

DRV11B

DRV11B DMA INTFC DIAG AH-T086B-MC  
CVDRABO FICHE 1 OF 1

MAR 1982  
COPYRIGHT © 76-81  
MADE IN USA



.REM %

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

IDENTIFICATION  
-----

PRODUCT CODE: AC-8178B-MC  
PRODUCT NAME: CVDRAB0 DRV11B DMA INTFC DIAG  
DATE: AUGUST 1981  
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1976,1981  
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE  
COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE  
COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT  
BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR  
USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE  
TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND  
SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE  
ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98

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 DRV11B BUS & VECTOR ADDRESS MODIFICATION  
7.2 XXDP/APT NOTES  
7.3 POWER FAIL  
7.4 MULTIPLE DRV11B INTERFACE TESTING  
7.5 RESTRICTIONS

8.0 EXECUTION TIME

9.0 PROGRAM TEST DESCRIPTIONS  
9.1 GENERAL  
9.2 REGISTER TESTS  
9.3 BYTE ADDRESSING TESTS  
9.4 'FNCT' TO 'STAT' WRAP AROUND TEST  
9.5 READY INTERRUPT TEST  
9.6 NPR DATA TRANSFER TESTS  
9.7 MAINT MODE NPR DATA TRANSFER TESTS  
9.8 BURST & NON-BURST MODE TESTS  
9.9 'NEX' ERROR CONDITION TEST

10.0 LISTING

100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155

1.0 ABSTRACT

-----  
THE DRV11B DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS ACCESSIBLE WITH THE LOOP BACK CABLE INSERTED IN THE USER I/O CONNECTORS. TOTAL PROGRAM CONTROL IS ACCOMPLISHED THRU THE CONSOLE TERMINAL VIA THE ODT/CONSOLE MICROCODE AND THE PROVISIONS OF SECTION 5 OF THIS DOCUMENT. IF THE SYSTEM ALSO INCLUDES AN 'REV11' (DMA REFRESH), THE DMA REFRESH MUST BE DISABLED AND CPU REFRESH MUST BE ENABLED.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. LSI-11 FAMILY PROCESSOR.  
11/03(LSI-11/02), 11/23(KDF11-A), 11/23B(KDF11-B)
2. DLV11 WITH I/O TYPE TERMINAL
3. DRV11B WITH LOOP BACK CABLE

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE

- IF USING PAPER TAPE READER FOLLOW THIS PROCEDURE:
1. ASSURE THAT THE LSI-11 IS IN THE ODT MICROCODE STATE.
  2. LOAD THE LOW OR HIGH SPEED READER WITH THE ABSOLUTE LOADER TAPE.
  3. TYPE THE READER'S CSR ADDRESS (177560-LOW OR 177550-HIGH) AND CHARACTER 'L'.
  4. AFTER TAPE IS LOADED, LOAD THE DRV11B BINARY TAPE INTO THE READER AND TYPE THE CHARACTER 'P'.
  5. IF THE ABSOLUTE LOADER HAS ALREADY BEEN LOADED (STEPS 2 & 3), THEN ONLY THE STARTING ADDRESS OF THE ABSOLUTE LOADER AND THE CHARACTER 'G' NEED BE TYPED (WITH THE DRV11B BINARY TAPE IN THE APPROPRIATE READER).

4.0 STARTING PROCEDURE

- FOR PAPER TAPE MEDIA:
1. MAKE SURE THE MAINTENANCE LOOP BACK CABLE IS INSERTED IN THE I/O CONNECTORS ON THE M7950 MODULE.
  2. MAKE SURE THE DEVICE BUS & VECTOR ADDRESSES AGREE WITH THE DEFAULT VALUES DEFINED IN SECTION 7.1. IF NOT, CHANGE LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' ODT COMMAND.
  3. INSURE THAT THE HALT SWITCH IS DISABLED (IF ANY).
  4. TYPE THE STARTING ADDRESS OF 200 AND THE CHARACTER G.
  5. THE PROGRAM WILL RESPOND BY TYPING THE SOFTWARE SWITCH REGISTER CONTENTS AND ALLOWING THE USER TO CHANGE ITS CONTENTS BY ENTERING OCTAL SWITCH REGISTER DATA TERMINATED BY A CARRIAGE RETURN - SEE SECTION 5.0 FOR SWITCH REGISTER

156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173

- OPTIONS.  
6. THE NEXT QUESTION ASKED IS ABOUT THE TYPE OF A PROCESSOR.  
TYPE 'Y' WITH CARRIGE RETURN IF USING KDF11-B, 'N' OTHER-  
WISE.

- IF RUNNING UNDER XXDP+ MONITOR:  
A) DO 1., 2., 3. OF THE ABOVE.  
B) IN MONITOR MODE TYPE IN 'R VDRAB?'.  
C) DO 5. AND 6.

\*\*\*\*\*  
IF USING LSI-11/23R(KDF11-B) AND WANT TO TEST DMA TRANSFERS TO  
I/O PAGE PUT THE ADDRESS OF YOUR I/O INTERFACE CONTROL RE-  
GISTER INTO LOCATION 1544 (IOPAGE). TO DO THIS PERMANENTLY USE  
LOAD-MOD-DUMP PROCEDURE DEFINED IN 7.2. FOR TEMPORARY CHANGES  
JUST MODIFY LOCATIONS AFTER LOADING THE PROGRAM.  
NOTE: THIS TEST IS NOT GOING TO BE PERFORMED UNLESS THE ABOVE  
MENTIONED LOCATIONS ARE MODIFIED.

175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
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  
223  
224  
225  
226  
227  
228  
229  
230

5.0 SOFTWARE SWITCH REGISTER  
-----

5.1 OPTIONS

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
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <5-0>
***SW07=1	000200	DEVICE IS KDF11-B

\*\*\*DIAGNOSTIC WILL SET THIS SWITCH AUTOMATICALLY, IF THIS TYPE OF A PROCESSOR HAS BEEN CONFIRMED IN A DIALOGUE. IF RUNNING IN AUTOMATIC MODE (CHAINS UNDER XXDP OR APT) SEE SECTION 7.2

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).
4. IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL 'CONTROL & G' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGE BY THE PROGRAM.

6.0 ERROR REPORTING  
-----

6.1 ERROR COMMENT

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

6.2 ERROR DATA

*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*TSTNUM	TEST NUMBER WHERE THE ERROR OCCURRED
BUSADR	DRV11B BUS REG ADDRESS OF CONCERNED OPERATION
EXPCT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED
ADRS	MEMORY ADDRESS OF DATA TRANSFER ON ERROR

231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286

7.0 \*ALWAYS REPORTED  
MISCELLANEOUS  
-----

7.1 DRV11B BUS & VECTOR ADDRESS MODIFICATION

MODIFY LOCATION '\$BASE' IF BASE BUS ADDRESS IS NOT 172410.  
MODIFY LOCATION '\$VECT1' IF VECTOR ADDRESS IS NOT 124.

\*NOTE: USE THE LSI-11 ODT FACILITIES TO MODIFY THESE LOCATIONS  
AFTER PROGRAM LOAD. NO VECTOR ASSIGNMENT ABOVE 774 SHOULD BE  
ALLOWED.

7.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REF. 7.5)(REQUIRES 8K OR MORE).  
THIS DIAGNOSTIC DOES SUPPORT "APT" AND HAS RUN UNDER IT.

IF THE PROCESSOR USED IS KDF11-B:  
FOR APT SET \$SWREG BIT07 TO 1 (000200)  
FOR XXDP CHAINS SET LOCATION 176 (SWR) BIT07 TO 1  
TO DO THIS UNDER XXDP:

1. R UPD2
2. LOAD VDRAB?.BIC
3. MOD 176  
THE TERMINAL WILL RESPOND: 176/000000
4. NOW TYPE IN 200
5. BE CAREFUL: AT THIS POINT IT IS NECESSARY TO DELETE THE FILE  
FROM THE DISK.  
DEL DLO:VDRAB?.BIC (IF MEDIA IS RL ON DRIVE 0)
6. DUMP VDRAB0.BIC  
TO BE SAFE, SKIP 5. AND DUMP THE FILE UNDER DIFFERENT NAME  
WITH .BIC EXTENSION.

NOTE: THIS PROCEDURE ASSUMES THAT DIAGNOSTIC IS ON THE DISK  
FROM WHICH THE SYSTEM IS BOOTED. IF THIS IS UNTRUE  
IN 2. AND 6. THE OPERATOR HAVE TO SPECIFY THE DRIVE BEFORE  
THE NAME.

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 DRV11B INTERFACE TESTING

THIS PROGRAM DOES NOT "AUTO-SIZE" THE NUMBER OF DRV11B'S CONNECTED.  
THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 DRV11B INTERFACES  
WITH CONTIGUOUS BUS AND VECTOR ADDRESSES. THIS IS ACCOMPLISHED  
BY THE OPERATOR SETTING UP LOCATION '\$DEVM' WITH A BIT MAP INDICATING WHAT  
INTERFACES ARE TO TESTED. I.E. BIT0=1 SAYS TEST 1ST DRV11B,  
BIT1=1 SAYS TEST 2ND DRV11B, BIT2=1 SAYS TEST 3RD DRV11B, ETC..

7.5 RESTRICTIONS

IF THE SYSTEM ALSO INCLUDES AN "REV11" (DMA REFRESH), THE DMA

287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
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

REFRESH MUST BE DISABLED AND CPU REFRESH MUST BE ENABLED.

8.0 EXECUTION TIME  
-----

EXECUTION TIME RANGES FROM ABOUT 5 SECONDS WITH NO ITERATIONS  
TO ABOUT 90 SECONDS WITH ITERATIONS ENABLED WITH ONE DRV11B CONNECTED.  
AN END PASS MESSAGE INDICATES ALL TESTS HAVE COMPLETED ON ALL SELECTED UNITS.

9.0 PROGRAM TEST DESCRIPTIONS  
-----

9.1 GENERAL

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED  
TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE DRV11B DMA INTERFACE.  
A HIGH DEGREE OF TESTING IS ACCOMPLISHED WITH THE AID OF THE  
MAINTENANCE 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.

9.2 REGISTER TESTS

THE FOLLOWING REGISTERS ARE READ/WRITE & RESET TESTED:

1. WORD COUNT
2. BUFFER ADDRESS
3. COMMAND/STATUS
4. DATA BUFFER

9.3 BYTE ADDRESSING TESTS

1. COMMAND/STATUS
2. DATA BUFFER

9.4 'FNCT' TO 'STAT' WRAP AROUND TEST

9.5 READY INTERRUPT TEST

9.6 NPR DATA TRANSFER TESTS

THE FOLLOWING NPR XFERS ARE CHECKED FOR CORRECT STATUS,  
WORD COUNT, BUFFER ADDRESS & DATA:

1. SINGLE 'DATI' XFER - FLOATING 1/0 PTRN
2. SINGLE 'DATO' XFER - FLOATING 1/0 PTRN
3. 200 'DATI' XFERS - FLOATING 1/0 PTRN
4. 200 'DATO' XFERS - FLOATING 1/0 PTRN
5. SINGLE 'DATI' XFER TO THE TTY PRINTER CSR  
FOR KDF11-B PROCESSOR THE USER HAVE TO SELECT  
CSR OF A PARTICULAR INTERFACE (DISK INTERFACES  
ARE SUGGESTED). SOME OF THEM ARE:  
RXV11 (RX01): 177170  
RXV21 (RX02): 177170  
RKV11-D (RK05): 177404



343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359

- 9.7 MAINT MODE NPR DATA TRANSFER TESTS
  - 1. THAT MAINT MODE CONTROLS 'FNCT' BITS
  - 2. 200 MAINT MODE XFERS - CHECKING STATUS & DATA
  - 3. 200 MAINT MODE XFERS TO EACH 4K AVAILABLE MEM
- 9.8 BURST & NON-BURST MODE TESTS
  - 1. THAT CPU IS LOCKED OUT IN BURST MODE
  - 2. THAT CPU IS NOT LOCKED OUT IN NON-BURST MODE
- 9.9 'NEX' ERROR CONDITION TEST
- 10.0 LISTING  
-----  
%

```
371 .TITLE MAINDEC-11-CVDRA-B DRV11B DMA INTERFACE DIAGNOSTIC
(1) ;*COPYRIGHT (C) 1981
(1) ;*DIGITAL EQUIPMENT CORP.
(1) ;*MAYNARD, MASS. 01754
(1) ;*
(1) ;*PROGRAM BY R. MOORE
(1) ;*
(1) ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1) ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
(1) ;*
(1) 000001 $TN=1
(1) 160000 $$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
372 167400 $$SWR=167400
373 000300 $$SWRMK=300
374 000001 $TN=1
375 .SBTTL OPERATIONAL SWITCH SETTINGS
(1) ;*
(1) ;* SWITCH USE
(1) ;* -----
(1) ;* 15 HALT ON ERROR
(1) ;* 14 LOOP ON TEST
(1) ;* 13 INHIBIT ERROR TYPEOUTS
(1) ;* 11 INHIBIT ITERATIONS
(1) ;* 10 BELL ON ERROR
(1) ;* 9 LOOP ON ERROR
(1) ;* 8 LOOP ON TEST IN SWR<5:0>
376 .SBTTL BASIC DEFINITIONS
(1) ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1) 001100 STACK= 1100
(1) .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
(1) .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
(1) ;*MISCELLANEOUS DEFINITIONS
(1) 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
(1) 000012 LF= 12 ;;CODE FOR LINE FEED
(1) 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
(1) 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1) 177776 PS= 177776 ;;PROCESSOR STATUS WORD
(1) 177774 .EQUIV PS,PSW
(1) 177772 STKLMT= 177774 ;;STACK LIMIT REGISTER
(1) 177570 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1) 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1) 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1) ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1) 000000 R0= %0 ;;GENERAL REGISTER
(1) 000001 R1= %1 ;;GENERAL REGISTER
(1) 000002 R2= %2 ;;GENERAL REGISTER
(1) 000003 R3= %3 ;;GENERAL REGISTER
(1) 000004 R4= %4 ;;GENERAL REGISTER
(1) 000005 R5= %5 ;;GENERAL REGISTER
(1) 000006 R6= %6 ;;GENERAL REGISTER
(1) 000007 R7= %7 ;;GENERAL REGISTER
(1) 000006 SP= %6 ;;STACK POINTER
(1) 000007 PC= %7 ;;PROGRAM COUNTER
```



L 1

```

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

(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;: "T" BIT
(1) 000014 TRTVEC= 14 ;:TRACE TRAP
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;:POWER FAIL
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;: "TRAP" TRAP
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
377 172410 ABASE= 172410 ;:BASE DRV11B BUS ADRS EQUATE
378 000124 AVECT1= 000124 ;:BASE DRV11B VECTOR ADRS EQUATE -
379 000001 ADEVM= 1 ;:DEFAULT TO ONE DRV11B
380 106427 MTPS=106427 ;:INSTR EQUATE THAT MOVES BYTE TO PSW
381 000000 TMAIN: 177522 ;:MAINTENCE REGISTER(FOR USE WITH KDF11-B)
382 .SBTTL TRAP CATCHER

(1) ;*=0
(1) 000000 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1) ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1) ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

(1) ;*=174
(1) 000174 000000 DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
(1) 000176 000000 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER
(1) .SBTTL STARTING ADDRESS(ES)
(1) 000200 000137 001550 JMP @#START ;:JUMP TO STARTING ADDRESS OF PROGRAM
383 000100 ;*=100
384 000100 000104 000200 000002 .WORD 104,200,2 ;:IF 'B EVENT' ON Q BUS IS CONNECTED
385 ;:IGNORE IT'S INTERRUPT - JUST DO A RTI
    
```

```

387      .SBTTL  ACT11 HOOKS
(1)
(2)      ::*****
(1)      ;HOOKS REQUIRED BY ACT11
(1)      $SVPC=.          ;SAVE PC
(1)      .=46
(1)      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1) 000046 010056
(1)      .=52
(1) 000052 000000      .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)      .=$SVPC        ;; RESTORE PC
(1)      .=1000
388      001000
389      .SBTTL  APT PARAMETER BLOCK
(1)
(2)      ::*****
(1)      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      ::*****
(1)      .SX=.          ;;SAVE CURRENT LOCATION
(1)      .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200      200          ;;FOR APT START UP
(1)      .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000      $APTHDR      ;;POINT TO APT HEADER BLOCK
(1)      .=$X          ;;RESET LOCATION COUNTER
(2)      ::*****
(1)      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)      ;INTERFACE SPEC.
(1)
(1) 001000      $APTHD:
(1) 001000 000000      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001174      $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000031      $STMT: .WORD 25.      ;;RUN TIM OF LONGEST TEST
(1) 001006 000006      $PASTM: .WORD 6.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000144      $UNITM: .WORD 100.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000052      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
    
```

