772 050040 051501 020123 $ENDMG: .ASCIZ <15><12>/END PASS #/
2046 013000 000043
    
```

2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079

013002 012705 001126
013006 013700 001342
013012 013701 001346
013016 013702 001352
013022 013703 001356
013026 005037 001124
013032 005015
013034 005010
013036 105061 000001
013042 005012
013044 105063 000001
013050 000207

013052 010046
013054 013700 001342
013060 105060 000011
013064 112760 000001 000001
013072 005060 000002
013076 105010
013100 105060 000010
013104 012600
013106 000207

```
.SBTTL PROGRAM SUBROUTINES

:*****
: CLEAR ALL CONTROL/STATUS REGISTERS
: INIT REGISTERS R0-R4 WITH CSRA-CSR4
: AND STORE $BDDAT INTO R5.
:*****
CLRCSR: MOV    # $BDDAT,R5      ;BAD DATA STORAGE IN R5
        MOV    DRCSA,R0       ;CSRA ADDRESS TO R0
        MOV    DRCSB,R1       ;CSRB ADDRESS TO R1
        MOV    DRCSA,R2       ;CSRC ADDRESS TO R2
        MOV    DRCSA,R3       ;CSR4 ADDRESS TO R3
        CLR    $GDDAT         ;CLEAR EXPECTED
        CLR    (R5)           ;CLEAR REC'D
        CLR    (R0)           ;CLEAR CSRA;CHIP RESET GROUP 1
        CLR    1(R1)          ;CLEAR HIGH BYTE CSRB
        CLR    (R2)           ;CLEAR CSRC;CHIP RESET GROUP 2
        CLR    1(R3)          ;CLEAR HIGH BYTE CSR4
        RTS    PC

; CLEAR IRR REGISTERS, GROUP 1, GROUP 2 WITH CHIP RESET
CLRIRR: MOV    R0,-(SF)
        MOV    DRCSA,R0       ;START OF CSR ADDRESS
        CLR    11(R0)         ;CSRC TO INPUT MODE
        MOV    #BIT0,1(R0)    ;CSRA TO OUTPUT MODE
        CLR    2(R0)         ;CLEAR DBRA
        CLR    (R0)           ;CHIP RESET OF GROUP1
        CLR    10(R0)         ;CHIP RESET OF GROUP 2
        MOV    (SP)+,R0       ;RESTORE REGISTER
        RTS    PC            ;EXIT
```

```
2080
2081      .SBTTL PATTERNS FOR REGISTER R/W
2082      ;
2083      ;PATTERNS USED FOR LOADING/READING REGISTERS
2084
2085      013110 000000      BEGPAT: 0          ;GROWING 1
2086      013112 000001      BGPAT1: 1
2087      013114 000003      3
2088      013116 000007      7
2089      013120 000017      17
2090      013122 000037      37
2091      013124 000077      77
2092      013126 000177      177
2093      013130 000377      377
2094      013132 000777      EDCHP1: 777
2095      013134 001777      1777
2096      013136 003777      3777
2097      013140 007777      7777
2098      013142 017777      17777
2099      013144 037777      37777
2100      013146 077777      77777
2101      013150 177777      177777
2102      013152 177776      BGCHP2: 177776      ;GROWING 0
2103      013154 177774      177774
2104      013156 177770      177770
2105      013160 177760      177760
2106      013162 177740      177740
2107      013164 177700      177700
2108      013166 177600      177600
2109      013170 177400      EDCHP2: 177400
2110      013172 177000      177000
2111      013174 176000      176000
2112      013176 174000      174000
2113      013200 170000      170000
2114      013202 160000      160000
2115      013204 140000      140000
2116      013206 100000      100000
2117
2118      013210 000000      000000
2119      013212 000001      BGCHP3: 1          ;WALKING 1
2120      013214 000002      2
2121      013216 000004      4
2122      013220 000010      10
2123      013222 000020      20
2124      013224 000040      40
2125      013226 000100      100
2126      013230 000200      200
2127      013232 000400      EDCHP3: 400
2128      013234 001000      1000
2129      013236 002000      2000
2130      013240 004000      4000
2131      013242 010000      10000
2132      013244 020000      20000
2133      013246 040000      40000
2134      013250 100000      100000
2135      013252 177777      177777      ;WALKING 0
```

2136	013254	177776	BGCHP4:	177776
2137	013256	177775		177775
2138	013260	177773		177773
2139	013262	177767		177767
2140	013264	177757		177757
2141	013266	177737		177737
2142	013270	177677		177677
2143	013272	177577		177577
2144	013274	177377	EDCHP4:	177377
2145	013276	176777		176777
2146	013300	175777		175777
2147	013302	173777		173777
2148	013304	167777		167777
2149	013306	157777		157777
2150	013310	137777		137777
2151	013312	077777		077777
2152	013314	177777		177777
2153	013316	000000	ENDPAT:	000000
2154				
2155			;DATA PATTERNS	
2156	013320	155555	PATDAT:	155555
2157	013322	133333		133333
2158	013324	066666		066666
2159	013326	125252		125252
2160	013330	052525		052525
2161	013332	177777		177777
2162	013334	000000		000000
2163	013336	107070		107070
2164	013340	070707		070707
2165	013342	144444		144444
2166	013344	033333		033333
2167	013346	011111		011111
2168	013350	022222		022222
2169	013352	044444		044444
2170	013354	111111		111111
2171	013356	166666		166666
2172	013360	010421		010421
2173	013362	021042		021042
2174	013364	031463		031463
2175	013366	042104		042104
2176	013370	063146		063146
2177	013372	073567		073567
2178	013374	104210		104210
2179	013376	114631		114631
2180	013400	135673		135673
2181	013402	146314		146314
2182	013404	156735		156735
2183	013406	167356		167356
2184	013410	000000	ENDDAT:	000000
2185				

```
2186 .SBTTL SYSMAC ROUTINES
2187
2188 .SBTTL TYPE ROUTINE
2189
2190 *****
2191 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2192 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2193 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2194 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2195 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2196
2197 *CALL:
2198 *1) USING A TRAP INSTRUCTION
2199 * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2200
2201 *OR
2202 * TYPE
2203 * MESADR
2204
2205 013412 105737 001157 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
2206 013416 100002 BPL 1$ ;; BR IF YES
2207 013420 000000 HALT ;; HALT HERE IF NO TERMINAL
2208 013422 000430 BR 3$ ;; LEAVE
2209 013424 010046 1$: MOV R0,-(SP) ;; SAVE R0
2210 013426 017600 000002 MOV @2(SP),R0 ;; GET ADDRESS OF ASCIZ STRING
2211 013432 122737 000001 001210 CMPB #APTENV,$ENV ;; RUNNING IN APT MODE
2212 013440 001011 BNE 62$ ;; NO,GO CHECK FOR APT CONSOLE
2213 013442 132737 000100 001211 BITB #APTPOOL,$ENVM ;; SPOOL MESSAGE TO APT
2214 013450 001405 BEQ 62$ ;; NO,GO CHECK FOR CONSOLE
2215 013452 010037 013462 MOV R0,61$ ;; SETUP MESSAGE ADDRESS FOR APT
2216 013456 004737 013702 JSR PC,$ATY3 ;; SPOOL MESSAGE TO APT
2217 013462 000000 61$: .WORD 0 ;; MESSAGE ADDRESS
2218 013464 132737 000040 001211 62$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
2219 013472 001003 BNE 60$ ;; YES,SKIP TYPE OUT
2220 013474 112046 2$: MOVB (R0)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2221 013476 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
2222 013500 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
2223 013502 012600 60$: MOV (SP)+,R0 ;; RESTORE R0
2224 013504 062716 000002 3$: ADD #2,(SP) ;; ADJUST RETURN PC
2225 013510 000002 RTI ;; RETURN
2226 013512 122716 000011 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
2227 013516 001430 BEQ 8$
2228 013520 122716 000200 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
2229 013524 001006 BNE 5$
2230 013526 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
2231 013530 104401 TYPE ;; TYPE A CR AND LF
2232 013532 001165 $CRLF
2233 013534 105037 013670 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
2234 013540 000755 BR 2$ ;; GET NEXT CHARACTER
2235 013542 004737 013624 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
2236 013546 123726 001156 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
2237 013552 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
2238 013554 013746 001154 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
2239 AND THE NULL CHAR.
2240 013560 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
2241 013564 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
```