.SBTTL COMMON TAGS

- 390
- (1)
- (2)
- (1)
- (1)
- (1)
- (1)
- (1) 001100
- (1) 001100 000000
- (1) 001102 000
- (1) 001103 000
- (1) 001104 000000
- (1) 001106 000000
- (1) 001110 000000
- (1) 001112 000000
- (1) 001114 000
- (1) 001115 001
- (1) 001116 000000
- (1) 001120 000000
- (1) 001122 000000
- (1) 001124 000000
- (1) 001126 000000
- (1) 001130 000000
- (1) 001132 000000
- (1) 001134 000
- (1) 001135 000
- (1) 001136 000000
- (1) 001140 177570
- (1) 001142 177570
- (1) 001144 177560
- (1) 001146 177562
- (1) 001150 177564
- (1) 001152 177566
- (1) 001154 000
- (1) 001155 002
- (1) 001156 012
- (1) 001157 000
- (1) 001160 000000
- (1) 001162 000000
- (1) 001164 177607 000377
- (1) 001170 077
- (1) 001171 015
- (1) 001172 000012
- (2)
- (2)
- (2)
- (3)
- (2)
- (2) 001174
- (2) 001174 000000
- (2) 001176 000000
- (2) 001200 000000
- (2) 001202 000000
- (2) 001204 000000
- (2) 001206 000000
- (2) 001210 000000

```

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

        .=1100
$CMTAG: .WORD 0          ;;START OF COMMON TAGS
$TSTNM: .BYTE 0          ;;CONTAINS THE TEST NUMBER
$ERFLG: .BYTE 0          ;;CONTAINS ERROR FLAG
$IICNT: .WORD 0          ;;CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD 0          ;;CONTAINS SCOPE LOOP ADDRESS
$LPERR: .WORD 0          ;;CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD 0          ;;CONTAINS TOTAL ERRORS DETECTED
$IITEMB: .BYTE 0        ;;CONTAINS ITEM CONTROL BYTE
$ERMAX: .BYTE 1          ;;CONTAINS MAX. ERRORS PER TEST
$ERKPC: .WORD 0          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD 0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD 0          ;;CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT: .WORD 0          ;;CONTAINS 'GOOD' DATA
$BDDAT: .WORD 0          ;;CONTAINS 'BAD' DATA
        .WORD 0          ;;RESERVED--NOT TO BE USED
$AUTOB: .BYTE 0          ;;AUTOMATIC MODE INDICATOR
$INTAG: .BYTE 0          ;;INTERRUPT MODE INDICATOR
        .WORD 0
$SWR:   .WORD  DSWR      ;;ADDRESS OF SWITCH REGISTER
$DISP:  .WORD  DDISP     ;;ADDRESS OF DISPLAY REGISTER
$TKS:   177560          ;;TTY KBD STATUS
$TKB:   177562          ;;TTY KBD BUFFER
$TPS:   177564          ;;TTY PRINTER STATUS REG. ADDRESS
$TPB:   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
$NULL:  .BYTE  0        ;;CONTAINS NULL CHARACTER FOR FILLS
$FILLS: .BYTE  ?        ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC: .BYTE  12       ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
$STPFLG: .BYTE  0       ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
$TIMES: 0              ;;MAX. NUMBER OF ITERATIONS
$ESCAPE:0             ;;ESCAPE ON ERROR ADDRESS
$BELL:   .ASCIZ <207><377><377> ;;CODE FOR BELL
$QUES:   .ASCII  /?/    ;;QUESTION MARK
$CRLF:   .ASCII <15>    ;;CARRIAGE RETURN
$LF:     .ASCIZ <12>    ;;LINE FEED
*****

.SBTTL APT MAILBOX-ETABLE

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

```

```

(2) 001212 000000 $MSG LG: .WORD AMSGLG ;;MESSAGE LENGTH
(2) 001214 $ETABLE: ;;APT ENVIRONMENT TABLE
(2) 001214 000 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
(2) 001215 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
(2) 001216 000000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
(2) 001220 000000 $USWR: .WORD AUSWR ;;USER SWITCHES
(2) 001222 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
(2) * BITS 15-11=CPU TYPE
(2) * 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2) * 11/70=06,PDQ=07,Q=10
(2) * BIT 10=REAL TIME CLOCK
(2) * BIT 9=FLOATING POINT PROCESSOR
(2) * BIT 8=MEMORY MANAGEMENT
(2) 001224 000 $MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
(2) 001225 000 $MTYP1: .BYTE AMTYP1 ;;MEM. TYPE,BLK#1
(2) * MEM.TYPE BYTE -- (HIGH BYTE)
(2) * 900 NSEC CORE=001
(2) * 300 NSEC BIPOLAR=002
(2) * 500 NSEC MOS=003
(2) 001226 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
(2) * MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2) 001230 000 $MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
(2) 001231 000 $MTYP2: .BYTE AMTYP2 ;;MEM. TYPE,BLK#2
(2) 001232 000000 $MADR2: .WORD AMADR2 ;;MEM.LAST ADDRESS,BLK#2
(2) 001234 000 $MAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
(2) 001235 000 $MTYP3: .BYTE AMTYP3 ;;MEM. TYPE,BLK#3
(2) 001236 000000 $MADR3: .WORD AMADR3 ;;MEM.LAST ADDRESS,BLK#3
(2) 001240 000 $MAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
(2) 001241 000 $MTYP4: .BYTE AMTYP4 ;;MEM. TYPE,BLK#4
(2) 001242 000000 $MADR4: .WORD AMADR4 ;;MEM.LAST ADDRESS,BLK#4
(2) 001244 000124 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001246 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001250 172410 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001252 000001 $DEV M: .WORD ADEV M ;;DEVICE MAP
(2) 001254 000000 $CDW1: .WORD ACDW1 ;;CONTROLLER DESCRIPTION WORD#1
(2) 001256 000000 $CDW2: .WORD ACDW2 ;;CONTROLLER DESCRIPTION WORD#2
(2) 001260 000000 $DDW0: .WORD ADDW0 ;;DEVICE DESCRIPTION WORD#0
(2) 001262 000000 $DDW1: .WORD ADDW1 ;;DEVICE DESCRIPTION WORD#1
(2) 001264 000000 $DDW2: .WORD ADDW2 ;;DEVICE DESCRIPTION WORD#2
(2) 001266 000000 $DDW3: .WORD ADDW3 ;;DEVICE DESCRIPTION WORD#3
(2) 001270 000000 $DDW4: .WORD ADDW4 ;;DEVICE DESCRIPTION WORD#4
(2) 001272 000000 $DDW5: .WORD ADDW5 ;;DEVICE DESCRIPTION WORD#5
(2) 001274 000000 $DDW6: .WORD ADDW6 ;;DEVICE DESCRIPTION WORD#6
(2) 001276 000000 $DDW7: .WORD ADDW7 ;;DEVICE DESCRIPTION WORD#7
(2) 001300 000000 $DDW8: .WORD ADDW8 ;;DEVICE DESCRIPTION WORD#8
(2) 001302 000000 $DDW9: .WORD ADDW9 ;;DEVICE DESCRIPTION WORD#9
(2) 001304 000000 $DDW10: .WORD ADDW10 ;;DEVICE DESCRIPTION WORD#10
(2) 001306 000000 $DDW11: .WORD ADDW11 ;;DEVICE DESCRIPTION WORD#11
(2) 001310 000000 $DDW12: .WORD ADDW12 ;;DEVICE DESCRIPTION WORD#12
(2) 001312 000000 $DDW13: .WORD ADDW13 ;;DEVICE DESCRIPTION WORD#13
(2) 001314 000000 $DDW14: .WORD ADDW14 ;;DEVICE DESCRIPTION WORD#14
(2) 001316 000000 $DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTION WORD#15
(2)
(2)
(2) 001320 $ETEND:
(2)
    
```







```

484 ;ERROR 20
485 001510 015004 EM20 ;DATA ER FROM I/O PAGE (XCSR)
486 001512 015106 DH2 ;ERRPC TSTNUM BUSADR ADRS EXPCT RCVD
487 001514 015226 DT2 ;$ERRPC TSTNUM $BDADR $GDADR $GDDAT $BDCAT
488 001516 000000 0
489
490
491 ;DRV11B BUS REGISTER ADDRESS POINTERS
492
493 001520 172410 DRVWCR: 172410 ;WORD COUNT
494 001522 172412 DRVBAR: 172412 ;BUFFER ADDRESS
495 001524 172414 DRVCSR: 172414 ;COMMAND/STATUS
496 001526 172416 DRVDDBR: 172416 ;DATA BUFFER
497
498 ;DRV11B VECTOR ADDRESS POINTERS
499
500 001530 000124 DRVCT0: 124 ;READY, NEX & INCOMPLETE DATIO VECTOR
501 001532 000126 DRVCT2: 126 ;NEW PSW ON INTR
502
503 ;COMMON PROGRAM LOCATION(S)
504
505 001534 000000 TSTNUM: 0 ;CONTAINS TEST NUMBER ON ERROR
506 001536 000001 DMAP: 1 ;DEVICE MAP - EA BIT SAYS TEST THAT DRV11B
507 001540 000000 CORSZ: 0 ;CONTAINS 1ST NON-EXISTANT MEM ADRS
508 001542 015400 DBUFP: DBUF ;CONTAINS CURRENT 4K NPR BUFFER ADRS
509 001544 000000 IOPAGE: 0 ;CONTAINS ADDRESS FOR I/O TRANSFERS
510 001546 000000 KDF: 0 ;IF KDF11-B
  
```

```

513          .SBTTL PROGRAM START
514 001550   START:
(1)          .SBTTL INITIALIZE THE COMMON TAGS
(1)          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001550   012706 001100   MOV    # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 001554   005026         CLR    (R6)+           ;;CLEAR MEMORY LOCATION
(1) 001556   022706 001140   CMP    #SWR,R6      ;;DONE?
(1) 001562   001374         BNE    -6             ;;LOOP BACK IF NO
(1) 001564   012706 001100   MOV    #STACK,SP    ;;SETUP THE STACK POINTER
(1)          ;;INITIALIZE A FEW VECTORS
(1) 001570   012737 012446 000020  MOV    # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001576   012737 000340 000022  MOV    #340,@#IOTVEC+2 ;;LEVEL 7
(1) 001604   012737 012104 000030  MOV    # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001612   012737 000340 000032  MOV    #340,@#EMTVEC+2 ;;LEVEL 7
(1) 001620   012737 014066 000034  MOV    # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001626   012737 000340 000036  MOV    #340,@#TRAPVEC+2;LEVEL 7
(1) 001634   012737 013662 000024  MOV    # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 001642   012737 000340 000026  MOV    #340,@#PWRVEC+2 ;;LEVEL 7
(1) 001650   005037 001160         CLR    $TIMES        ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001654   005037 001162         CLR    $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001660   112737 000001 001115  MOV    #1,$ERMAX     ;;ALLOW ONE ERROR PER TEST
(1) 001666   012737 001666 001106  MOV    #.,$LPADR     ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001674   012737 001674 001110  MOV    #.,$LPERR     ;;SETUP THE ERROR LOOP ADDRESS
(2)          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001702   013746 000004         MOV    @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 001706   012737 001742 000004  MOV    #64$,@#ERRVEC ;;SET UP ERROR VECTOR
(2) 001714   012737 177570 001140  MOV    #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001722   012737 177570 001142  MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001730   022777 177777 177202  CMP    #-1,@SWR     ;;TRY TO REFERENCE HARDWARE SWR
(2) 001736   001012         BNE    66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)          ;;AND THE HARDWARE SWR IS NOT = -1
(2) 001740   000403         BR    65$          ;;BRANCH IF NO TIMEOUT
(2) 001742   012716 001750 64$:   MOV    #65$,(SP)    ;;SET UP FOR TRAP RETURN
(2) 001746   000002         RTI
(2) 001750   012737 000176 001140 65$:   MOV    #SWREG,SWR   ;;POINT TO SOFTWARE SWR
(2) 001756   012737 000174 001142  MOV    #DISPREG,DISPLAY
(2) 001764   012637 000004 66$:   MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)          CLR    $PASS      ;;CLEAR PASS COUNT
(2) 001770   005037 001202         BITB  #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 001774   132737 000200 001215  BEQ    67$          ;;YES,USE NON-APT SWITCH
(2) 002002   001403         MOV    # $SWREG,SWR ;;NO,USE APT SWITCH REGISTER
(2) 002004   012737 001216 001140 67$:
(2) 002012
515 002012   012700 001520   START1: MOV    #DRVWCR,R0   ;SET UP RFG ADRS POINTERS
516 002016   013701 001250   MOV    $BASE,R1      ;GET BASE ADRS
517 002022   010120         SETUP2: MOV   R1,(R0)+  ;LOAD EM
518 002024   062701 000002   ADD    #2,R1         ;
519 002030   022700 001530   CMP    #DRVDBR+2,R0 ;ALL DONE?
520 002034   001372         BNE    SETUP2       ;BR IF NOT
521 002036   012700 001530   MOV    #DRVCT0,R0   ;SET UP DRV11B VECTOR ADRS POINTER
522 002042   013701 001244   MOV    $VECT1,R1    ;GET BASE VECTOR ADRS
523 002046   042701 170000   BIC    #170000,R1   ;CLR OUT PRIORITY BITS
524 002052   010120         SETUP3: MOV   R1,(R0)+  ;
525 002054   062701 000002   ADD    #2,R1        ;POINT TO NEXT
526 002060   022700 001534   CMP    #DRVCT2+2,R0 ;ALL DONE?