```

2242 013566 004737 013624      JSR    PC,$TYPEC      ::GO TYPE A NULI
2243 013572 105337 013670      DECB   $CHARCNT      ::DO NOT COUNT AS A COUNT
2244 013576 000770              BR     7$            ::LOOP
2245
2246      ;HORIZONTAL TAB PROCESSOR
2247
2248 013600 112716 000040      8$:    MOVB   #' ,(SP)      ::REPLACE TAB WITH SPACF
2249 013604 004737 013624      9$:    JSR    PC,$TYPEC      ::TYPE A SPACE
2250 013610 132737 000007 013670  BITB   #',$CHARCNT      ::BRANCH IF NOT AT
2251 013616 001372              BNE    9$            ::TAB STOP
2252 013620 005726              TST    (SP)+          ::POP SPACE OFF STACK
2253 013622 000724              BR     2$            ::GET NEXT CHARACTER
2254 013624 105777 165320      $TYPEC: TSTB   @STPS      ::WAIT UNTIL PRINTER IS READY
2255 013630 100375              BPL    $TYPEC
2256 013632 116677 000002 165312  MOVB   2(SP),@STPB     ::LOAD CHAR TO BE TYPED INTO DATA REG.
2257 013640 122766 000015 000002  CMPB   #CR,2(SP)      ::IS CHARACTER A CARRIAGE RETURN?
2258 013646 001003              BNE    1$            ::BRANCH IF NO
2259 013650 105037 013670      CLRB   $CHARCNT      ::YES--CLEAR CHARACTER COUNT
2260 013651 000406              BR     $TYPEX        ::EXIT
2261 013656 122766 000012 000002  1$:    CMPB   #LF,2(SF)   ::IS CHARACTER A LINE FEED?
2262 013664 001402              BEQ    $TYPEX        ::BRANCH IF YES
2263 013666 105227              INCB   (PC)+          ::COUNT THE CHARACTER
2264 013670 000000      $CHARCNT: WORD 0      ::CHARACTER COUNT STORAGE
2265 013672 000207      $TYPEX: RTS    PC
2266
2267      .SBTTL  APT COMMUNICATIONS ROUTINE
2268
2269      ;*****
2270 013674 112737 000001 014140  $ATY1: MOVB   #',$FFLG      ::TO REPORT FATAL ERROR
2271 013702 112737 000001 014136  $ATY3: MOVB   #',$MFLG      ::TO TYPE A MESSAGE
2272 013710 000403              BR     $ATYC
2273 013712 112737 000001 014140  $ATY4: MOVB   #',$FFLG      ::TO ONLY REPORT FATAL ERROR
2274 013720      $ATYC:
2275 013720 010046      MOV    R0,-(SP)      ::PUSH R0 ON STACK
2276 013722 010146      MOV    R1,-(SP)      ::PUSH R1 ON STACK
2277 013724 105737 014136      TSTB   $MFLG        ::SHOULD TYPE A MESSAGE?
2278 013730 001450      BEQ    5$            ::IF NOT: BR
2279 013732 122737 000001 001210  CMPB   #APTENV,$ENV   ::OPERATING UNDER APT?
2280 013740 001031              BNE    3$            ::IF NOT: BR
2281 013742 132737 000100 001 11  BITB   #APTPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
2282 013750 001425      BEQ    3$            ::IF NOT: BR
2283 013752 017600 000004      MOV    @4(SP),R0     ::GET MESSAGE ADDR.
2284 013756 062766 000002 000004  ADD    #2,4(SP)      ::BUMP RETURN ADDR.
2285 013764 005737 001170      1$:    TST    $MSGTYPE     ::SEE IF DONE W/ LAST XMISSION?
2286 013770 001375              BNE    1$            ::IF NOT: WAIT
2287 013772 010037 001204      MOV    R0,$MSGAD     ::PUT ADDR IN MAILBOX
2288 013776 105720      2$:    TSTB   (R0+        ::FIND END OF MESSAGE
2289 014000 001376              BNE    2$
2290 014002 163700 001204      SUB    $MSGAD,R0     ::SUB START OF MESSAGE
2291 014006 006200      ASR    R0            ::GET MESSAGE LGTH IN WORDS
2292 014010 010037 001206      MOV    R0,$MSGGLT    ::PUT LENGTH IN MAILBOX
2293 014014 012737 000004 001170  MOV    #4,$MSGTYPE   ::TELL APT TO TAKE MSG.
2294 014022 000413              BR     5$
2295 014024 017637 000004 014050  3$:    MOV    @4(SP),4$     ::PUT MSG ADDR IN JSR LINKAGE
2296 014032 062766 000002 000004  ADD    #2,4(SP)      ::BUMP RETURN ADDRESS
2297 014040 013746 177776      MOV    177776,-(SP)  ::PUSH 177776 ON STACK

```