```

```

527 002064 001372          BNE      SETUP3          ;BR IF NOT
528          .SBTTL    TYPE PROGRAM NAME
(1)          ;;TYPE  THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002066 005227 177777  INC      #-1          ;;FIRST TIME?
(1) 002072 001052          BNE      64$          ;;BRANCH IF NO
(1) 002074 104401 002142  TYPE     ,65$        ;;TYPE ASCIZ STRING
(2)          .SBTTL    GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002100 005737 000042  TST     @#42         ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002104 001012          BNE      66$          ;;BRANCH IF YES
(2) 002106 123727 001214 000001  CMPB    $ENV,#1      ;;ARE WE RUNNING UNDER APT?
(2) 002114 001406          BEQ     66$          ;;BRANCH IF YES
(2) 002116 023727 001140 000176  CMP     SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
(2) 002124 001005          BNE      67$          ;;BRANCH IF NO
(2) 002126 104406          GTSWR                    ;;GET SOFT-SWR SETTINGS
(2) 002130 000403          BR      67$
(2) 002132 112737 000001 001134 66$:    MOVB    #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
(2) 002140          67$:
(1) 002140 000427          BR      64$          ;;GET OVER THE ASCIZ
(1)          ;;65$:  .ASCIZ  <CRLF>#MD-11-CVDRA-B DRV11B DMA INTERFACE DIAG #<CRLF>
(1)          64$:
529 002220 005227 177777  INC      #-1          ;FIRST PASS?
530 002224 001022          BNE      100$         ;BRANCH IF NO
531 002226 122737 000001 001134  CMPB    #1,$AUTOB   ;MANUAL INTERVENTION PERMITTED?
532 002234 001416          BEQ     100$         ;BR IF NO
533 002236 104401 015254  TYPE     ,KDF11B    ;ASK ABOUT KDF11-B
534 002242 104410          RDCHR                    ;READ ANSWER
535 002244 012637 001546  MOV     (SP)+,KDF    ;STORE ANSWER
536 002250 104401 001546  TYPE     ,KDF        ;ECHO ANSWER
537 002254 123727 001546 000131  CMPB    KDF,#131    ;IS IT KDF11-B?
538 002262 001003          BNE      100$         ;BRANCH IF NOT
539 002264 152777 000200 176646  BISB    #BIT07,@SWR ;IF YES,SET SWREG
540 002272 005737 001540 100$:   TST     CORSZ       ;TEST IF FIRST PASS
541 002276 001002          BNE      CORSZR      ;BR IF NOT
542 002300 104401 014603  TYPE     ,WARN       ;TELL THE OPERATOR TO TURN OFF DMA REFRESH
543          ;;*****
544          ;;LET'S SEE HOW MUCH MEM WE HAVE
545          ;;*****
546 002304 012700 020000  CORZR:  MOV     #20000,R0 ;USE R0 TO LOOK
547 002310 012737 002322 000004  MOV     #2$,@#ERRVEC ;SET UP TIME OUT RETURN ADRS
548 002316 005720          1$:    TST     (R0)+    ;TAKE A LOOK
549 002320 000776          BR      1$          ;UNTIL TIMEOUT
550 002322 042700 017777  2$:    BIC     #17777,R0  ;POINT TO 1ST NON-EXSISTANT 4K BLK
551 002326 010037 001540  MOV     R0,CORSZ    ;SAVE FOR LATER
552 002332 012737 000006 000004  MOV     #ERRVEC+2,@#ERRVEC ;RESTORE VECTOR
553 002340 012737 015400 001542  MOV     #DBUF,DBUFP ;INITIALIZE TO LOWEST 4K
554 002346 012737 000000 001206  MOV     #0,$UNIT    ;SET UP UNIT COUNT
555 002354 013737 001252 001536  MOV     $DEV,M,DMAP ;GET THE # & POSITION OF DRV11B'S
556 002362 042737 177400 001536  BIC     #177400,DMAP ;UP TO 8 ONLY
557 002370 001406          BEQ     RESTRT      ;GO CONTINUE AS IF SOMETHING WAS SELECTED
558 002372 032737 000001 001536  BIT     #1,DMAP     ;IS 1ST DRV11B SELECTED?
559 002400 001002          BNE      RESTRT      ;BR IF SO
560 002402 000137 007674          JMP     NXDEV1      ;NO - GO ADVANCE BASE DRV11B ADDRESSES
561 002406 106427 000200          RESTRT: MTPS    #200 ;SET PRIORITY TO HIGHEST LEVEL
562 002412 012706 001100          MOV     #STACK,SP  ;ALWAYS RESET STACK PTR
563 002416 013737 001206 001204  MOV     $UNIT,$DEVCT ;LOAD APT COUNTER
564 002424 013700 001206          MOV     $UNIT,R0   ;MAKE AN INDEX
    
```

```

565 002430 006300          ASL      R0          ; VALUE
566 002432 013760 001520 001260  MOV     DRVWCR,$DDWO(R0) ;SAVE THE BUS ADDRESS
567 002440 000005          RESET    ;INITIALIZE DRV11B BEFORE TESTING
568                                     ;*****
(3)                                     ;*TEST 1      TEST THAT ALL DRV11B REGS ARE ACCESSIBLE
(3)                                     ;*****
(2) 002442 000240          TST1:   <NOP>
(1) 002444 012737 002460 001106  MOV     #10,$SLPADR    ;;SET SCOPE LOOP ADDRESS
(2) 002452 012737 000001 001200  MOV     #1,$TESTN     ;;SET TEST NUMBER 'N APT MAIL BOX
569 002460 112737 000001 001102 10$:   MOV     #1,$STSTM     ;SET TO TEST #1
570 002466 012737 002522 001110  MOV     #1,$SLPERR    ;SET UP SCOPE LOOP ADRS
571 002474 005037 001124          CLR     $GDDAT       ;NO DATA COMPARE
572 002500 005037 001126          CLR     $BDDAT       ;NO DATA COMPARE
573 002504 012737 002540 000004  MOV     #2,$@#ERRVEC  ;SET UP TIMEOUT RETURN ADRS
574 002512 013700 001520          MOV     DRVWCR,R0    ;SET UP 1ST DRV11 BUS ADRS
575 002516 012701 000004          MOV     #4,R1        ;SET UP REG COUNT
576 002522 010037 001122 1$:   MOV     R0,$BDADR    ;SET UP CURRENT DRV BUS ADRS
577 002526 005710          TST    (R0)         ;SEE IF THERE
578 002530 005720          TST    (R0)+        ;BUMP TO NEXT
579 002532 005301          DEC    R1           ;COUNT 4 OF THEM
580 002534 001403          BEQ    3$           ;BR IF ALL DONE
581 002536 000771          BR     1$           ;TRY NEXT
582 002540 022626 2$:   CMP     (SP)+,(SP)+  ;FIX STACK SINCE NO RTI
583 002542 104001          ERROR  1           ;BUS ADRS INDICATED DID NOT RESPOND
584 002544 012737 000006 000004 3$:   MOV     #ERRVEC+2,@#ERRVEC ;RESTORE LOC 4
585
586                                     ;*****
(3)                                     ;*TEST 2      TEST THAT THE WORD COUNT REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
(3)                                     ;*****
(2) 002552 000004          TST2:   SCOPE
587 002554 012737 002600 001110  MOV     #1,$SLPERR    ;SET UP SCOPE LOOP ADRS
588 002562 013737 001520 001122  MOV     DRVWCR,$BDADR ;SET UP WC REG ADRS
589 002570 005000          CLR    R0           ;R0 SAYS SHIFT PTRN WHEN 0
590 002572 012737 177776 001124  MOV     #-2,$GDDAT   ;FLOAT 0 RIGHT TO LEFT
591 002600 013777 001124 176712 1$:   MOV     $GDDAT,@DRVWCR ;LD WC
592 002606 017737 176706 001126  MOV     @DRVWCR,$BDDAT ;READ IT BACK
593 002614 023737 001124 001126  CMP     $GDDAT,$BDDAT ;CORRECT?
594 002622 001401          BEQ    2$           ;BR IF SO
595 002624 104002          ERROR  2           ;WORD COUNT WRITE/READ FAILURE
596 002626 005137 001124 2$:   COM    $GDDAT       ;COMPELEMENT ZERO
597 002632 005100          COM    R0           ;R0 SAYS SHIFT LEFT WHEN = 0
598 002634 001361          BNE    1$           ;TRY THE COMPLEMENT IF R0 NOT 0
599 002636 006337 001124          ASL    $GDDAT       ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
600 002642 005237 001124          INC    $GDDAT       ;KEEP LSB SET
601 002646 103754          BCS    1$           ;AGAIN TILL ALL PATRNS DONE
602
603                                     ;*****
(3)                                     ;*TEST 3      TEST THAT THE BUFFER ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
(3)                                     ;*****
(2) 002650 000004          TST3:   SCOPE
604 002652 012737 002676 001110  MOV     #1,$SLPERR    ;SET UP SCOPE LOOP ADRS
605 002660 013737 001522 001122  MOV     DRVBAR,$BDADR ;SET UP BA REG ADRS
606 002666 005000          CLR    R0           ;R0 SAYS SHIFT PTRN WHEN 0
607 002670 012737 177774 001124  MOV     #-4,$GDDAT   ;FLOAT 0 RIGHT TO LEFT
608 002676 013777 001124 176616 1$:   MOV     $GDDAT,@DRVBAR ;LD BA
609 002704 017737 176612 001126  MOV     @DRVBAR,$BDDAT ;READ IT BACK

```

```

610 002712 042737 000001 001126      BIC    #BIT00,$BDDAT    ;DON'T WANT BIT00
611 002720 023737 001124 001126      CMP    $GDDAT,$BDDAT    ;CORRECT?
612 002726 001401                      BEQ    2$                ;BR IF SO
613 002730 104002                      ERROR  2                ;BUS ADRS WRITE/READ FAILURE
614 002732 005137 001124 001124 2$:  COM    $GDDAT          ;COMPLEMENT ZERO
615 002736 042737 000001 001124      BIC    #BIT00,$GDDAT    ;BIT 00 NOT INVOLVED
616 002744 005100                      COM    R0                ;R0 SAYS SHIFT LEFT WHEN = 0
617 002746 001353                      BNE    1$                ;TRY THE COMPLEMENT IF R0 NOT 0
618 002750 006337 001124                      ASL    $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
619 002754 103004                      BCC    TST4             ;:NEXT TEST IF BIT 15 DONE
620 002756 062737 000002 001124      ADD    #2,$GDDAT        ;KEEP ADDR LSB SET
621 002764 000744                      BR     1$                ;AGAIN TILL ALL PATTERNS DONE

```

\*\*\*\*\*  
 :\*TEST 4 TEST THAT THE DATA BUFFER REG IS WRITE/READABLE (FLOAT 0 COM PTRN)  
 \*\*\*\*\*

```

(3)
(3)
(2) 002766 000004
624 002770 012737 003014 001110      TST4: SCOPE
625 002776 013737 001526 001122      MOV    #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
626 003004 005000                      MOV    DRVDBR,$BDADR   ;SET UP DB REG ADRS
627 003006 012737 177776 001124      CLR    R0               ;R0 SAYS SHIFT PTRN WHEN 0
628 003014 013777 001124 176504 1$:  MOV    #-2,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
629 003022 017737 176500 001126      MOV    $GDDAT,@DRVDBR ;LD DB
630 003030 023737 001124 001126      MOV    @DRVDBR,$BDDAT ;READ IT BACK
631 003036 001401                      CMP    $GDDAT,$BDDAT   ;CORRECT?
632 003040 104002                      BEQ    2$                ;BR IF SO
633 003042 005137 001124 001124 2$:  ERROR  2                ;DATA BUFFER WRITE/READ FAILURE (LOOP BACK)
634 003046 005100                      COM    $GDDAT          ;COMPLEMENT ZERO
635 003050 001361                      COM    R0                ;R0 SAYS SHIFT LEFT WHEN = 0
636 003052 006337 001124                      BNE    1$                ;TRY THE COMPLEMENT IF R0 NOT 0
637 003056 005237 001124                      ASL    $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
638 003062 103754                      INC    $GDDAT          ;KEEP LSB SET
639                                     BCS    1$                ;AGAIN TILL ALL PATRNS DONE

```

\*\*\*\*\*  
 :\*TEST 5 TEST THAT THE DATA BUFFER REG IS BYTE ADDRESSABLE  
 \*\*\*\*\*

```

(3)
(3)
(2) 003064 000004
641 003066 013700 001526              TST5: SCOPE
642 003072 010037 001122              MOV    DRVDBR,R0       ;GET DB REG ADRS
643 003076 005010                      MOV    R0,$BDADR      ;SET UP DB REG ADRS
644 003100 012737 177177 001124      CLR    (R0)            ;ZERO DATA BUFFER REG
645 003106 012737 077776 001126      MOV    #177177,$GDDAT ;LD EXPECTED
646 003114 153760 001126 000001      MOV    #77776,$BDDAT  ;SEND DATA FROM 'BDDAT'
647 003122 153710 001127              BISB  $BDDAT,1(R0)    ;LOAD HI BYTE DB
648 003126 011037 001126              BISB  $BDDAT+1,(R0)  ;LOAD LO BYTE DB
649 003132 023737 001124 001126      MOV    (R0),$BDDAT    ;READ IT BACK
650 003140 001401                      CMP    $GDDAT,$BDDAT   ;CORRECT?
651 003142 104002                      BEQ    TST6            ;:NEXT TEST IF SO
652                                     ERROR  2                ;DATA ERROR ON BYTE ADDRESSING THE DATA BUFFER REG

```

\*\*\*\*\*  
 :\*TEST 6 TEST THAT RESET CLEARS WORD COUNT, BUS ADDRESS & DATA REGS  
 \*\*\*\*\*

```

(3)
(3)
(2) 003144 000004
(1) 003146 012737 000010 001160      TST6: SCOPE
654 003154 005037 001124              MOV    #10,$TIMES     ;;DO 10 ITERATIONS
655 003160 012777 177777 176332      CLR    $GDDAT         ;LD EXPECTED
                                      MOV    #-1,@DRVWCR    ;SET ALL BITS - WC REG

```

656	003166	012777	177776	176326	MOV	#-2,@DRVBAR	:SET ALL BITS - BUS ADRS REG
657	003174	012777	177777	176324	MOV	#-1,@DRVDBR	:SET ALL BITS - DB OUT REG
658	003202	000005			RESET		:DO A BUS RESET
659	003204	017737	176310	001126	MOV	@DRVWCP,\$BDDAT	:READ WC REG
660	003212	001404			BEQ	1\$	:BR IF CLRED
661	003214	013737	001520	001122	MOV	DRVWCR,\$BDADR	:SET UP WC REG ADRS
662	003222	104003			ERROR	3	:RESET FAILED TO CLR WC REG
663	003224	017737	176272	001126	1\$: MOV	@DRVBAR,\$BDDAT	:READ BUS ADRS REG
664	003232	042737	000001	001126	BIC	#BIT00,\$BDDAT	:DON'T WANT BITT00
665	003240	001404			BEQ	2\$	:BR IF CLRED
666	003242	013737	001522	001122	MOV	DRVBAR,\$BDADR	:SET UP BA REG ADRS
667	003250	104003			ERROR	3	:RESET FAILED TO CLR BUS ADRS REG
668	003252	017737	176250	001126	2\$: MOV	@DRVDBR,\$BDDAT	:READ DATA BUFFER REG
669	003260	001404			BEQ	TST7	:NEXT TEST IF CLRED
670	003262	013737	001526	001122	MOV	DRVDBR,\$BDADR	:SET UP DB REG ADRS
671	003270	104003			ERROR	3	:RESET FAILED TO CLR DATA BUFFER OUT REG

672  
673  
(3)  
(3)  
\*\*\*\*\*  
\*TEST 7 TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT PTRN)  
\*\*\*\*\*

(2)	003272	000004			TST7: SCOPE		
(1)	003274	012737	000010	001160	MOV	#10,\$TIMES	:DO 10 ITERATIONS
674	003302	106427	000200		MTPS	#200	:DON'T WANT ANY INTRS
675	003306	004537	010112		JSR	R5,SETVEC	:SET UP INTR RETURN ADRS IN CASE
676	003312	003410			3\$		:RETURN TO 3\$ ON ILLEGAL INTR
677	003314	013737	003334	001110	MOV	1\$,\$LPERR	:SET UP SCOPE LOOP ADRS
678	003322	013737	001524	001122	MOV	DRVCSR,\$BDADR	:SET UP CSR ADRS
679	003330	012700	160000		MOV	#160000,R0	:START AT 0 - HI BITS FOR NOISE
680	003334	010037	001124		1\$: MOV	R0,\$GDDAT	:LD EXPECTED
681	003340	042737	167201	001124	BIC	#167201,\$GDDAT	:MASK TO WRITEABLE BITS
682	003346	010077	176152		MOV	R0,@DRVCSR	:LD CSR
683	003352	017737	176146	001126	MOV	@DRVCSR,\$BDDAT	:READ IT BACK
684	003360	042737	007200	001126	BIC	#7200,\$BDDAT	:DON'T LOOK AT STAT & RDY BITS
685	003366	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:CORRECT?
686	003374	001401			BEQ	2\$	:BR IF SO
687	003376	104002			ERROR	2	:CONTROL/STATUS REG WRITE/READ FAILURE
688	003400	062700	000002		2\$: ADD	#2,R0	:ADVANCE COUNT PATTERN
689	003404	001353			BNE	1\$	:WRITE NEXT PATTERN IF NOT ALL TESTED
690	003406	000413			BR	4\$	:GO RESTORE VECTOR
691	003410	022626			3\$: CMP	(SP)+,(SP)+	:FIX STACK - SHOULD NOT HAVE INTR'ED
692	003412	052737	000200	001124	BIS	#200,\$GDDAT	:CORRECT EXPECTED
693	003420	017737	176100	001126	MOV	@DRVCSR,\$BDDAT	:READ CSR
694	003426	042737	007000	001126	BIC	#7000,\$BDDAT	:DON'T WANT 'STAT' BITS
695	003434	104005			ERROR	5	:CPU FAILED TO LOCK OUT DRV11B INTR REQ
696	003436	004737	010132		4\$: JSR	PC,RSTVEC	:GO RESTORE VECTOR

697  
698  
(3)  
(3)  
\*\*\*\*\*  
\*TEST 10 TEST THAT RESET CLEARS ALL WRITEABLE BITS & SET READY IN CSR  
\*\*\*\*\*

(2)	003442	000004			TST10: SCOPE		
(1)	003444	012737	000010	001160	MOV	#10,\$TIMES	:DO 10 ITERATIONS
699	003452	013737	001524	001122	MOV	DRVCSR,\$BDADR	:SET UP CSR ADRS
700	003460	012737	000200	001124	MOV	#200,\$GDDAT	:LD EXPECTED
701	003466	012777	177776	176030	MOV	#-2,@DRVCSR	:LD ALL CSR BITS
702	003474	000005			RESET		:DO A BUS RESET
703	003476	017737	176022	001126	MOV	@DRVCSR,\$BDDAT	:READ CSR

```

704 003504 023737 001124 001126
705 003512 001401
706 003514 104003
707
708
(3)
(3)
(2) 003516 000004
709 003520 013700 001524
710 003524 010037 001122
711 003530 005010
712 003532 012737 010300 001124
713 003540 012737 040020 001126
714 003546 153760 001126 000001
715 003554 153710 001127
716 003560 011037 001126
717 003564 023737 001124 001126
718 003572 001401
719 003574 104002
720 003576 005010
721
722
(3)
(3)
(2) 003600 000004
723 003602 012737 003630 001110
724 003610 013737 001524 001122
725 003616 012737 007000 001124
726 003624 012700 000016
727 003630 010077 175670
728 003634 017737 175664 001126
729 003642 042737 170777 001126
730 003650 023737 001124 001126
731 003656 001401
732 003660 104004
733 003662 162737 001000 001124
734 003670 162700 000002
735 003674 100355
736
737
(3)
(3)
(2) 003676 000004
(1) 003700 012737 000010 001160
738 003706 106427 000200
739 003712 000005
740 003714 012777 003774 175606
741 003722 013737 001524 001122
742 003730 012737 000300 001124
743 003736 106427 000000
744 003742 021616
745 003744 012777 004010 175556
746 003752 052777 000100 175544
747 003760 021616
748 003762 017737 175536 001126
749 003770 104005

```

```

CMP $GDDAT,$BDDAT :CORRECT?
BEQ TST11 :NEXT TEST IF CSR CLRED & READY SET
ERROR 3 :RESET FAILED TO SET UP THE CSR

:*****
:*TEST 11 TEST THAT THE CSR IS BYTE ADDRESSABLE
:*****
TST11: SCOPE
MOV DRVCSR,R0 :GET CSR ADRS
MOV R0,$BDADR :SET UP CSR ADRS
CLR (R0) :ZERO CSR
MOV #10300,$GDDAT :LD EXPECTED
MOV #40020,$BDDAT :SEND DATA FROM 'BDDAT' - USE MAIN + IE
BISB $BDDAT,1(R0) :LOAD HI BYTE CSR
BISB $BDDAT+1,(R0) :LOAD LO BYTE CSR
MOV (R0),$BDDAT :READ IT BACK
CMP $GDDAT,$BDDAT :CORRECT?
BEQ 1$ :BR IF SO
ERROR 2 :DATA ERROR ON BYTE ADDRESSING THE CSR
1$ CLR (R0) :ZERO CSR BEFORE ADVANCING

:*****
:*TEST 12 TEST THAT THE 3 'FNCT' BITS CONTROL THE 3 'STAT' BITS (COUNT PTRN)
:*****
TST12: SCOPE
MOV #1$,$LPERR :SET UP SCOPE LOOP ADRS
MOV DRVCSR,$BDADR :SET UP CSR ADRS
MOV #7000,$GDDAT :LD EXPECTED
MOV #16,R0 :R0 CONTAINS 'FNCT' BITS WRITTEN
1$: MOV R0,@DRVCSR :WRITE INTO 'FNCT' BITS
MOV @DRVCSR,$BDDAT :READ BACK THRU 'STAT' BITS
BIC #170777,$BDDAT :MASK TO 'STAT' BITS ONLY
CMP $GDDAT,$BDDAT :CORRECT?
BEQ 2$ :BR IF SO
ERROR 4 :'FNCT' BITS FAILED TO SET 'STAT' BITS (LOOP BACK)
2$: SUB #1000,$GDDAT :CHANGE TO NEXT EXPECTED
SUB #2,R0 :DECREASE COUNT PATTERN
BPL 1$ :DO AGAIN UNTIL 0 TESTED

:*****
:*TEST 13 TEST THAT READY SET WILL CAUSE AN INTERRUPT AT LEVEL 0
:*****
TST13: SCOPE
MOV #10,$TIMES :DO 10 ITERATIONS
MTPS #200 :DONT WANT INTR YET
RESET :SET THE READY FLAG BY INIT
MOV #1$,@DRVCTO :SET UP PREMATURE INTR RETURN ADRS
MOV DRVCSR,$BDADR :SET UP CSR ADRS
MOV #300,$GDDAT :LD EXPECTED (READY + IE)
MTPS #0 :ALLOW AN INTR
CMP (SP),(SP) :STALL
MOV #2$,@DRVCTO :SET UP EXPECTED INTR RETURN ADRS
BIS #BIT6,@DRVCSR :ENABLE THE EXPECTED INTERRUPT
CMP (SP),(SP) :STALL
MOV @DRVCSR,$BDDAT :GET THE CSR
ERROR 5 :READY FAILED TO CAUSE AN INTERRUPT

```



```

750 003772 000417          BR      3$          :GO RESTORE VECTOR
751 003774 022626    1$:  CMP      (SP)+,(SP)+ :SHOULD NEVER GET HERE - IE NOT WORKING?
752 003776 017737    175522 001126  MOV     @DRVCSR,$BDDAT :GET THE CSR
753 004004 104005          ERROR   5          :READY INTERRUPTED WITHOUT THE IE BIT
754 004006 000411          BR      3$          :GO RESTORE VECTOR
755 004010 022626    2$:  CMP      (SP)+,(SP)+ :FIX STACK SINCE NO RETURN
756 004012 017737    175506 001126  MOV     @DRVCSR,$BDDAT :READ STATUS
757 004020 023737    001124 001126  CMP     $GDDAT,$BDDAT :CORRECT?
758 004026 001401          BEQ     3$          :BR IF SO
759 004030 104005          ERROR   5          :INCORRECT STATUS ON READY INTR
760 004032 004737    010132 3$:  JSR     PC,RSTVEC   :GO RESTORE VECTOR
  
```

```

761
762  (3)  :*****
763  (3)  :*TEST 14 TEST THAT GO CLRS READY & FNCT 2 WILL SET IT
764  (2)  :*****
765  (2)  TST14: SCOPE
766  004036 000004          MTPS    #200          :DONT WANT ANY INTRs
767  004040 106427    000200  MOV     DRVCSR,$BDADR  :SET UP CSR ADRS
768  004044 013737    001524 001122  CLR     $GDDAT         :EXPECT 0
769  004052 005037    001124  MOV     #1,@DRVCSR    :SET GO WHICH SHOULD CLR READY
770  004056 012777    000001 175440  MOV     @DRVCSR,$BDDAT :READ THE CSR
771  004064 017737    175434 001126  CMP     $GDDAT,$BDDAT :CORRECT?
772  004072 023737    001124 001126  BEQ     1$            :BR IF SO
773  004100 001401          ERROR   6          :THE GO BIT FAILED TO CLR READY
774  004102 104006    1$:  BJS     #4,@DRVCSR    :FNCT 2 SHOULD SET READY
775  004104 052777    000004 175412  MOV     #2204,$GDDAT  :LD EXPECTED
776  004112 012737    002204 001124  MOV     @DRVCSR,$BDDAT :GET CSR
777  004120 017737    175400 001126  CMP     $GDDAT,$BDDAT :CORRECT?
778  004126 023737    001124 001126  BEQ     TST15         :NEXT TEST IF SET
779  004134 001401          ERROR   6          :FNCT 2 (VIA ATTN) FAILED TO SET READY
780  004136 104006
  
```

```

781  (3)  :*****
782  (3)  :*TEST 15 TEST THAT READY CONTROLS 'BAR' BIT00
783  (2)  :*****
784  (2)  TST15: SCOPE
785  004140 000004          MOV     #4,@DRVCSR    :SET READY
786  004142 012777    000004 175354  MOV     DRVBAR,$BDADR :SET UP BAR ADRS
787  004150 013737    001522 001122  MOV     #1,$GDDAT     :EXPECT LSB OF BAR
788  004156 012737    000001 001124  CLR     @DRVBAR        :CLR BAR
789  004164 005077    175332  MOV     @DRVBAR,$BDDAT :READ BAR
790  004170 017737    175326 001126  CMP     $GDDAT,$BDDAT :CORRECT?
791  004176 023737    001124 001126  BEQ     1$            :BR IF SO
792  004204 001401          ERROR   2          :A00 FAILED TO READ A ONE(SB TIED TO RDY)
793  004206 104002    1$:  MOV     #1,@DRVCSR    :SET GO(CLRs BAR BIT00)
794  004210 012777    000001 175306  MOV     @DRVBAR,$BDDAT :READ BAR
795  004216 017737    175300 001126  BEQ     2$            :BR IF ZERO
796  004224 001403          CLR     $GDDAT       :EXPECTED ZERO
797  004226 005037    001124  ERROR   2          :WHEN RDY CLRED-A00 FAILED TO READ A ZERO
798  004232 104002    2$:  MOV     #4,@DRVCSR    :INSURE RDY SET BEFORE ADVANCING
799  004234 012777    000004 175262
  
```

```

800  (3)  :*****
801  (3)  :*TEST 16 TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)
802  (2)  :*****
803  (2)  TST16: SCOPE
804  004242 000004          MOV     DRVDDBR,$BDADR :SET UP DBR ADRS
805  004244 013737    001526 001122  MOV     #125252,$GDDAT :LD EXPECTED
806  004252 012737    125252 001124
  
```

```

797 004260 013777 001124 175240      MOV      $GDDAT,@DRVDBR      ;LD DBR WITH #125252
798 004266 012777 000400 175230      MOV      #400,@DRVCSR       ;SET CYCLE - SHOULD CLK DBR (IN)
799 004274 012777 052525 175224      MOV      #52525,@DRVDBR     ;CHANGE DBR (OUT) DATA - SHOULD NOT AFFECT (IN)
800 004302 017737 175220 001126      MOV      @DRVDBR,$BDDAT     ;READ DBR
801 004310 023737 001124 001126      CMP      $GDDAT,$BDDAT     ;CORRECT?
802 004316 001401                BEQ      1$                 ;BR IF SO
803 004320 104017                ERROR   17                 ;CYCLE DID NOT LATCH DBR (IN) DATA
804 004322 042777 000400 175174 1$:   BIC      #400,@DRVCSR       ;REMOVE CYCLE
805 004330 012737 052525 001124      MOV      #52525,$GDDAT     ;NOW EXPECT #52525
806 004336 017737 175164 001126      MOV      @DRVDBR,$BDDAT     ;READ DBR
807 004344 023737 001124 001126      CMP      $GDDAT,$BDDAT     ;CORRECT?
808 004352 001401                BEQ      TST17              ;NEXT TEST IF SO
809 004354 104002                ERROR   2                 ;DBR FAILED TO READ WHEN CYCLE CLRED (NORMAL)
810
811
(3)
(3)
(2) 004356 000004                TST17: SCOPE
812 004360 012737 004402 001110      MOV      #1$,$LPERR        ;SET UP SCOPE LOOP ADRS
813 004366 004537 010112                JSR      R5,SETVEC         ;GO SET UP INTERRUPT RETURN
814 004372 004476                2$
815 004374 012700 177776                MOV      #-2,R0            ;FLOAT ZERO RIGHT TO LEFT
816 004400 005001                CLR      R1                 ;R1 CONTROLS DATA SHIFTING
817 004402 012777 177777 175110 1$:   MOV      #-1,@DRVWCR        ;DO ONE XFER
818 004410 012777 015400 175104      MOV      #DBUF,@DRVBAR     ;GET DATA WORD FROM 'DBUF'
819 004416 010037 015400                MOV      R0,DBUF           ;SET UP MEM DATA
820 004422 012777 000101 175074      MOV      #101,@DRVCSR      ;SET IE & GO
821 004430 052777 000400 175066      BIS      #400,@DRVCSR      ;SET CYCLE
822 004436 106427 000000                MTPS    #0                 ;ENABLE THE INTR
823 004442 013737 001524 001122      MOV      DRVCSR,$BDDADR    ;SET UP CSR ADRS
824 004450 012737 000700 001124      MOV      #700,$GDDAT      ;LD EXPECTED
825 004456 017737 175042 001126      MOV      @DRVCSR,$BDDAT    ;READ THE CSR
826 004464 042777 000100 175032      BIC      #100,@DRVCSR      ;CLR IE
827 004472 104005                ERROR   5                 ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
828 004474 000442                BR       4$                 ;GO RESTORE VECTOR
829 004476 022626                2$:   CMP      (SP)+,(SP)+       ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
830 004500 004537 010156                JSR      R5,CKSTAT         ;GO CHECK STATUS
831 004504 000700                700
832 004506 000001                1
833 004510 104007                ERROR   7                 ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
834 004512 000433                BR       4$                 ;GO RESTORE VECTOR
835 004514 104010                ERROR   10                ;RETURN HERE IF WC ER - EXPECTED 0
836 004516 000431                BR       4$                 ;GO RESTORE VECTOR
837 004520 104011                ERROR   11                ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
838 004522 000427                BR       4$                 ;GO RESTORE VECTOR
839 004524 012737 015400 001120      MOV      #DBUF,$GDADR      ;RETURN HERE IF OK - SET UP XFER ADRS
840 004532 010037 001124                MOV      R0,$GDDAT        ;LD EXPECTED
841 004536 017737 174764 001126      MOV      @DRVDBR,$BDDAT    ;READ DATA XFERED
842 004544 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;CORRECT?
843 004552 001405                BEQ      3$                 ;BR IF SO
844 004554 013737 001526 001122      MOV      DRVDBR,$BDDADR    ;SET UP DBR ADRS
845 004562 104012                ERROR   12                ;DATA ER - DBR CONTAINS WRONG DATA
846 004564 000406                BR       4$                 ;GO RESTORE VECTOR
847 004566 005100                3$:   COM      R0                 ;RETURN HERE ON GOOD DATA - NOW COM PATRN
848 004570 005101                COM      R1                 ;KEEP TRACK OF COMPLEMENT
849 004572 001303                BNE     1$                 ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0