```
2298 014044 004737 013412          JSR    PC,$TYPE      ;;CALL TYPE MACRO
2299 014050 000000          4$:    .WORD    0
2300 014052 000000          5$:
2301 014052 105737 014140          10$:   TSTB    $FFLG      ;;SHOULD REPORT FATAL ERROR?
2302 014056 001416          BEQ    12$           ;;IF NOT: BR
2303 014060 005737 001210          TST    $ENV         ;;RUNNING UNDER APT?
2304 014064 001413          BEQ    11$           ;;IF NOT: BR
2305 014066 005737 001170          11$:   TST    $MSGTYPE   ;;FINISHED LAST MESSAGE?
2306 014072 001375          BNE    11$           ;;IF NOT: WAIT
2307 014074 017637 000004 001172  MOV    @4(SP),$FATAL ;;GET ERROR #
2308 014102 062766 000002 000004  ADD    #2,4(SP)      ;;BUMP RETURN ADDR.
2309 014110 005237 001170          INC    $MSGTYPE     ;;TFLL APT TO TAKE ERROR
2310 014114 105037 014140          12$:   CLRB    $FFLG      ;;CLEAR FATAL FLAG
2311 014120 105037 014137          CLRB    $LFLG       ;;CLEAR LOG FLAG
2312 014124 105037 014136          CLRB    $MFLG       ;;CLEAR MESSAGE FLAG
2313 014130 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
2314 014132 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
2315 014134 000207          RTS    PC           ;;RETURN
2316 014136 000          $MFLG: .BYTE    0   ;;MESSG. FLAG
2317 014137 000          $LFLG: .BYTE    0   ;;LOG FLAG
2318 014140 000          $FFLG: .BYTE    0   ;;FATAL FLAG
2319          014142          .EVEN
2320          000200          APT SIZE=200
2321          000001          APTENV=001
2322          000100          APTSPOOL=100
2323          000040          APTCSUP=040
2324          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
2325
2326          ;;*****
2327          ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2328          ;;*OCTAL (ASCII) NUMBER AND TYPE IT.
2329          ;;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2330          ;;*CALL:
2331          ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
2332          ;;*   TYPOS      ;;CALL FOR TYPEOUT
2333          ;;*   .BYTE    N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2334          ;;*   .BYTE    M          ;;M=1 OR 0
2335          ;;*                               ;;1=TYPE LEADING ZEROS
2336          ;;*                               ;;0=SUPPRESS LEADING ZEROS
2337
2338          ;;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2339          ;;*$TYPOS OR $TYPOC
2340          ;;*CALL:
2341          ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
2342          ;;*   TYPON      ;;CALL FOR TYPEOUT
2343
2344          ;;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2345          ;;*CALL:
2346          ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
2347          ;;*   TYPOC      ;;CALL FOR TYPEOUT
2348
2349 014142 017646 000000 014365  $TYPOS: MOV    @2(SP),-(SP)   ;;PICKUP THE MODE
2350 014146 116637 000001          MOVB   1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
2351 014154 112637 014367          MOVB   (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
2352 014160 062716 000002          ADD    #2,(SP)      ;;ADJUST RETURN ADDRESS
2353 014164 000406          BR     $TYPON
```

```

2354 014166 112737 000001 014365 $TYPOC: MOVB #1,$OFILL      ;;SET THE ZERO FILL SWITCH
2355 014174 112737 000006 014367      MOVB #6,$SOMODE+1  ;;SET FOR SIX(6) DIGITS
2356 014202 112737 000005 014364 $TYPON: MOVB #5,$OCNT  ;;SET THE ITERATION COUNT
2357 014210 010346      MOV R3,-(SP)      ;;SAVE R3
2358 014212 010446      MOV R4,-(SP)      ;;SAVE R4
2359 014214 010546      MOV R5,-(SP)      ;;SAVE R5
2360 014216 113704 014367      MOVB $SOMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
2361 014222 005404      NEG R4
2362 014224 062704 000006      ADD #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
2363 014230 110437 014366      MOVB R4,$SOMODE   ;;SAVE IT FOR USE
2364 014234 113704 014365      MOVB $OFILL,R4    ;;GET THE ZERO FILL SWITCH
2365 014240 016605 000012      MOV 12(SP),R5    ;;PICKUP THE INPUT NUMBER
2366 014244 005003      CLR R3           ;;CLEAR THE OUTPUT WORD
2367 014246 006105      1$: ROL R5      ;;ROTATE MSB INTO 'C'
2368 014250 000404      BR 3$           ;;GO DO MSB
2369 014252 006105      2$: ROL R5      ;;FORM THIS DIGIT
2370 014254 006105      ROL R5
2371 014256 006105      ROL R5
2372 014260 010503      MOV * R5,R3
2373 014262 006103      3$: ROL R3      ;;GET LSB OF THIS DIGIT
2374 014264 105337 014366      DECB $SOMODE     ;;TYPE THIS DIGIT?
2375 014270 100016      BPL 7$         ;;BR IF NO
2376 014272 042703 177770      BIC #177770,R3  ;;GET RID OF JUNK
2377 014276 001002      BNE 4$         ;;TEST FOR 0
2378 014300 005704      TST R4         ;;SUPPRESS THIS 0?
2379 014302 001403      BEQ 5$         ;;BR IF YES
2380 014304 005204      4$: INC R4     ;;DON'T SUPPRESS AN YMORE 0'S
2381 014306 052703 000060      BIS #'0,R3     ;;MAKE THIS DIGIT ASCII
2382 014312 052703 000040      5$: BIS #' ,R3  ;;MAKE ASCII IF NOT ALREADY
2383 014316 110337 014362      MOVB R3,R8     ;;SAVE FOR TYPING
2384 014322 104401 014362      TYPE R8       ;;GO TYPE THIS DIGIT
2385 014326 105337 014364      7$: DECB $OCNT  ;;COUNT BY 1
2386 014332 003347      BGT 2$         ;;BR IF MORE TO DO
2387 014334 002402      BLT 6$         ;;BR IF DONE
2388 014336 005204      INC R4        ;;INSURE LAST DIGIT ISN'T A BLANK
2389 014340 000744      BR 2$         ;;GO DO THE LAST DIGIT
2390 014342 012605      6$: MOV (SP)+,R5  ;;RESTORE R5
2391 014344 012604      MOV (SP)+,R4  ;;RESTORE R4
2392 014346 012603      MOV (SP)+,R3  ;;RESTORE R3
2393 014350 016666 000002 000004      MOV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING.
2394 014356 012616      MOV (SP)+,(SP)
2395 014360 000002      RTI          ;;RETURN
2396 014362 000      8$: .BYTE 0    ;;STORAGE FOR ASCII DIGIT
2397 014363 000      .BYTE 0     ;;TERMINATOR FOR TYPE ROUTINE
2398 014364 000      $OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
2399 014365 000      $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
2400 014366 000000      $SOMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.
*CALL

```

2401
2402
2403
2404
2405
2406
2407
2408
2409