```

```

850 004574 006300 ASL R0 ;WAS DONE - NOW SHIFT ZERO LEFT
851 004576 005200 INC R0 ;KEEP LSB SET
852 004600 103700 BCS 1$ ;AGAIN TILL ZERO BIT IN CARRY
853 004602 004737 010132 4$: JSR PC,RSTVEC ;GO RESTORE VECTOR
854
855
(3) ::*****
(3) :*TEST 20 TEST SINGLE 'DATO' NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
(2) TST20: SCOPE
856 004606 000004 MOV #1,$LPERR ;SET UP SCOPE LOOP ADRS
857 004610 012737 004632 001110 JSR R5,SETVEC ;GO SET UP INTERRUPT RETURN
858 004616 004537 010112 2$ ;RETURN TO 2$ ON INTR
859 004622 004720 ;FLOAT ZERO RIGHT TO LEFT
860 004624 012700 177776 MOV #-2,R0 ;R1 CONTROLS DATA SHIFTING
861 004630 005001 CLR R1 ;DO ONE XFER
862 004632 012777 177777 174660 1$: MOV #-1,@DRVWCR ;WRITE DATA WORD TO 'DBUF'
863 004640 012777 015400 174654 MOV #DBUF,@DRVBAR ;SET UP DATA IN DBR
864 004646 010077 174654 MOV R0,@DRVDBR ;SET IE, GO & FNCT1 (C1 CONTROL)
865 004652 012777 000103 174644 MOV #103,@DRVCSR ;SET CYCLE
866 004660 052777 000400 174636 BIS #400,@DRVCSR ;ENABLE THE INTR
867 004666 106427 000000 MTPS #0 ;SET UP CSR ADRS
868 004672 013737 001524 001122 MOV DRVCSR,$BADDR ;LD EXPECTED
869 004700 012737 001702 001124 MOV #1702,$GDDAT ;READ THE CSR
870 004706 017737 174612 001126 MOV @DRVCSR,$BDDAT ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
871 004714 104005 ERROR 5 ;GO RESTORE VECTOR
872 004716 000442 BR 4$ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
873 004720 022626 010156 2$: CMP (SP)+,(SP)+ ;GO CHECK STATUS
874 004722 004537 JSR R5,CKSTAT ;CSR STATUS EXPECTED
875 004726 001702 1702 ;# OF XFERS
876 004730 000001 1 ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
877 004732 104007 ERROR 7 ;CYCLE, READY, IE & FNCT 1
878 004734 000433 BR 4$ ;GO RESTORE VECTOR
879 004736 104010 ERROR 10 ;RETURN HERE IF WC ER - EXPECTED 0
880 004740 000431 BR 4$ ;GO RESTORE VECTOR
881 004742 104011 ERROR 11 ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
882 004744 000427 BR 4$ ;GO RESTORE VECTOR
883 004746 012737 015400 001120 MOV #DBUF,$GDADR ;RETURN HERE IF OK - SET UP XFER ADRS
884 004754 010037 001124 MOV R0,$GDDAT ;LD EXPECTED
885 004760 013737 015400 001126 MOV DBUF,$BDDAT ;GET DATA XFERED
886 004766 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
887 004774 001405 BEQ 3$ ;BR IF SO
888 004776 013737 001526 001122 MOV DRVDBR,$BADDR ;SET UP DBR ADRS
889 005004 104013 ERROR 13 ;DATA ER - MEM CONTAINS WRONG DATA
890 005006 000406 BR 4$ ;GO RESTORE VECTOR
891 005010 005100 3$: COM R0 ;RETURN HERE ON GOOD DATA - NOW COM PATRN
892 005012 005101 COM R1 ;KEEP TRACK OF COMPLEMENT
893 005014 001306 BNE 1$ ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
894 005016 006300 ASL R0 ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
895 005020 005200 INC R0 ;KEEP LSB SET
896 005022 103703 BCS 1$ ;AGAIN TILL ZERO BIT IN CARRY
897 005024 004737 010132 4$: JSR PC,RSTVEC ;GO RESTORE VECTOR
898
899
(3) ::*****
(3) :*TEST 21 TEST 200 'DATI' NPR TRANSFERS (BURST MODE)
(2) TST21: SCOPE
005030 000004
    
```

```

(1) 005032 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
900 005040 004537 010112 JSR R5,SETVEC ;GO SET UP INTERRUPT RETURN
901 005044 005142 1$ ;RETURN TO 1$ ON INTR
902 005046 004737 010336 JSR PC,LDBUF ;GO LOAD BUFFER WITH COMPLEMENTING PATRN
903 005052 012777 177470 174440 MOV #-200,@DRVWCR ;LOAD WC REG - WILL DO 200 XFERS
904 005060 012777 015400 174434 MOV #DBUF,@DRVBAR ;SET UP CURRENT ADPS
905 005066 106427 000000 MTPS #0 ;ENABLE THE INTR
906 005072 012777 000101 174424 MOV #101,@DRVCSR ;SET IE & GO
907 005100 052777 000400 174416 BIS #400,@DRVCSR ;SET CYCLE
908 005106 013737 001524 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADPS
909 005114 012737 000700 001124 MOV #700,$GDDAT ;LD EXPECTED
910 005122 017737 174376 001126 MOV @DRVCSR,$BDDAT ;READ THE CSR
911 005130 042777 000100 174366 BIC #100,@DRVCSR ;CLR INTR ENABLE
912 005136 104005 ERROR 5 ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
913 005140 000434 BR 2$ ;GO RESTORE VECTOR
914 005142 022626 1$: CMP (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
915 005144 004537 010156 JSR R5,CKSTAT ;GO CHECK STATUS
916 005150 000700 700 ;CSR STATUS EXPECTED
917 005152 000310 200. ;# OF XFERS
918 005154 104007 ERROR 7 ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
919 005156 000425 BR 2$ ;GO RESTORE VECTOR
920 005160 104010 ERROR 10 ;RETURN HERE IF WC ER - EXPECTED 0
921 005162 000423 BR 2$ ;GO RESTORE VECTOR
922 005164 104011 ERROR 11 ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
923 005166 000421 BR 2$ ;GO RESTORE VECTOR
924 005170 012737 016216 001120 MOV #DBUF+616,$GDADR ;OK - SET UP LAST XFER ADPS WHERE #70707 SHOULD BE
925 005176 012737 070707 001124 MOV #70707,$GDDAT ;LD EXPECTED
926 005204 017737 174316 001126 MOV @DRVDBR,$BDDAT ;DBR SHOULD HAVE LAST DATUM
927 005212 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
928 005220 001404 BEQ 2$ ;BR IF SO
929 005222 013737 001526 001122 MOV DRVDBR,$BDADR ;SET UP DBR ADPS
930 005230 104012 ERROR 12 ;DATA ER - DBR DID NOT CONTAIN EXPECTED LAST XFER
931 005232 004737 010132 2$: JSR PC,RSTVEC ;GO RESTORE VECTOR
    
```

```

*****
;*TEST 22 TEST 200 'DATO' NPR TRANSFERS (BURST MODE)
*****
    
```

```

(2) 005236 000004 TST22: SCOPE
(1) 005240 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
934 005246 004537 010112 JSR R5,SETVEC ;GO SET UP INTERRUPT RETURN
935 005252 005352 1$ ;RETURN TO 1$ ON INTR
936 005254 012777 177470 174236 MOV #-200,@DRVWCR ;WORD WC REG - WILL DO 200 XFER'S
937 005262 012777 015400 174232 MOV #DBUF,@DRVBAR ;SET UP CURRENT ADPS
938 005270 012777 177377 174230 MOV #177377,@DRVDBR ;THIS WILL BE WRITTEN TO MEM
939 005276 106427 000000 MTPS #0 ;ENABLE THE INTR
940 005302 012777 000103 174214 MOV #103,@DRVCSR ;SET IE, FNCT 1 & GO
941 005310 052777 000400 174206 BIS #400,@DRVCSR ;SET CYCLE
942 005316 013737 001524 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADPS
943 005324 012737 001702 001124 MOV #1702,$GDDAT ;LD EXPECTED
944 005332 017737 174166 001126 MOV @DRVCSR,$BDDAT ;READ THE CSR
945 005340 042777 000100 174156 BIC #100,@DRVCSR ;CLR INTR ENABLE
946 005346 104005 ERROR 5 ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
947 005350 000416 BR 2$ ;GO RESTORE VECTOR
948 005352 022626 1$: CMP (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
949 005354 004537 010156 JSR R5,CKSTAT ;GO CHECK STATUS
950 005360 001702 1702 ;CSR STATUS EXPECTED
    
```

```

951 005362 000310 200. ;# OF XFERS
952 005364 104007 ERROR 7 ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
953 ;CYCLE, READY, IE & FNCT 1
954 005366 000407 BR 2$ ;GO RESTORE VECTOR
955 005370 104010 ERROR 10 ;RETURN HERE IF WC ER - EXPECTED 0
956 005372 000405 BR 2$ ;GO RESTORE VECTOR
957 005374 104011 ERROR 11 ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
958 005376 000403 BR 2$ ;GO RESTORE VECTOR
959 005400 004737 010460 JSR PC,CKDAT ;RETURN HERE IF OK - NOW GO CHECK DATA
960 005404 104013 ERROR 13 ;RETURN HERE IF DATA ER - DBR CONTAINS WRONG DATA
961 005406 004737 010132 2$: JSR PC,RSTVEC ;RETURN HERE IF DATA CHECK OK - GO RESTORE VECTOR
  
```

```

962
963 ;*****
(3) ;*TEST 23 TEST THAT THE CPU IS LOCKED OUT WITH SINGLE CYCLE OFF
(3) ;*****
  
```

```

(2) 005412 000004 TST23: SCOPE
(1) 005414 012737 000010 001160 MOV #10,$TIMES ;:DO 10 ITERATIONS
964 005422 013737 001524 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
965 005430 004537 010112 JSR R5,SETVEC ;GO SET UP INTR RETURN
966 005434 005544 3$ ;RETURN TO 3$ ON INTR
967 005436 012700 000010 MOV #10,R0 ;DO EIGHT 200 WORD XFER'S
968 005442 005037 001126 CLR $BDDAT ;USR $BDDAT AS A COUNTER
969 005446 012777 177470 174044 1$: MOV #-200.,@DRVWCR ;DO 200 XFERS (DATI'S)
970 005454 012777 015400 174040 MOV #DBUF,@DRVBAR ;FROM DBUF
971 005462 106427 000000 MTPS #0 ;ALLOW AN INTR
972 005466 012777 000101 174030 MOV #101,@DRVCSR ;SET IE & GO
973 005474 052777 000400 174022 BIS #400,@DRVCSR ;SET CYCLE
974 005502 000240 NOP ;FREEBEE
975 005504 000240 NOP
976 005506 000240 NOP
977 005510 005237 001126 2$: INC $BDDAT ;START COUNTING - SHOULD NEVER GET HERE
978 005514 001375 BNE 2$ ;UNTIL 64K
979 005516 012737 000700 001124 MOV #700,$GDDAT ;LD EXPECTED
980 005524 017737 173774 001126 MOV @DRVCSR,$BDDAT ;READ STATUS
981 005532 042777 000100 173764 BIC #100,@DRVCSR ;CLR IE
982 005540 104005 ERROR 5 ;NO INTERRUPT ON 200 DATI'S
983 005542 000407 BR 4$ ;GO RESTORE VECTOR
984 005544 022626 3$: CMP (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
985 005546 005300 DEC R0 ;DONE 8 TIMES"
986 005550 001336 BNE 1$ ;BR IF NOT
987 005552 005737 001126 TST $BDDAT ;SHOULD STILL BE ZERO
988 005556 001401 BEQ 4$ ;BR IF SO
989 005560 104014 ERROR 14 ;BURST MD (SINGLE CYCLE=0) FAILS TO LOCK OUT CPU
990 005562 004737 010132 4$: JSR PC,RSTVEC ;GO RESTORE VECTOR
  
```

```

991
992 ;*****
(3) ;*TEST 24 TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON
(3) ;*****
  
```

```

(2) 005566 000004 TST24: SCOPE
(1) 005570 012737 000010 001160 MOV #10,$TIMES ;:DO 10 ITERATIONS
993 005576 013737 001524 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
994 005604 004537 010112 JSR R5,SETVEC ;GO SET UP INTR RETURN
995 005610 005716 3$ ;RETURN TO 3$ ON INTR
996 005612 012700 000010 MOV #10,R0 ;DO EIGHT 200 WORD XFER'S
997 005616 012737 000000 001126 MOV #0,$BDDAT ;USE $BDDAT AS A COUNTER
998 005624 012777 177470 173666 1$: MOV #-200.,@DRVWCR ;DO 200 XFERS (DATI'S)
  
```

```

999 005632 012777 015400 173662 MOV #DBUF,@DRVBAR ;FROM DBUF
1000 005640 106427 000000 MTPS #0 ;ALLOW AN INTR
1001 005644 012777 000111 173652 MOV #111,@DRVCSR ;SET IE, FNCT3 & GO
1002 005652 052777 000400 173644 BIS #400,@DRVCSR ;SET CYCLE
1003 005660 000240 NOP ;FREEBEE
1004 005662 005237 001126 2$: INC $BDDAT ;START COUNTING
1005 005666 001375 BNE 2$ ;UNTIL 64K - SHOULD INTR BEFORE OVERFLOW
1006 005670 012737 004710 001124 MOV #4710,$GDDAT ;LD EXPECTED
1007 005676 017737 173622 001126 MOV @DRVCSR,$BDDAT ;READ STATUS
1008 005704 042777 000100 173612 BIC #100,@DRVCSR ;CLR IE
1009 005712 104005 ERROR 5 ;NO INTERRUPT ON 200 DATI'S (WITH SINGLE CYCLE)
1010 005714 000423 BR 5$ ;GO RESTORE VECTOR
1011 005716 022626 3$: CMP (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
1012 005720 005300 DEC R0 ;DONE 8 TIMES?
1013 005722 001340 BNE 1$ ;BR IF NOT
1014 005724 022737 000000 001126 CMP #0,$BDDAT ;$BDDAT SHOULD HAVE BEEN COUNTED
1015 005732 103401 BCS 4$ ;BR IF SO
1016 005734 104015 ERROR 15 ;CPU APPEARED LOCKED OUT WITH SINGLE CYCLE SET
1017 005736 017737 173562 001126 4$: MOV @DRVCSR,$BDDAT ;READ STATUS
1018 005744 012737 004710 001124 MOV #4710,$GDDAT ;LD EXPECTED
1019 005752 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1020 005760 001401 BEQ 5$ ;BR IF SO
1021 005762 104007 ERROR 7 ;STATUS INCORRECT ON XFER WITH SINGLE CYCLE SET
1022 005764 004737 010132 5$: JSR PC,RSTVEC ;GO RESTORE VECTOR
1023
1024
(3)
(3)
(2) 005770 000004
(1) 005772 012737 000200 001160
1025 006000 004537 010112
1026 006004 006122
1027 006006 004737 010410
1028 006012 012737 011702 006130
1029 006020 012737 000001 006132
1030 006026 106427 000000 1$: MTPS #0 ;ALLOW INTR
1031 006032 012777 015400 173462 MOV #DBUF,@DRVBAR ;SET UP CURRENT ADRS
1032 006040 013777 006132 173452 MOV 4$,@DRVWCR ;GET XFER #
1033 006046 005477 173446 NEG @DRVWCR ;NEGATE FOR WC
1034 006052 012777 010101 173444 MOV #10101,@DRVCSR ;SET UP MAINT, IE & GO
1035 006060 052777 000400 173436 BIS #400,@DRVCSR ;SET CYCLE
1036 006066 013737 001524 001122 MOV DRVCSR,$BDDADR ;SET UP CSR ADRS
1037 006074 013737 006130 001124 MOV 3$,$GDDAT ;LD EXPECTED
1038 006102 017737 173416 001126 MOV @DRVCSR,$BDDAT ;READ STATUS
1039 006110 042777 000100 173406 BIC #100,@DRVCSR ;DISABLE IE
1040 006116 104005 ERROR 5 ;NO INTR ON XFER (IN MAINT MD)
1041 006120 000440 BR 6$ ;GO RESTORE VECTOR
1042 006122 022626 2$: CMP (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1043 006124 004537 010156 JSR R5,CKSTAT ;GO CHECK STATUS
1044 006130 011702 3$: 11702 ;THIS LOCATION WILL CONTAIN EXPECTED STATUS
1045 006132 000001 4$: 1 ;THIS LOCATION WILL CONTAIN CURRENT XFER # (MAX 8)
1046 006134 104007 ERROR 7 ;RETURN HERE IF STATUS ER - WILL EXPECT
1047 ;MAINT, COUNT INCREASE OF FNCT & STAT BITS,
1048 ;CYCLE, READY & IE
1049 006136 000431 BR 6$ ;GO RESTORE VECTOR
1050 006140 104010 ERROR 10 ;RETURN HERE IF WC ER - SHOULD BE 0

```

1051	006142	000427			BR	6\$	:GO RESTORE VECTOR
1052	006144	104011			ERROR	11	:RETURN HERE IF BAR ER-
1053	006146	000425			BR	6\$	:GO RESTORE VECTOR
1054	006150	062737	001002	006130	ADD	#1002,3\$	:RETURN HERE IF OK - ADVANCE EXPECTED STATUS LOC
1055	006156	032737	020000	006130	BIT	#BIT13,3\$	:LOOK FOR OVERFLOW
1056	006164	001403			BEQ	5\$	:BR IF NOT
1057	006166	012737	010700	006130	MOV	#10700,3\$	:FNCT & STAT BITS SHOULD BE ZERO THIS TIME
1058	006174	005237	006132		5\$: INC	4\$	:ADVANCE CURRENT XFER #
1059	006200	022737	000011	006132	CMP	#11,4\$	:HAVE 10 XFERS BEEN DONE
1060	006206	001307			BNE	1\$	:BR IF NOT
1061	006210	004537	010526		JSR	R5,CKDAT1	:NOW GO CHECK DATA
1062	006214	000010			10		:# OF XFER'S TO CHECK
1063	006216	104013			ERROR	13	:RETURN HERE IF DATA ER - (WITH MAINT SET)
1064	006220	000240			NOP		:RESTORE VECTOR NEXT
1065	006222	004737	010132		6\$: JSR	PC,RSTVEC	:GO RESTORE VECTOR