```
2410      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
2411      TYPDS      ;;GO TO THE ROUTINE
2412
2413      014370      $TYPDS:
2414      014370      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
2415      014372      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
2416      014374      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
2417      014376      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
2418      014400      010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
2419      014402      012746      020200      MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
2420      014406      016605      000020      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
2421      014412      100004      BPL      1$      ;;BR IF INPUT IS POS.
2422      014414      005405      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
2423      014416      112766      000055      000001      MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
2424      014424      005000      1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
2425      014426      012703      014604      MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
2426      014432      112723      000040      MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2427      014436      005002      2$:      CLR      R2      ;;CLEAR THE BCD NUMBER
2428      014440      016001      014574      MOV      $DTBL(R0),R1      ;;GET THE CONSTANT
2429      014444      160105      3$:      SUB      R1,R5      ;;FORM THIS BCD DIGIT
2430      014446      002402      BLT      4$      ;;BR IF DONE
2431      014450      005202      INC      R2      ;;INCREASE THE BCD DIGIT BY 1
2432      014452      000774      BR      3$
2433      014454      060105      4$:      ADD      R1,R5      ;;ADD BACK THE CONSTANT
2434      014456      005702      TST      R2      ;;CHECK IF BCD DIGIT=0
2435      014460      001002      BNE      5$      ;;FALL THROUGH IF 0
2436      014462      105716      TSTB     (R2)      ;;STILL DOING LEADING 0'S?
2437      014464      100407      BMI      7$      ;;BR IF YES
2438      014466      106316      5$:      ASLB     (SP)      ;;MSD?
2439      014470      103003      BCF      6$      ;;BR IF NO
2440      014472      116663      000001      177777      MOVB     1(SP),-1(R3)      ;;YES--SET THE SIGN
2441      014500      052702      000060      6$:      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
2442      014504      052702      000040      7$:      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2443      014510      110223      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2444      014512      005720      TST      (R0)+      ;;JUST INCREMENTING
2445      014514      020027      000010      CMP      R0,#10      ;;CHECK THE TABLE INDEX
2446      014520      002746      BLT      2$      ;;GO DO THE NEXT DIGIT
2447      014522      003002      BGT      8$      ;;GO TO EXIT
2448      014524      010502      MOV      R5,R2      ;;GET THE LSD
2449      014526      000764      BR      6$      ;;GO CHANGE TO ASCII
2450      014530      105726      8$:      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
2451      014532      100003      BPL      9$      ;;BR IF NO
2452      014534      116663      177777      177776      MOVB     -1(SP),-2(R3)      ;;YES--SET THE SIGN FOR TYPING
2453      014542      105013      9$:      CLRB     (R3)      ;;SET THE TERMINATOR
2454      014544      012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
2455      014546      012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
2456      014550      012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
2457      014552      012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
2458      014554      012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
2459      014556      104401      014604      TYPE     $SDBLK      ;;NOW TYPE THE NUMBER
2460      014562      016666      000002      000004      MOV      2(SP),4(SP)      ;;ADJUST THE STACK
2461      014570      012616      MOV      (SP)+,(SP)
2462      014572      000002      RTM
2463      014574      023420      SETBL   10000
2464      014576      001750      1000
2465      014600      000144      100
```

2466 014602 000012
2467 014604 000004
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481 014614
2482 014614 104407
2483 014616 104407
2484 014620 105237 001103
2485 014624 001725
2486 014626 013777 001102 164306
2487 014634 005237 001112
2488 014640 011637 001116
2489 014644 162737 000002 001116
2490 014652 117737 164240 001114
2491 014660 032777 020000 164252
2492 014666 001004
2493 014670 004737 014770
2494 014674 104401 001165
2495 014700
2496 014700 122737 000001 001210
2497 014706 001007
2498 014710 113737 001114 014722
2499 014716 004737 013712
2500 014722 000
2501 014723 000
2502 014724 000777
2503 014726 005777 164206
2504 014732 100002
2505 014734 000000
2506 014736 104407
2507 014740 032777 001000 164172
2508 014746 001402
2509 014750 013716 001110
2510 014754 005737 001162
2511 014760 001402
2512 014762 013716 001162
2513 014766
2514 014766 000002
2515
2516
2517
2518
2519 014770 113737 001102 001362
2520 014776 004737 015006
2521 015002 104407

\$DBLK: BLKW 4
\$BTTL 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:
*SW5=1 HALT ON ERROR
*SW13=1 INHIBIT ERROR TIMEOUTS
*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
7S: INCB \$ERFLG ;:SET THE ERROR FLAG
BEQ 7S ;: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 NUMBER
BIT #BIT13,@SWR ;:SKIP TIMEOUT IF SET
BNE 20S ;:SKIP TIMEOUTS
JSR PC,SWRCK ;:GO TO USER ERROR ROUTINE
TYPE \$CRLF
20S: CMPS #APTENV,\$ENV ;:RUNNING IN APT MODE
BNE 2S ;:NO SKIP APT ERROR REPORT
MOVB \$ITEMB,21S ;:SET ITEM NUMBER AS ERROR NUMBER
JSR PC,\$ATY4 ;:REPORT FATAL ERROR TO APT
2S: BYTE 0
BYTE 0
22S: BR 22S ;:APT ERROR LOOP
SI @SWR ;:HALT ON ERROR
BPL 3S ;:SKIP IF CONTINUE
HALT ;:HALT ON ERROR
CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
3S: BIT #BIT09,@SWR ;:LOOP ON ERROR SWITCH SET?
BEQ 4S ;:BR IF NO
MOV \$LPERR,(SP) ;:FUDGE RETURN FOR LOOPING
4S: TST \$ESCAPE ;:CHECK FOR AN ESCAPE ADDRESS
BEQ 5S ;:BR IF NONE
MOV \$ESCAPE,(SP) ;:FUDGE RETURN ADDRESS FOR ESCAPE
5S: RTI ;:RETURN

*GO TYPE ERROR
*GO UPDATE SOFTWARE SWR-IF 'CNTRL/G'

SWRCK: MOVB \$TSTNM,\$TSTNUM ;:SET UP TEST NUMBER
JSR PC,\$ERRTYP ;:GO TYPE ERROR
CKSWR ;:GO LOOK FOR SWR CHANGE

2522 015004 000207
2523
2524
2525
2526
2527
2528
2529
2530 015006
2531 015006 104401 001165
2532 015012 010046
2533 015014 005000
2534 015016 153700 001114
2535 015022 001004
2536
2537 015024 013746 001116
2538
2539 015030 104402
2540 015032 000426
2541 015034 005300
2542 015036 006300
2543 015040 006300
2544 015042 006300
2545 015044 062700 001252
2546 015050 012037 015060
2547 015054 001404
2548 015056 104401
2549 015060 000000
2550 015062 104401 001165
2551 015066 012037 015076
2552 015072 001404
2553 015074 104401
2554 015076 000000
2555 015100 104401 001165
2556 015104 011000
2557 015106 001004
2558 015110 012600
2559 015112 104401 001165
2560 015116 000207
2561 015120
2562 015120 013046
2563 015122 104402
2564 015124 005710
2565 015126 001770
2566 015130 104401 015136
2567 015134 000771
2568 015136 020040 000
2569 015142
2570
2571
2572
2573
2574
2575
2576
2577

```
RTS PC ;RETURN TO ERROR HANDLER
.SBTTL ERROR MESSAGE TIMEOUT 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 $CRLF ;'CARRIAGE RETURN' & 'LINE FEED'
MOV R0,-(SP) ;SAVE R0
CLR R0 ;PICKUP THE ITEM INDEX
BISB @($ITEMB),R0
BNF 1$ ;IF ITEM NUMBER IS ZERO, JUST
;TYPE THE PC OF THE ERROR
MOV $ERRPC,-(SP) ;SAVE $ERRPC FOR TIMEOUT
;ERROR ADDRESS
TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
BR 6$ ;GET OUT
1$: DEC R0 ;ADJUST THE INDEX SO THAT IT WILL
; WORK FOR THE ERROR TABLE
ASL R0
ASL R0
ASL R0
ADD #($ERRTB),R0 ;FORM TABLE POINTER
MOV (R0)+,2$ ;PICKUP 'ERROR MESSAGE' POINTER
BEQ 3$ ;SKIP TIMEOUT IF NO POINTER
TYPE ;TYPE THE 'ERROR MESSAGE'
2$: .WORD 0 ;'ERROR MESSAGE' POINTER GOES HERE
TYPE $CRLF ;'CARRIAGE RETURN' & 'LINE FEED'
3$: MOV (R0)+,4$ ;PICKUP 'DATA HEADER' POINTER
BEQ 5$ ;SKIP TIMEOUT IF 0
TYPE ;TYPE THE 'DATA HEADER'
4$: .WORD 0 ;'DATA HEADER' POINTER GOES HERE
TYPE $CRLF ;'CARRIAGE RETURN' & 'LINE FEED'
5$: MOV (R0),R0 ;PICKUP 'DATA TABLE' POINTER
BNE 7$ ;GO TYPE THE DATA
MOV (SP)+,R0 ;RESTORE R0
TYPE $CRLF ;'CARRIAGE RETURN' & 'LINE FEED'
6$: .PC ;RETURN
7$: MOV @ (R0)+,-(SP) ;SAVE @ (R0)+ FOR TIMEOUT
TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
TST (R0) ;IS THERE ANOTHER NUMBER?
BEQ 6$ ;BR IF NO
TYPE 8$ ;TYPE TWO(2) SPACES
BR 7$ ;LOOP
8$: .ASCIIZ / / ;TWO(2) SPACES
.EVEN

.SBTTL SCOPE HANDLER ROUTINE

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

J 5

```
2578      : *SW11=1      INHIBIT ITERATIONS
2579      : *SW09=1      LOOP ON ERROR
2580      : *SW08=1      LOOP ON TEST IN SWR<7:0>
2581      : *CALL
2582      : *          SCOPE          .:SCOPE=IOT
2583
2584      015142      $SCOPE:
2585      015142      104407      CKSWR          ::TEST FOR CHANGE IN SOFT-SWR
2586      015144      032777      040000      1$: BIT      #BIT14,@SWR      ::LOOP ON PRESENT TEST?
2587      015152      001114      BNE      $OVER          ::YES IF SW14=1
2588      :*****START OF CODE FOR THE XOR TESTER*****
2589      015154      000416      $XTSTR: BR      6$          ::IF RUNNING ON THE 'XOR' TESTER CHANGE
2590      :THIS INSTRUCTION TO A 'NOP' (NOP=240)
2591      015156      013746      000004      MOV      @ERRVEC,-(SP)      ::SAVE THE CONTENTS OF THE ERROR VECTOR
2592      015162      012737      015202      000004      MOV      #5$,@ERRVEC      ::SET FOR TIMEOUT
2593      015170      005737      177060      TST      @177060          ::TIME OUT ON XOR?
2594      015174      012637      000004      MOV      (SP)+,@ERRVEC      ::RESTORE THE ERROR VECTOR
2595      015200      000463      BR      $$VLAD          ::GO TO THE NEXT TEST
2596      015202      022626      5$: CMP      (SP)+,(SP)+      ::CLEAR THE STACK AFTER A TIME OUT
2597      015204      012637      000004      MOV      (SP)+,@ERRVEC      ::RESTORE THE ERROR VECTOR
2598      015210      000423      BR      7$          ::LOOP ON THE PRESENT TEST
2599      015212      6$:*****END OF CODE FOR THE XOR TESTER*****
2600      015212      032777      000400      163720      BIT      #BIT08,@SWR      ::LOOP ON SPEC. TEST?
2601      015220      001404      BEQ      2$          ::BR IF NO
2602      015222      127737      163712      001102      CMPB     @SWR,$STNM      ::ON THE RIGHT TEST? SWR<7:0>
2603      015230      001465      BEQ      $OVER          ::BR IF YES
2604      015232      105737      001103      2$: TSTB     $ERFLG      ::HAS AN ERROR OCCURRED?
2605      015236      001421      BEQ      3$          ::BR IF NO
2606      015240      123737      001115      001103      CMPB     $ERMAX,$ERFLG      ::MAX. ERRORS FOR THIS TEST OCCURRED?
2607      015246      101015      BHI      3$          ::BR IF NO
2608      015250      032777      001000      163662      BIT      #BIT07,@SWR      ::LOOP ON ERROR?
2609      015256      001404      BEQ      4$          ::BR IF NO
2610      015260      013737      001110      001106      7$: MOV      $LPERR,$LPADR      ::SET LOOP ADDRESS TO LAST SCOPE
2611      015266      000446      BR      $OVER
2612      015270      105037      001103      4$: CLRB     $ERFLG          ::ZERO THE ERROR FLAG
2613      015274      005037      001160      CLR      $TIMES          ::CLEAR THE NUMBER OF ITERATIONS TO MAKE
2614      015300      000415      BR      1$          ::ESCAPE TO THE NEXT TEST
2615      015302      032777      004000      163630      3$: BIT      #BIT11,@SWR      ::INHIBIT ITERATIONS?
2616      015310      001011      BNE      1$          ::BR IF YES
2617      015312      005737      001176      TST      $PASS          ::IF FIRST PASS OF PROGRAM
2618      015316      001406      BEQ      1$          ::INHIBIT ITERATIONS
2619      015320      005237      001104      INC      $ICNT          ::INCREMENT ITERATION COUNT
2620      015324      023737      001160      001104      CMP      $TIMES,$ICNT      ::CHECK THE NUMBER OF ITERATIONS MADE
2621      015332      002024      BGE      $OVER          ::BR IF MORE ITERATION REQUIRED
2622      015334      012737      000001      001104      1$: MOV      #1,$ICNT          ::REINITIALIZE THE ITERATION COUNTER
2623      015342      013737      015420      001160      MOV      $MXCNT,$TIMES      ::SET NUMBER OF ITERATIONS TO DO
2624      015350      105237      001102      $SVLAD: INCB     $STNM          ::COUNT TEST NUMBERS
2625      015354      113737      001102      001174      MOVB     $STNM,$TESTN      ::SET TEST NUMBER IN APT MAILBOX
2626      015362      011637      001106      MOV      (SP),$LPADR      ::SAVE SCOPE LOOP ADDRESS
2627      015366      011637      001110      MOV      (SP),$LPERR      ::SAVE ERROR LOOP ADDRESS
2628      015372      005037      001162      CLR      $ESCAPE          ::CLEAR THE ESCAPE FROM ERROR ADDRESS
2629      015376      112737      000001      001115      MOVB     #1,$ERMAX          ::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2630      015404      013777      001102      163530      $OVER: MOV      $STNM,@DISPLAY      ::DISPLAY TEST NUMBER
2631      015412      013716      001106      MOV      $LPADR,(SP)      ::FUJGE RETURN ADDRESS
2632      015416      000002      RTI
2633      015420      000062      $MXCNT: 50.          ::MAX. NUMBER OF ITERATIONS
```

```

2534      .SBTTL  TTY INPUT ROUTINE
2535
2536      :*****
2537      .ENABL  LSB
2538
2539      :*****
2540      :*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2541      :*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WI'L
2542      :*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2543      :*WHEN OPERATING IN TTY FLAG MODE.
2544      015422 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ;; IS THE SOFT-SWR SELECTED?
2545      015430 001074          BNE      15$          ;; BRANCH IF NO
2546      015432 105777 163506      TSTB     @STKS          ;; CHAR THERE?
2547      015436 100071          BPL      15$          ;; IF NO, DON'T WAIT AROUND
2548      015440 117746 163502      MOVB     @STKB,-(SP)    ;; SAVE THE CHAR
2549      015444 042716 177600      BIC     #'C177,(SP)   ;; STRIP-OFF THE ASCII
2550      015450 022726 000007      CMP     #'7,(SP)+     ;; IS IT A CONTROL G?
2551      015454 001062          BNE      15$          ;; NO, RETURN TO USER
2552      015456 123727 001134 000001  CMPB     $AUTOB,#1    ;; ARE WE RUNNING IN AUTO-MODE?
2553      015464 001456          BFC     15$          ;; BRANCH IF YES
2554
2555      015466 104401 016147          TYPE     , $CNTLG     ;; ECHO THE CONTROL-G (^G)
2556      015472 104401 016154      $GTSWR: TYPE     , $MSWR     ;; TYPE CURRENT CONTENTS
2557      015476 013746 000176      MOV     SWREG,-(SP)   ;; SAVE SWREG FOR TYPEOUT
2558      015502 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2559      015504 104401 016165          TYPE     , $MNEW     ;; PROMPT FOR NEW SWR
2560      015510 005046          19$:  CLR     -(SP)    ;; CLEAR COUNTER
2561      015512 005046          CLR     -(SP)    ;; THE NEW SWR
2562      015514 105777 163424      7$:  TSTB     @STKS    ;; CHAR THERE?
2563      015520 100375          BPL     7$       ;; IF NOT TRY AGAIN
2564
2565      015522 117746 163420      MOVB     @STKB,-(SP)  ;; PICK UP CHAR
2566      015526 042716 177600      BIC     #'^C177,(SP) ;; MAKE IT 7-BIT ASCII
2567
2568
2569
2570      015532 021627 000025          9$:  CMP     (SP),#25   ;; IS IT A CONTROL-U?
2571      015536 001005          BNE     10$        ;; BRANCH IF NOT
2572      015540 104401 016142          TYPE     , $CNTLU    ;; YES, ECHO CONTROL-U (^U)
2573      015544 062706 000006      20$:  ADD     #6,SP     ;; IGNORE PREVIOUS INPUT
2574      015550 000757          BR     19$        ;; LET'S TRY IT AGAIN
2575
2576
2577      015552 021627 000015          10$:  CMP     (SP),#15   ;; IS IT A <CR>?
2578      015556 001022          BNE     16$        ;; BRANCH IF NO
2579      015560 005766 000004          TST     4(SP)      ;; YES, IS IT THE FIRST CHAR?
2580      015564 001403          BEQ     11$        ;; BRANCH IF YES
2581      015566 016677 000002 163344  MOV     2(SP),@SWR   ;; SAVE NEW SWR
2582      015574 062706 000006      11$:  ADD     #6,SP     ;; CLEAR UP STACK
2583      015600 104401 001165          14$:  TYPE     , $CRLF    ;; ECHO <CR> AND <LF>
2584      015604 123727 001135 000001  CMPB     $INTAG,#1   ;; RE-ENABLE TTY KBD INTERRUPTS?
2585      015612 001003          BNE     15$        ;; BRANCH IF NOT
2586      015614 012777 000100 163322  MOV     #10U,@STKS  ;; RE-ENABLE TTY KBD INTERRUPTS
2587      015622 000002          RTI          ;; RETURN
2588      015624 004737 013624          16$:  JSR     PC,$TYPEC   ;; ECHO CHAR
2589      015630 021627 000060          CMP     (SP),#60   ;; CHAR < 0?