(3) :\*\*\*\*\*  
 (3) :\*TEST 26 TEST THAT A DATI FROM A NON-EXISTANT BUS ADRS SETS 'NEX'  
 (2) :\*\*\*\*\*

(1)	006226	000004			TST26: SCOPE		
1068	006230	012737	000100	001160	MOV	#100,\$TIMES	:DO 100 ITERATIONS
1069	006236	012700	000002		MOV	#2,R0	:R0 WHEN ZERO SAYS CLR 'NEX' WITH RESET
1070	006242	004537	010112		1\$: JSR	R5,SETVEC	:GO SET UP INTERRUPT RETURN
1071	006246	006340			2\$		:RETURN TO 2\$ ON TIMEOUT INTR
1072	006250	012777	177777	173242	MOV	#-1,@DRVWCR	:SET UP FOR ONE XFER'S
1073	006256	012777	160000	173236	MOV	#160000,@DRVBAR	:SET UP CA TO 160000 (RESERVED)
1074	006264	106427	000000		MTPS	#0	:ALLOW INTR
1075	006270	012777	000161	173226	MOV	#161,@DRVCSR	:SET IE, XAD 17,16 & GO
1076	006276	052777	000400	173220	BIS	#400,@DRVCSR	:SET CYCLE
1077	006304	013737	001524	001122	MOV	DRVCSR,\$BDADR	:SET UP CSR ADRS - SHOULD NEVER GET HERE
1078	006312	012737	140760	001124	MOV	#140760,\$GDDAT	:LD EXPECTED
1079	006320	017737	173200	001126	MOV	@DRVCSR,\$BDDAT	:GIVE THEM THE STATUS
1080	006326	042777	000100	173170	BIC	#100,@DRVCSR	:CLR IE
1081	006334	104016			ERROR	16	:NEX FAILED TO CAUSE AN INTERRUPT
1082	006336	000521			BR	7\$	:GO RESTORE VECTOR
1083	006340	022626			2\$: CMP	(SP)+,(SP)+	:SHOULD INTR RETURN HERE - FIX STACK
1084	006342	017737	173156	001126	MOV	@DRVCSR,\$BDDAT	:READ THE CSR
1085	006350	012737	140760	001124	MOV	#140760,\$GDDAT	:LD EXPECTED
1086	006356	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:CORRECT?
1087	006364	001405			BEQ	3\$	:BR IF SO
1088	006366	013737	001524	001122	MOV	DRVCSR,\$BDADR	:SET UP CSR ADRS
1089	006374	104016			ERROR	16	:STATUS ER - EXPECTED
1090	006376	000501			BR	7\$	:ER, NEX, CYCLE, READY, IE, XAD17 & XAD16
1091	006400	017737	173114	001126	3\$: MOV	@DRVWCR,\$BDDAT	:GO RESTORE VECTOR
1092	006406	012737	177777	001124	MOV	#-1,\$GDDAT	:READ WORD COUNT
1093	006414	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:SHOULD STILL HAVE -1
1094	006422	001405			BEQ	4\$	:CORRECT?
1095	006424	013737	001520	001122	MOV	DRVWCR,\$BDADR	:BR IF SO
1096	006432	104010			ERROR	10	:SET UP WCR ADRS
1097	006434	000462			BR	7\$	:WC INCREMENTED ON A TIMEOUT ER
1098	006436	017737	173060	001126	4\$: MOV	@DRVBAR,\$BDDAT	:GO RESTORE VECTOR
1099	006444	012737	160000	001124	MOV	#160000,\$GDDAT	:READ BUFFER ADRS
1100	006452	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:SHOULD NOT HAVE INCREMENTED
1101	006460	001405			BEQ	5\$	:CORRECT?
1102	006462	013737	001522	001122	MOV	DRVBAR,\$BDADR	:BR IF SO

```

1103 006470 104011          ERROR 11          :BAR INCREMENTED ON A TIMEOUT ER
1104 006472 000443          BR      7$          :GO RESTORE VECTOR
1105 006474 005300          5$: DEC      R0          :KEEP TRACK ON HOW TO CLR
1106 006476 001422          BEQ     6$          :BR IF CLR BY RESET
1107 006500 042777 040000 173016 BIC     #40000,@DRVCSR :WRITE NEX TO ZERO
1108 006506 017737 173012 001126 MOV     @DRVCSR,$BDDAT :READ CSR
1109 006514 012737 000760 001124 MOV     #760,$GDDAT    :LD EXPECTED
1110 006522 023737 001124 001126 CMP     $GDDAT,$BDDAT :CORRECT?
1111 006530 001644          BEQ     1$          :BR IF SO + REPEAT TEST FOR RESET TEST
1112 006532 013737 001524 001122 MOV     DRVCSR,$BDADR  :SET UP CSR ADRS
1113 006540 104016          ERROR 16          :'NEX' FAILED TO WRITE TO ZERO
1114 006542 000417          BR      7$          :GO RESTORE VECTOR
1115 006544 000005          6$: RESET          :ISSUE BUS RESET
1116 006546 017737 172752 001126 MOV     @DRVCSR,$BDDAT :READ THE CSR
1117 006554 012737 000200 001124 MOV     #200,$GDDAT    :EXPECT ONLY READY
1118 006562 023737 001124 001126 CMP     $GDDAT,$BDDAT :CORRECT?
1119 006570 001404          BEQ     7$          :BR IF SO
1120 006572 013737 001524 001122 MOV     DRVCSR,$BDADR  :SET UP CSR ADRS
1121 006600 104016          ERROR 16          :RESET FAILED TO CLR 'NEX'
1122 006602 004737 010132          7$: JSR     PC,RSTVEC   :GO RESTORE VECTOR
1123
(3)
(3)
(2) 006606 000004          :*****
(1) 006610 012737 000010 001160 *TEST 27 TEST 200 NPR TRANSFERS IN MAINT MODE
1124 006616 004537 010112          :*****
1125 006622 006734          TST27: SCOPE
1126 006624 004737 010410          MOV     #10,$TIMES    ;;DO 10 ITERATIONS
1127 006630 012777 015400 172664 JSR     R5,SETVEC     :GO SET UP INTR RETURN
1128 006636 012777 177470 172654 2$          :RETURN TO 2$ ON INTR
1129 006644 106427 000000          JSR     PC,LDBUF1    :GO SET UP DBUF (SPECIAL COM PATTERN)
1130 006650 012777 010101 172646 MOV     #DBUF,@DRVBAR :SET UP CURRENT ADRS
1131 006656 052777 000400 172640 MOV     #-200.,@DRVWCR :SET UP FOR 200 XFER'S
1132 006664 012737 000000 001126 MTPS   #0            :ALLOW INTR
1133 006672 005237 001126          MOV     #10101,@DRVCSR :SET MAINT, IE & GO
1134 006676 001375          BIS     #400,@DRVCSR  :SET CYCLE
1135 006700 013737 001524 001122 MOV     #0,$BDDAT     :SET UP A COUNTER
1136 006706 012737 010700 001124 1$: INC     $BDDAT     :COUNT AWAY
1137 006714 017737 172604 001126 BNE     1$          :WAIT TILL DONE - SHOULD INTR BEFORE OVFL0
1138 006722 042777 000100 172574 MOV     DRVCSR,$BDADR  :SET UP CSR ADRS
1139 006730 104005          MOV     #10700,$GDDAT :LD EXPECTED
1140 006732 000420          MOV     @DRVCSR,$BDDAT :READ STATUS
1141 006734 022626          BIC     #100,@DRVCSR  :DISABLE IE
1142 006736 004537 010156          ERROR 5            :NO INTR AFTER 200 MAINT MODE XFER'S
1143 006742 010700          BR      3$          :GO RESTORE VECTOR
1144 006744 000310          2$: CMP     (SP)+,(SP)+ :RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1145 006746 104007          JSR     R5,CKSTAT    :GO CHECK STATUS
1146          10700          :EXPECTED STATUS
1147 006750 000411          200.          :# OF XFER'S
1148 006752 104010          BR      3$          :RETURN HERE IF STATUS ER - EXPECTED MAINT,
1149 006754 000407          ERROR 7          :CYCLE, READY & IE
1150 006756 104011          BR      3$          :GO RESTORE VECTOR
1151 006760 000405          ERROR 10         :RETURN HERE IF WC ER - SHOULD BE 0
1152 006762 004537 010526          BR      3$          :GO RESTORE VECTOR
1153 006766 000310          JSR     R5,CKDAT1    :RETURN HERE IF OK - NOW GO CHECK DATA
1154 006770 104013          200.          :# OF XFER'S TO CHECK
          ERROR 13          :RETURN HERE IF DATA ER - (WITH MAINT SET)
  
```



```

1155 006772 000240
1156 006774 004737 010132
1157
(3)
(3)
(2) 007000 000004
(1) 007002 012737 000005 001160
1158 007010 004537 010112
1159 007014 007150
1160 007016 005037 001542
1161 007022 062737 020000 001542
1162 007030 023737 001540 001542
1163 007036 001465
1164 007040 004737 010410
1165 007044 013777 001542 172450
1166 007052 012777 177470 172440
1167 007060 106427 000000
1168 007064 012777 010101 172432
1169 007072 052777 000400 172424
1170 007100 012737 000000 001126
1171 007106 005237 001126
1172 007112 001375
1173 007114 013737 001524 001122
1174 007122 012737 010700 001124
1175 007130 017737 172370 001126
1176 007136 042777 000100 172360
1177 007144 104005
1178 007146 000421
1179 007150 022626
1180 007152 004537 010156
1181 007156 010700
1182 007160 000310
1183 007162 104007
1184
1185 007164 000412
1186 007166 104010
1187 007170 000410
1188 007172 104011
1189 007174 000406
1190 007176 004537 010526
1191 007202 000310
1192 007204 104013
1193 007206 000401
1194 007210 000704
1195 007212 012737 015400 001542
1196 007220 004737 010132
1197
(3)
(3)
(2) 007224 000004
1198 007226 105777 171706
1199 007232 100507
1200 007234 004537 010112
1201 007240 007344
1202 007242 012777 177777 172250
1203 007250 013737 001150 001542

3$: NOP ;RESTORE VECTOR NEXT
JSR PC,RSTVEC ;GO RESTORE VECTOR
:*****
:*TEST 30 TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILABLE 4K
:*****
TST30: SCOPE
MOV #5,$TIMES ;;DO 5 ITERATIONS
JSR R5,SETVEC ;GO SET UP INTR RETURN
3$ ;RETURN TO 3$ ON INTR
CLR DBUFP ;GET LOWEST BUFFER ADRS
1$: ADD #20000,DBUFP ;POINT TO NEXT 4K
CMP CORSZ,DBUFP ;IS THE 4K THERE?
BEQ 4$ ;BR IF NOT
JSR PC,LDBUF1 ;GO SET UP SPECIAL COMPEMENT PATTERN
MOV DBUFP,@DRVBAR ;SET UP BUFFER ADRS
MOV #-200.,@DRVWCR ;SET UP FOR 200 XFER'S
MTPS #0 ;ALLOW INTR
MOV #10101,@DRVCSR ;SET MAINT,IE & GO
BIS #400,@DRVCSR ;SET CYCLE
MOV #0,$BDDAT ;SET UP A COUNTER
2$: INC $BDDAT ;COUNT AWAY
BNE 2$ ;SHOULD ALWAYS INTR FROM THIS LOOP
MOV DRVCSR,$BDDADR ;SET UP CSR ADRA
MOV #10700,$GDDAT ;LD EXPECTED
MOV @DRVCSR,$BDDAT ;READ CSR
BIC #100,@DRVCSR ;DISABLE IE
ERROR 5 ;NO INTR AFTER 200 MAINT MODE XFER'S
BR 4$ ;GO RESTORE VECTOR
3$: CMP (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
JSR R5,CKSTAT ;GO CHECK STATUS
10700 ;EXPECTED STATUS
200. ;# OF XFER'S
ERROR 7 ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
;CYCLE, READY & IE
BR 4$ ;GO RESTORE VECTOR
ERROR 10 ;RETURN HERE IF WC ER - SHOULD = 0
BR 4$ ;GO RESTORE VECTOR
ERROR 11 ;RETURN HERE IF BAR ER - SHOULD = DBUFP+620
BR 4$ ;GO RESTORE VECTOR
JSR R5,CKDAT1 ;RETURN HERE IF OK - NOW GO CK DATA
200. ;# OF XFER'S TO CK
ERROR 13 ;RETURN HERE IF DATA ER
BR 4$ ;GO RESTORE VECTOR
BR 1$ ;TRY NEXT BANK
4$: MOV #DBUF,DBUFP ;RESTORE BUFFER ADRS TO LOWEST 4K
JSR PC,RSTVEC ;GO RESTORE VECTOR
:*****
:*TEST 31 TEST THE ADDRESS (I/O) ABILITY TO THE TTY PRINTER CSR
:*****
TST31: SCOPE
TSTB @SWR ;KDF11-B?
BMI ALTERN ;IF YES,GO TO THE NEXT SUBTEST
101$: JSR R5,SETVEC ;GO SET UP INTR RETURN
1$ ;RETURN TO 1$ ON INTR
MOV #-1,@DRVWCR ;SET UP WC - 1 XFER
MOV $TPS,DBUFP ;SET UP BUFFER ADRS TO PRINTER CSR ADRS
    
```

```

1204 007256 013777 001150 172236      MOV      $TPS,@DRVBAR      ;SET UP BUFFER ADRS - FROM PRINTER CSR
1205 007264 106427 000000              MTPS     #0                ;ALLOW INTR
1206 007270 005077 172232              CLR      @DRVDBR          ;ZERO THE DBR
1207 007274 012777 000161 172222      MOV      #161,@DRVCSR     ;SET IE, XAD17 & XAD16, & GO
1208 007302 052777 000400 172214      BIS      #400,@DRVCSR     ;SET CYCLE
1209 007310 013737 001524 001122      MOV      DRVCSR,$BDADR    ;SET UP CSR ADRS
1210 007316 012737 000760 001124      MOV      #760,$GDDAT      ;LD EXPECTED STATUS
1211 007324 017737 172174 001126      MOV      @DRVCSR,$BDDAT   ;READ THE CSR
1212 007332 042777 000100 172164      BIC      #100,@DRVCSR     ;DISABLE IE
1213 007340 104005              ERROR    5                ;NO INTR ON 1 WD XFER FROM XCSR
1214 007342 000434              BR       2$               ;GO RESTORE VECTOR
1215 007344 022626              1$:    CMP      (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STK SINCE NO RTI
1216 007346 004537 010156      JSR      R5,CKSTAT        ;GO CK STATUS
1217 007352 000760              760
1218 007354 000001              1
1219 007356 104007              ERROR    7                ;# OF XFER'S
1220
1221 007360 000425              BR       2$               ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
1222 007362 104010              ERROR    10              ;XAD17 & XAD16, RDY & IE
1223 007364 000423              BR       2$               ;GO RESTORE VECTOR
1224 007366 104011              ERROR    11              ;RETURN HERE IF WC ER - EXPECTED 0
1225 007370 000421              BR       2$               ;GO RESTORE VECTOR
1226 007372 017737 171552 001124      MOV      @$TPS,$GDDAT     ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
1227 007400 017737 172122 001126      MOV      @DRVDBR,$BDDAT   ;GO RESTORE VECTOR
1228 007406 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;RETURN HERE IF OK - GET XCSR CONTENTS
1229 007414 001407              BEQ      2$               ;READ DBR
1230 007416 013737 001526 001122      MOV      DRVDBR,$BDADR    ;CORRECT?
1231 007424 013737 001150 001120      MOV      $TPS,$GDADR      ;BR IF SO
1232 007432 104020              ERROR    20              ;SET UP DBR ADRS
1233 007434 012737 015400 001542 2$:    MOV      #DBUF,DBUFP      ;GET ADRS OF DATA (XCSR)
1234 007442 004737 010132              JSR      PC,RSTVEC        ;DATA ER FROM TTY PRINTER CSR
1235 007446 000137 007672              JMP      NXDEV            ;RESTORE BUFFER ADRS TO LOWEST 4K
1236
1237
1238
1239