```

```

2690 015634 002420          BIT      18$          ;;BRANCH IF YES
2691 015636 021627 000067    CMP      (SP),#67      ;;CHAR > 7?
2692 015642 003015          BGT      18$          ;;BRANCH IF YES
2693 015644 042726 000060    BIC      #60,(SP)+    ;;STRIP-OFF ASC:1
2694 015650 005766 000002    TST      2(SP)        ;;IS THIS THE FIRST CHAR
2695 015654 001403          BEQ      17$          ;;BRANCH IF YES
2696 015656 006316          ASL      (SP)         ;;NO, SHIFT PRESENT
2697 015660 006316          ASL      (SP)         ;;CHAR OVER TO MAKE
2698 015662 006316          ASL      (SP)         ;;ROOM FOR NEW ONE.
2699 015664 005266 000002    17$: INC      2(SP)        ;;KEEP COUNT OF CHAR
2700 015670 056616 177776    BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
2701 015674 000707          BR       7$           ;;GET THE NEXT ONE
2702 015676 104401 001164    18$: TYPE   ,SQUES    ;;TYPE ?<CR><LF>
2703 015702 000720          BR       20$          ;;SIMULATE CONTROL-U
2704
2705
2706
2707
2708 *****
2709 * THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2710 *CALL:
2711 * RDCHR      *          ;;INPUT A SINGLE CHARACTER FROM THE TTY
2712 * RETURN HERE          ;;CHARACTER IS ON THE STACK
2713 *
2714 *
2715 $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC
2716 MOV      4(SP),2(SP)          ;;SAVE THE PS
2717 1$: TSTB     @STKS             ;;WAIT FOR
2718 BPL      1$                   ;;A CHARACTER
2719 MOVB     @STKB,4(SP)          ;;READ THE TTY
2720 BIC      #^C<177>,4(SP)      ;;GET RID OF JUNK IF ANY
2721 CMP      4(SP),#23           ;;IS IT A CONTROL-S?
2722 BNE     3$                   ;;BRANCH IF NO
2723 2$: TSTB     @STKS             ;;WAIT FOR A CHARACTER
2724 BPL      2$                   ;;LOOP UNTIL ITS THERE
2725 MOVB     @STKB,-(SP)         ;;GET CHARACTER
2726 BIC      #^(177),(SP)        ;;MAKE IT 7-BIT ASCII
2727 CMP      (SP)+,#21           ;;IS IT A CONTROL-Q?
2728 BNE     2$                   ;;IF NOT DISCARD IT
2729 BR       1$                   ;;YES, RESUME
2730 3$: CMP      4(SP),#140        ;;IS IT UPPER CASE?
2731 BLT      4$                   ;;BRANCH IF YES
2732 CMP      4(SP),#175          ;;IS IT A SPECIAL CHAR?
2733 BGT      4$                   ;;BRANCH IF YES
2734 BIC      #40,4(SP)           ;;MAKE IT UPPER CASE
2735 4$: RTI                       ;;GO BACK TO USER
2736 *****
2737 * THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2738 *CALL:
2739 * RDLIN      *          ;;INPUT A STRING FROM THE TTY
2740 * RETURN HERE          ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2741 *
2742 *
2743 $RDLIN: MOV      R3,-(SP)       ;;SAVE R3
2744 1$: MOV      #TTYIN,R3        ;;GET ADDRESS
2745 2$: CMP      #TTYIN+8.,R3     ;;BUFFER FULL?

```



```

2746 016036 101405          BLOS      4$          ;;BR IF YES
2747 016040 104410          RDCHR                    ;;GO READ ONE CHARACTER FROM THE TTY
2748 016042 112613          MOV      (SP)+,(R3)    ;;GET CHARACTER
2749 016044 122713 000177 10$:  CMP      #177,(R3)    ;;IS IT A RUBOUT
2750 016050 001003          BNE      3$          ;;SKIP IF NOT
2751 016052 104401 001164 4$:  TYPE    ,SQUES      ;;TYPE A '?'
2752 016056 000763          BR       1$          ;;CLEAR THE BUFFER AND LOOP
2753 016060 111337 016130 3$:  MOV      (R3),9$      ;;ECHO THE CHARACTER
2754 016064 104401 016130  TYPE    ,9$
2755 016070 122723 000015  CMP      #15,(R3)+    ;;CHECK FOR RETURN
2756 016074 001356          BNE      2$          ;;LOOP IF NOT RETURN
2757 016076 105063 177777  CLRB    -1(R3)        ;;CLEAR RETURN (THE 15)
2758 016102 104401 001166  TYPE    ,SLF          ;;TYPE A LINE FEED
2759 016106 012603          MOV      (SP)+,R3     ;;RESTORE R3
2760 016110 011646          MOV      (SP),-(SP)   ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2761 016112 016666 000004 000002 MOV      4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
2762 016120 012766 016132 000004 MOV      #STTYIN,4(SP)
2763 016126 000002          RTI                    ;;RETURN
2764 016130 000          9$:  .BYTE   0            ;;STORAGE FOR ASCII CHAR. TO TYPE
2765 016131 000          .BYTE   0            ;;TERMINATOR
2766 016132 000010          $TTYIN: .BLKB      8.  ;;RESERVE 8 BYTES FOR TTY INPUT
2767 016142 052536 005015 000  $CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
2768 016147 136 006507 000012 $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
2769 016154 005015 053523 020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
2770 016162 020075 000          $MNEW:  .ASCIZ  / NEW = /
2771 016165 040 047040 053505
2772 016172 036440 000040
2773          .SBTTL  POWER DOWN AND UP ROUTINES
2774
2775          ;;*****
2776          ;;POWER DOWN ROUTINE
2777 016176 012737 016342 000024 $PWRDN: MOV      #SILLUP,@PWRVEC  ;;SET FOR FAST UP
2778 016204 012737 000340 000026 MOV      #340,@PWRVEC+2  ;;PRIO:7
2779 016212 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
2780 016214 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
2781 016216 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
2782 016220 010346          MOV      R3,-(SP)      ;;PUSH R3 ON STACK
2783 016222 010446          MOV      R4,-(SP)      ;;PUSH R4 ON STACK
2784 016224 010546          MOV      R5,-(SP)      ;;PUSH R5 ON STACK
2785 016226 017746 162706 MOV      @SWR,-(SP)     ;;PUSH @SWR ON STACK
2786 016232 010637 016346 MOV      SP,$SAVR6     ;;SAVE SP
2787 016236 012737 016250 000024 MOV      #SPWRUP,@PWRVEC ;;SFT UP VECTOR
2788 016244 000000          HALT
2789 016246 000776          BR       .-2          ;;HANG UP
2790
2791          ;;*****
2792          ;;POWER UP ROUTINE
2793 016250 012737 016342 000024 $PWRUP: MOV      #SILLUP,@PWRVEC  ;;SET FOR FAST DOWN
2794 016256 013706 016346 MOV      $SAVR6,SP     ;;GET SP
2795 016262 005037 016346 CLR      $SAVR6        ;;WAIT LOOP FOR THE TTY
2796 016266 005237 016346 1$:  INC      $SAVR6        ;;WAIT FOR THE INC
2797 016272 001375          BNE      1$          ;;OF WORD
2798 016274 012677 162640 MOV      (SP)+,@SWR    ;;POP STACK INTO @SWR
2799 016300 012605          MOV      (SP)+,R5     ;;POP STACK INTO R5
2800 016302 012604          MOV      (SP)+,R4     ;;POP STACK INTO R4
2801 016304 012603          MOV      (SP)+,R3     ;;POP STACK INTO R3

```

```

2802 016306 012602          MOV      (SP)+,R2          ;;POP STACK INTO R2
2803 016310 012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
2804 016312 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
2805 016314 012737 016176 000024  MOV      #PWRDN,@PWRVEC    ;;SET UP THE POWER DOWN VECTOR
2806 016322 012737 000340 000026  MOV      #340,@PWRVEC+2    ;;PRIO:7
2807 016330 104401          TYPE                                ;;REPORT THE POWER FAILURE
2808 016332 016350  $PWRMG: .WORD  PWRMSG          ;;POWER FAIL MESSAGE POINTER
2809 016334 012716          MOV      (PC)+,(SP)        ;;RESTART AT START1
2810 016336 001766  $PWRAD: .WORD  START1        ;;RESTART ADDRESS
2811 016340 000002          RTI
2812 016342 000000  $ILLUP: HALT                ;;THE POWER UP SEQUENCE WAS STARTED
2813 016344 000776          BR      .-?                ;; BEFORE THE POWER DOWN WAS COMPLETE
2814 016346 000000  $SAVR6: 0                    ;;PUT THE SP HERE
2815 016350 005015 042522 052123  PWRMSG: .ASCIZ <15><12>/RESTARTED FROM PWR FAIL/
2816 016356 051101 042524 020104
2817 016364 051106 046517 050040
2818 016372 051127 043040 044501
2819 016400 000114
2820
2821      .EVEN
2822      .SBTTL TRAP DECODER
2823
2824      ;*****
2825      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2826      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2827      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2828      ;*GO TO THAT ROUTINE.
2829 016402 010046  $TRAP:  MOV      R0,-(SP)          ;;SAVE R0
2830 016404 016600 000002  MOV      2(SP),R0          ;;GET TRAP ADDRESS
2831 016410 005740  TST      -(R0)              ;;BACKUP BY 2
2832 016412 111000  MOVVB   (R0),R0            ;;GET RIGHT BYTE OF TRAP
2833 016414 006300  ASL     R0                  ;;POSITION FOR INDEXING
2834 016416 016000 016436  MOV      $TRPAD(R0),R0      ;;INDEX TO TABLE
2835 016422 000200  RTS     R0                  ;;GO TO ROUTINE
2836
2837
2838      ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2839
2840 016424 011646  $TRAP2: MOV      (SP),-(SP)      ;;MOVE THE PC DOWN
2841 016426 016666 000004 00000?  MOV      4(SP),2(SP)        ;;MOVE THE PSW DOWN
2842 016434 000002  RTI                          ;;RESTORE THE PSW
2843
2844      .SBTTL TRAP TABLE
2845
2846      ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2847      ;*BY THE "TRAP" INSTRUCTION.
2848
2849      ;
2850      ; ROUTINE
2851      ;-----
2851 016436 016424  $TRPAD: .WORD  $TRAP2
2852 016440 013412  $TYPE   ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
2853 016442 014166  $TYPOC  ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2854 016444 014142  $TYPOS  ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2855 016446 014202  $TYPON  ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2856 016450 014370  $TYPDS  ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
2857