```

```

:*****
:IN CASE OF KDF11-B TEST SOME SPECIFIED I/O PAGE
:*****

```

```

1240 007452 005737 001544      ALTERN: TST     IOPAGE      ;I/O TRANSFER DESIRED?
1241 007456 001505              BEQ      NXDEV            ;IF NOT, SKIP THE TEST
1242 007460 004537 010112      JSR      R5,SETVEC        ;GO SET UP INTR RETURN
1243 007464 007570              1$
1244 007466 012777 177777 172024      MOV      #-1,@DRVWCR     ;RETURN TO 1$ ON INTR
1245 007474 013737 001544 001542      MOV      IOPAGE,DBUFP     ;SET UP WC - 1 XFER
1246 007502 013777 001544 172012      MOV      IOPAGE,@DRVBAR   ;SET UP BUFFER ADRS TO I/O CSR ADRS
1247 007510 106427 000000              MTPS     #0                ;SET UP BUFFER ADRS - FROM I/O CSR
1248 007514 005077 172006              CLR      @DRVDBR          ;ALLOW INTR
1249 007520 012777 000161 171776      MOV      #161,@DRVCSR     ;ZERO THE DBR
1250 007526 052777 000400 171770      BIS      #400,@DRVCSR     ;SET IE, XAD17 & XAD16, & GO
1251 007534 013737 001524 001122      MOV      DRVCSR,$BDADR    ;SET CYCLE
1252 007542 012737 000760 001124      MOV      #760,$GDDAT      ;SET UP CSR ADRS
1253 007550 017737 171750 001126      MOV      @DRVCSR,$BDDAT   ;LD EXPECTED STATUS
1254 007556 042777 000100 171740      BIC      #100,@DRVCSR     ;READ THE CSR
1255 007564 104005              ERROR    5                ;DISABLE IE
1256 007566 000434              BR       2$               ;NO INTR ON 1 WD XFER FROM XCSR
1257 007570 022626              1$:    CMP      (SP)+,(SP)+ ;GO RESTORE VECTOR
1258 007572 004537 010156      JSR      R5,CKSTAT        ;INTR RETURNS HERE - FIX STK SINCE NO RTI
1259 007576 000760              760                        ;GO CK STATUS
                                ;CSR EXPECTED STATUS

```

```

1260 007600 000001          1          :# OF XFER'S
1261 007602 104007          ERROR 7          :RETURN HERE IF STATUS ER - EXPECTED CYCLE,
1262                                     :XAD17 & XAD16, RDY & IE
1263 007604 000425          BR          2$          :GO RESTORE VECTOR
1264 007606 104010          ERROR 10         :RETURN HERE IF WC ER - EXPECTED 0
1265 007610 000423          BR          2$          :GO RESTORE VECTOR
1266 007612 104011          ERROR 11         :RETURN HERE IF BAR ER - SHOULD = XCSR+2
1267 007614 000421          BR          2$          :GO RESTORE VECTOR
1268 007616 017737 171722 001124      MOV @IOPAGE,$GDDAT :RETURN HERE IF OK - GET XCSR CONTENTS
1269 007624 017737 171676 001126      MOV @DRVDBR,$BDDAT :READ DBR
1270 007632 023737 001124 001126      CMP $GDDAT,$BDDAT :CORRECT?
1271 007640 001407          BEQ         2$          :BR IF SO
1272 007642 013737 001526 001122      MOV DRVDBR,$BADDR :SET UP DBR ADRS
1273 007650 013737 001544 001120      MOV IOPAGE,$GADDR :GET ADRS OF DATA (XCSR)
1274 007656 104020          ERROR 20         :DATA ER FROM TTY PRINTER CSR
1275 007660 012737 015400 001542 2$: MOV #DBUF,DBUFP :RESTORE BUFFER ADRS TO LOWEST 4K
1276 007666 004737 010132          JSR PC,RSTVEC    :GO RESTORE VECTOR
  
```

```

:*****
:DON T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11'S HAVE BEEN TESTED
:*****
  
```

```

1281 007672 000004          NXDEV: SCOPE
1282 007674 000241          NXDEV1: CLC          :CLR CARRY
1283 007676 006037 001536          ROR DMAP          :LOOK FOR NEXT
1284 007702 001432          BEQ $EOP          :BR IF ALL TESTED
1285 007704 062737 000010 001520      ADD #10,DRVWCR    :OFFSET BASE BUS ADRS TO NEXT DRV11B
1286 007712 062737 000010 001522      ADD #10,DRVBAR    :
1287 007720 062737 000010 001524      ADD #10,DRVCSR    :
1288 007726 062737 000010 001526      ADD #10,DRVDBR    :
1289 007734 062737 000004 001530      ADD #4,DRVCT0     :OFFSET VECTOR ADRS TO NEXT
1290 007742 062737 000004 001532      ADD #4,DRVCT2     :
1291 007750 005237 001206          INC $UNIT         :COUNT DEVICE
1292 007754 032737 000001 001536      BIT #1,DMAP       :IS IT SELECTED?
1293 007762 001744          BEQ NXDEV1        :BR IF NOT
1294 007764 000137 002406          JMP RESTRT        :TEST NEXT
1295 .SBTTL END OF PASS ROUTINE
  
```

```

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO START1
  
```

```

(1) 007770          $EOP:
(2) 007770 000240          NOP
(1) 007772 005037 001102          CLR $TSTNM        :ZERO THE TEST NUMBER
(1) 007776 005037 001160          CLR $TIMES        :ZERO THE NUMBER OF ITERATIONS
(1) 010002 005237 001202          INC $PASS         :INCREMENT THE PASS NUMBER
(1) 010006 042737 100000 001202      BIC #100000,$PASS :DON'T ALLOW A NEG. NUMBER
(1) 010014 005327          DEC (PC)+         :LOOP?
(1) 010016 000001          $EOPCT: .WORD 1
(1) 010020 003022          BGT $DOAGN        :YES
(1) 010022 012737          MOV (PC)+,@(PC)+ :RESTORE COUNTER
(1) 010024 000001          $ENDCT: .WORD 1
(1) 010026 010016          $EOPCT
(1) 010030 104401 010075          TYPE ,SENDMG     :TYPE 'END PASS #'
  
```

```

(2) 010034 013746 001202          MOV    $PASS,-(SP)      ;;SAVE $PASS FOR TYPEOUT
(2) 010040 104405                    TYPDS                    ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 010042 104401 010072          TYPE    , $NULL          ;;TYPE A NULL CHARACTER
(1) 010046 013700 000042          $GET42: MOV    @#42,R0    ;;GET MONITOR ADDRESS
(1) 010052 001405                    BEQ    $DOAGN            ;;BRANCH IF NO MONITOR
(1) 010054 000005                    RESET                   ;;CLEAR THE WORLD
(1) 010056 004710          $ENDAD: JSR    PC,(R0)    ;;GO TO MONITOR
(1) 010060 000240                    NOP                      ;;SAVE ROOM
(1) 010062 000240                    NOP                      ;;FOR
(1) 010064 000240                    NOP                      ;;ACT11
(1) 010066                    $DOAGN:
(1) 010066 000137          JMP    @(PC)+            ;;RETURN
(1) 010070 002012          $RTNAD: .WORD  START1
(1) 010072 377 377 000          $NULL: .BYTE  -1,-1,0    ;;NULL CHARACTER STRING
(1) 010075 015 042412 042116          $ENDMG: .ASCIZ <15><12>/END PASS #/
(1) 010102 050040 051501 020123
(1) 010110 000043
  
```

K 3

1297  
 1298  
 1299  
 1300  
 1301  
 1302  
 1303  
 1304  
 1305 010112 106427 000200  
 1306 010116 012577 171406  
 1307 010122 012777 000200 171402  
 1308 010130 000205  
 1309  
 1310  
 1311  
 1312  
 1313  
 1314 010132 005077 171366  
 1315 010136 013777 001532 171364  
 1316 010144 005077 171362  
 1317 010150 106427 000200  
 1318 010154 000207  
 1319  
 1320  
 1321  
 1322  
 1323  
 1324  
 1325  
 1326 010156 017737 171342 001126  
 1327 010164 042777 000100 171332  
 1328 010172 012537 001124  
 1329 010176 023737 001124 001126  
 1330 010204 001406  
 1331 010206 013737 001524 001122  
 1332 010214 062705 000002  
 1333 010220 000205  
 1334 010222 017737 171272 001126 1\$:  
 1335 010239 001410  
 1336 010232 013737 001520 001122  
 1337 010240 005037 001124  
 1338 010244 062705 000006  
 1339 010250 000205  
 1340 010252 011537 001124 2\$:  
 1341 010256 006337 001124  
 1342 010262 063737 001542 001124  
 1343 010270 017737 171226 001126  
 1344 010276 042737 000001 001126  
 1345 010304 023737 001124 001126  
 1346 010312 001406  
 1347 010314 013737 001522 001122  
 1348 010322 062705 000012  
 1349 010326 000205  
 1350 010330 062705 000016 3\$:  
 1351 010334 000205  
 1352

```

.SBTTL PROGRAM SUBROUTINES

:*****
:THIS ROUTINE SETS THE PRIORITY LEVEL FOR NO INTERRUPT -
:SETS UP THE DRV11B INTERRUPT TO RETURN ON INTERRUPT
:TO THE ADDRESS INDICATED ((R5)) BY THE CALL +2
:*****
SETVEC: MTPS #200 ;SET UP FOR NO INTERRUPT
        MOV (R5)+,@DRVCT0 ;SET UP INTR RETURN ADRS
        MOV #200,@DRVCT2 ;KEEP PRIORITY LEVEL AT TOP 'N INTR
        RTS R5 ;EXIT

:*****
:THIS ROUTINE CLEARS THE DRV11B CSR - RESTORES THE DRV11B
:INTERRUPT VECTOR TO A HALT - RAISES PRIORITY LEVEL
:*****
RSTVEC: CLR @DRVCSR ;CLR STATUS & CONTROL
        MOV DRVCT2,@DRVCT0 ;POINT VECTOR TO HALT
        CLR @DRVCT2 ;SET UP HALT
        MTPS #200 ;RAISE PRIORITY LEVEL
        RTS PC ;EXIT

:*****
:THIS ROUTINE CHECKS ON ALL DATA TRANSFERS FOR: CORRECT STATUS,
:CORRECT WORD COUNT & CORRECT BUFFER ADDRESS - THE EXPECTED DATA IS
:SUPPLIED IN THE CALL +2 & +4 - THE RETURN IS TO +22 IF NO ERRORS
:DETECTED - IF AN ERROR IS DETECTED THE RETURN IS TO THE APPROPRIATE ERROR EMT
:*****
CKSTAT: MOV @DRVCSR,$BDDAT ;READ THE STATUS
        BIC #100,@DRVCSR ;DISABLE THE IE BIT
        MOV (R5)+,$GDDAT ;SET UP EXPECTED STATUS
        CMP $GDDAT,$BDDAT ;CORRECT?
        BEQ 1$ ;BR IF SO
        MOV DRVCSR,$BDADR ;SET UP CSR ADRS
        ADD #2,R5 ;POINT TO THE CSR ER
        RTS R5 ;EXIT HERE ON STATUS ERROR
1$: MOV @DRVWCR,$BDDAT ;GET WC
    BEQ 2$ ;BR IF ZERO
    MOV DRVWCR,$BDADR ;SET UP WCR ADRS
    CLR $GDDAT ;EXPECTED 0
    ADD #6,R5 ;POINT TO THE WCR ER
    RTS R5 ;EXIT HERE ON WCR ER
2$: MOV (R5),$GDDAT ;GET XFER #
    ASL $GDDAT ;CONVERT TO WORD
    ADD DBUFP,$GDDAT ;POINT TO LAST XFER +2
    MOV @DRVBAR,$BDDAT ;GET BA
    BIC #BIT00,$BDDAT ;DON'T WANT BIT00
    CMP $GDDAT,$BDDAT ;CORRECT?
    BEQ 3$ ;BR IF SO
    MOV DRVBAR,$BDADR ;SET UP BAR ADRS
    ADD #12,R5 ;POINT TO BAR ER
    RTS R5 ;EXIT HERE ON BAR ER
3$: ADD #16,R5 ;ALL OK - POINT TO GOOD EXIT
    RTS R5 ;EXIT HERE IF NO ERRORS
    
```

L 3

1353  
 1354  
 1355  
 1356  
 1357  
 1358  
 1359 010336 012703 015400  
 1360 010342 012704 177776  
 1361 010346 010423  
 1362 010350 005104  
 1363 010352 022703 016216  
 1364 010356 001003  
 1365 010360 012713 070707  
 1366 010364 000207  
 1367 010366 010423  
 1368 010370 022703 016216  
 1369 010374 001771  
 1370 010376 005104  
 1371 010400 006304  
 1372 010402 005204  
 1373 010404 103356  
 1374 010406 000757  
 1375  
 1376  
 1377  
 1378  
 1379  
 1380  
 1381  
 1382 010410 013703 001542  
 1383 010414 010305  
 1384 010416 062705 000620  
 1385 010422 012704 177776  
 1386 010426 010423  
 1387 010430 005023  
 1388 010432 005104  
 1389 010434 010423  
 1390 010436 005023  
 1391 010440 020503  
 1392 010442 001001  
 1393 010444 000207  
 1394 010446 005104  
 1395 010450 006304  
 1396 010452 005204  
 1397 010454 103362  
 1398 010456 000763  
 1399  
 1400  
 1401  
 1402  
 1403  
 1404  
 1405 010460 012701 015400  
 1406 010464 022721 177377  
 1407 010470 001410  
 1408 010472 013737 001526 001122

```

:*****
:THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN
:FOR 199 LOCATIONS - THE LAST LOCATION IS LOADED WITH THE
:#70707 WHICH SHOULD BE THE DATA WORD AVAILABLE IN THE DBR
:AT THE COMPLETION OF A 200 WORD TRANSFER
:*****
LDBUF: MOV #DBUF,R3 ;GET BUFFER ADRS
1$: MOV #177776,R4 ;SET UP FLOATING ZERO PATRN
2$: MOV R4,(R3)+ ;LOAD IT (FLOATING 0)
;COM R4 ;MAKE INTO FLOATING 1
;CMP #DBUF+616,R3 ;AT END OF BUFFER?
;BNE 4$ ;BR IF NOT
3$: MOV #70707,(R3) ;LOAD LAST DATVAR (SPECIAL)
;RTS PC ;GET OUT
4$: MOV R4,(R3)+ ;LOAD IT (FLOATING 1)
;CMP #DBUF+616,R3 ;AT END OF BUFFER?
;BEQ 3$ ;BR IF SO
;COM R4 ;BACK TO FLOATING ZERO
;ASL R4 ;SHIFT LEFT
;INC R4 ;KEEP LSB SET
;BCC 1$ ;GO RESET FLOATING PATRN
;BR 2$ ;GO LOAD NEXT PATRN
    
```

```

:*****
:THIS ROUTINE LOADS 'DBUF' WITH A UNIQUE FLOATING ZERO/ONE PATTERN
:(177776,0,1,0,177775,0,2,0,177773,0,4,0,177767,0,10,0 ETC.)
:IT IS USED WITH MAINT BIT SET (DATI/DATO SEQUENCE) - 200 LOCS
:ARE LOADED WITH THIS PATTERN
:*****
LDBUF1: MOV DBUF,R3 ;GET BUFFER ADRS
;MOV R3,R5 ;SAVE IN R5
;ADD #620,R5 ;POINT TO END OF BUFFER
1$: MOV #177776,R4 ;SET UP FLOATING ZERO PATRN
2$: MOV R4,(R3)+ ;LOAD IT (FLOATING 0)
;CLR (R3)+ ;ZERO NEXT
;COM R4 ;SET UP FLOATING 1
;MOV R4,(R3)+ ;LOAD IT
;CLR (R3)+ ;ZERO NEXT
;CMP R5,R3 ;200 LOCS DONE?
;BNE 3$ ;BR IF NOT
;RTS PC ;GET OUT
3$: ;COM R4 ;BACK TO FLOATING ZERO
;ASL R4 ;SHIFT LEFT
;INC R4 ;KEEP LSB SET
;BCC 1$ ;GO RESET FLOATING PATRN
;BR 2$ ;GO FLOAT NEXT PATRN
    
```

```

:*****
:THIS ROUTINE CHECKS 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED
:DATA (#177377) ON 'DATO' TRANSFERS - IF AN ERROR IS DETECTED
:THE RETURN IS TO CALL +2 - IF NO ERROR THE RETURN IS TO CALL +4
:*****
CKDAT: MOV #DBUF,R1 ;GET BUFFER ADRS
1$: CMP #177377,(R1)+ ;DATA OK?
;BEQ 2$ ;BR IF SO
;MOV DRVDBR,$BDADR ;SET UP DBR ADRS
    
```

```

1409 010500 014137 001126      MOV    -(R1), $BDDAT    ;GET ACTUAL DATA XFERED
1410 010504 010137 001120      MOV    R1, $GDADR      ;GET MEMORY ADRS
1411 010510 000207              RTS    PC               ;RETURN TO ERROR
1412 010512 022701 016220      2$:   CMP    #DBUF+620, R1 ;AT END OF 'DBUF'?
1413 010516 001362              BNE    1$              ;BR IF MORE
1414 010520 062716 000002      ADD    #2, (SP)        ;ADJUST STACK FOR GOOD RETURN
1415 010524 000207              RTS    PC               ;GET OUT