```

2858	016452	015472			\$GTSWR	::CALL=GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING
2859								
2860	016454	015422			\$CKSWR	::CALL=CKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR
2861	016456	015704			\$RDCHR	::CALL=RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE
2862	016460	016024			\$RDLIN	::CALL=RDLIN	TRAP+11(104411)	TTY TYPEIN STRING ROUTINE
2863								
2864					.SBTTL	ASCII MESSAGES		
2865	016462	005015	053103	051104	TITLED:	.ASCIZ <15><12>/CVDRCA	DRV11J DIAG TEST PART 1	/<15><12>
2866	016470	040503	020040	051104				
2867	016476	030526	045061	042040				
2868	016504	040511	020107	042524				
2869	016512	052123	050040	051101				
2870	016520	020124	020061	006440				
2871	016526	000012						
2872	016530	005015	051104	030526	TLCABL:	.ASCIZ <15><12>/DRV11J	CABLE REQ'D/<15><12>	
2873	016536	045061	041440	041101				
2874	016544	042514	051040	050505				
2875	016552	042047	005015	000				
2876								
2877	016557	122	043505	052040	EM1:	.ASCIZ /REG TIMEOUT ER/		
2878	016564	046511	047505	052125				
2879	016572	042440	000122					
2880	016576	042522	020107	042522	EM2:	.ASCIZ 'REG READ/WRITE ER'		
2881	016604	042101	053457	044522				
2882	016612	042524	042440	000122				
2883	016620	051111	020122	042522	EM3:	.ASCIZ /IRR REG ER/		
2884	016626	020107	051105	000				
2885	016633	101	051103	051040	EM4:	.ASCIZ /ACR REG ER/		
2886	016640	043505	042440	000122				
2887	016646	046511	020122	042522	EM5:	.ASCIZ /IMR REG ER/		
2888	016654	020107	051105	000				
2889	016661	111	051123	051040	EM6:	.ASCIZ /ISR REG ER/		
2890	016666	043505	042440	000122				
2891	016674	044103	050111	051440	EM7:	.ASCIZ /CHIP STAT ER/		
2892	016702	040524	020124	051105				
2893	016710	000						
2894	016711	105	051122	041520	DH1:	.ASCIZ /ERRPC TSTNUM	BUSADR EXPCT RCVD/	
2895	016716	020040	052040	052123				
2896	016724	052516	020115	041040				
2897	016732	051525	042101	020122				
2898	016740	042440	050130	052103				
2899	016746	020040	051040	053103				
2900	016754	000104						
2901	016756	051105	050122	020103	DH2:	.ASCIZ /ERRPC TSTNUM	BUSADR ADRS EXPCT RCVD/	
2902	016764	020040	051524	047124				
2903	016772	046525	020040	052502				
2904	017000	040523	051104	020040				
2905	017006	042101	051522	020040				
2906	017014	020040	054105	041520				
2907	017022	020124	020040	041522				
2908	017030	042126	000					
2909								
2910		017034			.EVEN			
2911	017034	001116	001362	001122	DT1:	\$ERRPC, TSTNUM, \$BDADR, \$GDDAT, \$BDDAT, 0		
2912	017042	001124	001126	000900				
2913	017050	001116	001362	001122	DT2:	\$ERRPC, TSTNUM, \$BDADR, \$GDADR, \$GDDAT, \$BDDAT, 0		

2914 017056 001120 001124 001126
2915 017064 000000
2916
2917 000001

.END

CVDRCA DRV1J DIAG TST PRT1
CVDRCA.P11 18-OCT-79 12:53

MACY11 30A(1052) 18-OCT-79 12:58 PAGE 64
CROSS REFERENCE TABLE -- USER SYMBOLS

SW2	=	000004	90#			
SW3	=	000010	89#			
SW4	=	000020	88#			
SW5	=	000040	87#			
SW6	=	000100	86#			
SW7	=	000200	85#			
SW8	=	000400	84#			
SW9	=	001000	83#			
TBITVE	=	000014	125#			
TITLED		016462	462	2865#		
TKVEC	=	000060	132#			
TECABL		016530	463	2872#		
TPVEC	=	000064	133#			
TRAPVE	=	000034	131#	411*	412*	
TRTVEC	=	000014	126#			
TSTNUM		001362	389#	2519*	2911	2913
TST1		002074	492#			
TST10		003260	699	705#		
TST11		003354	727#			
TST12		003450	750#			
TST13		003544	773#			
TST14		003640	796#			
TST15		004010	828	834#		
TST16		004156	865	872#		
TST17		004456	930#			
TST2		002152	511#			
TST20		004744	985#			
TST21		005244	1042#			
TST22		005532	1096#			
TST23		005666	1121	1129#		
TST24		006076	1168	1176#		
TST25		006302	1214	1222#		
TST26		006506	1260	1269#		
TST27		006624	1289	1297#		
TST3		002344	545	551#		
TST30		006742	1317	1325#		
TST31		007034	1339	1347#		
TST32		007134	1362	1370#		
TST33		007262	1393	1402#		
TST34		007410	1425	1433#		
TST35		007510	1448	1457#		
TST36		007612	1473	1481#		
TST37		007744	1505	1513#		
TST4		002454	569	575#		
TST40		010076	1537	1545#		
TST41		010222	1567	1575#		
TST42		010376	1605	1613#		
TST43		010604	1650	1662#		
TST44		011120	1715	1724#		
TST45		011406	1785#			
TST46		011666	1836	1847#		
TST47		012154	1908#			
TST5		002552	599	599#		
TST50		012364	1947#			
TST51		012502	1966	1973#		
TST6		002720	1973#			

\$\$\$SKIP	1#	135#	545	569	592	627	665	699	828	865	1121	1168	1214	1260	1289
	1317	1339	1362	1393	1425	1448	1473	1505	1537	1567	1605	1650	1715	1836	1966
.EQUAT	1#	25													
.HEADE	1#														
.K*11	1#														
.SETUP	1#	397													
.SWRHI	1#	15													
.SWRLO	25#														
.\$ACT1	1#	187													
.\$APT8	1#	266#													
.\$APTH	1#	201													
.\$APTY	1#	2267													
.\$ASTA	1#														
.\$CATC	1#	173													
.\$CMTA	1#	223													
.\$DB2D	1#														
.\$DB20	1#														
.\$DIV	1#														
.\$EOP	1#	2009													
.\$ERRO	1#	2468													
.\$ERRT	1#	2523													
.\$MULT	1#														
.\$POWE	1#	2773													
.\$RAND	1#														
.\$RDDE	1#														
.\$RDOC	1#														
.\$READ	1#	2634													
.\$R2AZ	1#														
.\$SAVE	1#														
.\$SB2D	1#														
.\$SB20	1#														
.\$SCOP	1#	2570													
.\$SIZE	1#														
.\$SUPR	1#														
.\$TRAP	1#	2821													
.\$TYPB	1#														
.\$TYPD	1#	2401													
.\$TYPE	1#	2188													
.\$TYPO	1#	2324													
.\$4OCA	1#														
.1170	1#														

. ABS. 017066 000

ERRORS DETECTED: 0

DSKZ:CVDRCA,DSKZ:CVDRCA/CRF/SOL=DSKZ:SYSMAC.SML,DSKZ:CVDRCA.P11
RUN-TIME: 51 52 3 SECONDS
RUN-TIME RATIO: 331/108=3.0
CORE USED: 33K (65 PAGES)