```

```

1416
1417
1418
1419
1420
1421
1422
1423

```

:\*\*\*\*\*  
: THIS ROUTINE CHECK 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED  
: DATA (177776,177776,1,1,177775,177775,2,2,177773,177773,ETC.)  
: ON MAINT MODE TRANSFERS - THE NUMBER OF CHECKS REQUIRED IS INDICATED  
: BY THE CALL +2 - IF AN ERROR IS DETECTED THE RETURN IS TO CALL +4 -  
: IF NO ERROR THE RETURN IS TO CALL +10  
:\*\*\*\*\*

```

1424 010526 012500              CKDAT1: MOV    (R5)+, R0      ;GET # OF CHECKS
1425 010530 013702 001542      MOV    DBUF, R2        ;GET BUFFER ADRS
1426 010534 012701 177776      1$:   MOV    #177776, R1   ;SET UP FLOATING ZERO PATRN
1427 010540 005003              CLR    R3              ;R3 SAYS WHEN TO SHIFT PATRN
1428 010542 020122              2$:   CMP    R1, (R2)+     ;DATA OK?
1429 010544 001010              BNE    3$              ;BR IF NOT
1430 010546 020122              CMP    R1, (R2)+     ;DATA WRITTEN OK?
1431 010550 001006              BNE    3$              ;BR IF NOT
1432 010552 162700 000002      SUB    #2, R0          ;ACCOUNT FOR TWO ADRS'S
1433 010556 003015              BGT    4$              ;BR IF MORE
1434 010560 062705 000004      ADD    #4, R5          ;ADJUST FOR GOOD RETURN
1435 010564 000205              RTS    R5              ;EXIT
1436 010566 014237 001126      3$:   MOV    -(R2), $BDDAT  ;GET BAD DATA
1437 010572 010237 001120      MOV    R2, $GDADR      ;GET MEM ADRS
1438 010576 010137 001124      MOV    R1, $GDDAT      ;LD EXPECTED DATA
1439 010602 013737 001526 001122  MOV    DRVDBR, $BDADR   ;SET UP DBR ADRS
1440 010610 000205              RTS    R5              ;RETURN TO ERROR
1441 010612 005101              4$:   COM    R1           ;NOW EXPECT COMPLEMENT
1442 010614 005103              COM    R3              ;TIME TO SHIFT?
1443 010616 001351              BNE    2$              ;BR IF NOT
1444 010620 006301              ASL    R1              ;SHIFT LEFT
1445 010622 005201              INC    R1              ;KEEP LSB SET
1446 010624 103343              BCC    1$              ;GO RESET FLOATING PATRN
1447 010626 000745              BR     2$              ;DO NEXT

```





```
(1) 011004 004737 011042      JSR    PC,$TYPEC      ;;GO TYPE A NULL
(1) 011010 105337 011160      DECB   $CHARCNT       ;;DO NOT COUNT AS A COUNT
(1) 011014 000770              BR     7$             ;;LOOP
(1)
(1)
(1)      ;HORIZONTAL TAB PROCESSOR
(1) 011016 112716 000040      8$:    MOVB   #' (SP)      ;;REPLACE TAB WITH SPACE
(1) 011022 004737 011042      9$:    JSR    PC,$TYPEC      ;;TYPE A SPACE
(1) 011026 132737 000007 011160  BITB   #7,$CHARCNT     ;;BRANCH IF NOT AT
(1) 011034 001372              BNE    9$             ;;TAB STOP
(1) 011036 005726              TST   (SP)+          ;;POP SPACE OFF STACK
(1) 011040 000724              BR     2$             ;;GET NEXT CHARACTER
(1) 011042
(1) 011042 105777 170076      $TYPEC: TSTB   @$TKS          ;;CHAR IN KYBD BUFFER?      ;MJD001
(1) 011046 100022              BPL   10$            ;;BR IF NOT                ;MJD001
(1) 011050 017746 170072      MOV    @$TKB,-(SP)    ;;GET CHAR                  ;MJD001
(1) 011054 042716 177600      BIC   #177600,(SP)   ;;STRIP EXTRANEIOUS BITS   ;MJD001
(1) 011060 122716 000023      CMPB  #$XOFF,(SP)    ;;WAS CHAR XOFF           ;MJD001
(1) 011064 001012              BNE   102$          ;;BR IF NOT                ;MJD001
(1) 011066
(1) 011066 105777 170052      101$: TSTB   @$TKS          ;;WAIT FOR CHAR            ;MJD001
(1) 011072 100375              BPL   101$          ;;BR IF NOT                ;MJD001
(1) 011074 117716 170046      MOVB  @$TKB,(SP)     ;;GET CHAR                  ;MJD001
(1) 011100 042716 177600      BIC   #177600,(SP)   ;;STRIP IT                  ;MJD001
(1) 011104 122716 000021      CMPB  #$XON,(SP)    ;;WAS IT XON?             ;MJD001
(1) 011110 001366              BNE   101$          ;;BR IF NOT                ;MJD001
(1) 011112
(1) 011112 005726              102$: TST   (SP)+          ;;FIX STACK                ;MJD001
(1) 011114
(1) 011114 105777 170030      10$:  TSTB   @$STPS       ;;WAIT UNTIL PRINTER IS READY ;MJD001
(1) 011120 100375              BPL   10$            ;;BR IF NOT                ;MJD001
(1) 011122 116677 000002 170022  MOVB  2(SP),@$TPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 011130 122766 000015 000002  CMPB  #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
(1) 011136 001003              BNE   1$             ;;BRANCH IF NO
(1) 011140 105037 011160      CLRB  $CHARCNT       ;;YES--CLEAR CHARACTER COUNT
(1) 011144 000406              BR    $TYPEX        ;;EXIT
(1) 011146 122766 000012 000002  1$:   CMPB  #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
(1) 011154 001402              BEQ   $TYPEX        ;;BRANCH IF YES
(1) 011156 105227              INCB  (PC)+         ;;COUNT THE CHARACTER
(1) 011160 000000      $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
(1) 011162 000207      $TYPEX: RTS    PC
(1)
1452      .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)      ;*****
(1) 011164 112737 000001 011430  $ATY1: MOVB   #1,$FFLG      ;;TO REPORT FATAL ERROR
(1) 011172 112737 000001 011426  $ATY3: MOVB   #1,$MFLG      ;;TO TYPE A MESSAGE
(1) 011200 000403              BR    $ATYC
(1) 011202 112737 000001 011430  $ATY4: MOVB   #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
(1) 011210
(1) 011210 010046              $ATYC: MOV    R0,-(SP)     ;;PUSH R0 ON STACK
(1) 011212 010146              MOV    R1,-(SP)     ;;PUSH R1 ON STACK
(1) 011214 105737 011426              TSTB  $MFLG         ;;SHOULD TYPE A MESSAGE?
(1) 011220 001450              BEQ   5$            ;;IF NOT: BR
(1) 011222 122737 000001 001214  CMPB  #APTENV,$ENV   ;;OPERATING UNDER APT?
(1) 011230 001031              BNE   3$            ;;IF NOT: BR
```

```
(1) 011232 132737 000100 001215 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 011240 001425 BEQ 3$ ;;IF NOT: BR
(1) 011242 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
(1) 011246 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 011254 005737 001174 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) 011260 001375 BNE 1$ ;;IF NOT: WAIT
(1) 011262 010037 001210 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
(1) 011266 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
(1) 011270 001376 BNE 2$
(1) 011272 163700 001210 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 011276 006200 ASR R0 ;;GET MESSAGE LNGTH IN WORDS
(1) 011300 010037 001212 MOV R0,$MSGLGT ;;PUT LENGTH IN MAILBOX
(1) 011304 012737 000004 001174 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 011312 000413 BR 5$
(1) 011314 017637 000004 011340 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) 011322 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 011330 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 011334 004737 010630 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 011340 000000 4$: .WORD 0
(1) 011342 5$:
(1) 011342 105737 011430 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
(1) 011346 001416 BEQ 12$ ;;IF NOT: BR
(1) 011350 005737 001214 TST $ENV ;;RUNNING UNDER APT?
(1) 011354 001413 BEQ 12$ ;;IF NOT: BR
(1) 011356 005737 001174 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
(1) 011362 001375 BNE 11$ ;;IF NOT: WAIT
(1) 011364 017637 000004 001176 MOV @4(SP),$FATAL ;;GET ERROR #
(1) 011372 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 011400 005237 001174 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
(1) 011404 105037 011430 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
(1) 011410 105037 011427 CLRB $LFLG ;;CLEAR LOG FLAG
(1) 011414 105037 011426 CLRB $MFLG ;;CLEAR MESSAGE FLAG
(3) 011420 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 011422 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 011424 000207 RTS PC ;;RETURN
(1) 011426 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
(1) 011427 000 $LFLG: .BYTE 0 ;;LOG FLAG
(1) 011430 000 $FFLG: .BYTE 0 ;;FATAL FLAG
(1) 011432 .EVEN
(1) 000200 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTSPOOL=100
(1) 000040 APTCSUP=040
1453 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)
(1) *****
(1) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1) *OCTAL (ASCII) NUMBER AND TYPE IT.
(1) *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1) *CALL:
(1) * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
(1) * TYPOS ;;CALL FOR TYPEOUT
(1) * .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1) * .BYTE M ;;M=1 OR 0
(1) * ;;1=TYPE LEADING ZEROS
(1) * ;;0=SUPPRESS LEADING ZEROS
```

```

(1)          ;*
(1)          ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)          ;*$TYPOS OR $TYPOC
(1)          ;*CALL:
(1)          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(1)          ;*      TYPON                    ;;CALL FOR TYPEOUT
(1)          ;*
(1)          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)          ;*CALL:
(1)          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
(1)          ;*      TYPOC                    ;;CALL FOR TYPEOUT
(1)
(1) 011432 017646 000000          $TYPOS: MOV      @(SP),-(SP)          ;;PICKUP THE MODE
(1) 011436 116637 000001 011655  MOVB     1(SP),$OFILL          ;;LOAD ZERO FILL SWITCH
(1) 011444 112637 011657          MOVB     (SP)+,$OMODE+1          ;;NUMBER OF DIGITS TO TYPE
(1) 011450 062716 000002          ADD      #2,(SP)              ;;ADJUST RETURN ADDRESS
(1) 011454 000406                    BR      $TYPON
(1) 011456 112737 000001 011655  $TYPOC: MOVB     #1,$OFILL          ;;SET THE ZERO FILL SWITCH
(1) 011464 112737 000006 011657  MOVB     #6,$OMODE+1          ;;SET FOR SIX(6) DIGITS
(1) 011472 112737 000005 011654  $TYPON: MOVB     #5,$OCNT          ;;SET THE ITERATION COUNT
(1) 011500 010346                    MOV      R3,-(SP)              ;;SAVE R3
(1) 011502 010446                    MOV      R4,-(SP)              ;;SAVE R4
(1) 011504 010546                    MOV      R5,-(SP)              ;;SAVE R5
(1) 011506 113704 011657          MOVB     $OMODE+1,R4          ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 011512 005404                    NEG      R4
(1) 011514 062704 000006          ADD      #6,R4                ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 011520 110437 011656          MOVB     R4,$OMODE          ;;SAVE IT FOR USE
(1) 011524 113704 011655          MOVB     $OFILL,R4          ;;GET THE ZERO FILL SWITCH
(1) 011530 016605 000012          MOV      12(SP),R5          ;;PICKUP THE INPUT NUMBER
(1) 011534 005003                    CLR      R3
(1) 011536 006105                    1$:     ROL      R5              ;;CLEAR THE OUTPUT WORD
(1) 011540 000404                    BR      3$                    ;;ROTATE MSB INTO 'C'
(1) 011542 006105                    2$:     ROL      R5              ;;GO DO MSB
(1) 011544 006105                    ROL      R5                    ;;FORM THIS DIGIT
(1) 011546 006105                    ROL      R5
(1) 011550 010503                    MOV      R5,R3
(1) 011552 006103                    3$:     ROL      R3              ;;GET LSB OF THIS DIGIT
(1) 011554 105337 011656          DECB     $OMODE          ;;TYPE THIS DIGIT?
(1) 011560 100016                    BPL      7$                    ;;BR IF NO
(1) 011562 042703 177770          BIC      #177770,R3          ;;GET RID OF JUNK
(1) 011566 001002                    BNE      4$                    ;;TEST FOR 0
(1) 011570 005704                    TST      R4                    ;;SUPPRESS THIS 0?
(1) 011572 001403                    BEQ      5$                    ;;BR IF YES
(1) 011574 005204                    4$:     INC      R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1) 011576 052703 000060          BIS      #'0,R3              ;;MAKE THIS DIGIT ASCII
(1) 011602 052703 000040          5$:     BIS      #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1) 011606 110337 011652          MOVB     R3,8$                ;;SAVE FOR TYPING
(1) 011612 104401 011652          TYPE     ,8$                  ;;GO TYPE THIS DIGIT
(1) 011616 105337 011654          7$:     DECB     $OCNT          ;;COUNT BY 1
(1) 011622 003347                    BGT      2$                    ;;BR IF MORE TO DO
(1) 011624 002402                    BLT      6$                    ;;BR IF DONE
(1) 011626 005204                    INC      R4                    ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 011630 000744                    BR      2$                    ;;GO DO THE LAST DIGIT
(1) 011632 012605                    6$:     MOV      (SP)+,R5          ;;RESTORE R5
(1) 011634 012604                    MOV      (SP)+,R4          ;;RESTORE R4
(1) 011636 012603                    MOV      (SP)+,R3          ;;RESTORE R3
    
```

```
(1) 011640 016666 000002 000004      MOV      2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
(1) 011646 012616      MOV      (SP)+,(SP)
(1) 011650 000002      RTI                      ;;RETURN
(1) 011652      000      8$:      .BYTE 0                ;;STORAGE FOR ASCII DIGIT
(1) 011653      000      .BYTE 0                ;;TERMINATOR FOR TYPE ROUTINE
(1) 011654      000      $OCNT: .BYTE 0           ;;OCTAL DIGIT COUNTER
(1) 011655      000      $OFILL: .BYTE 0          ;;ZERO FILL SWITCH
(1) 011656 000000      $OMODE: .WORD 0         ;;NUMBER OF DIGITS TO TYPE
1454      .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)      ;;*****
(1)      ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)      ;;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)      ;;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)      ;;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)      ;;REPLACED WITH SPACES.
(1)      ;;CALL:
(1)      ;;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
(1)      ;;*      TYPDS      ;;GO TO THE ROUTINE
(1)
(1) 011660      $TYPDS:
(3) 011660 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 011662 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 011664 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(3) 011666 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
(3) 011670 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
(1) 011672 012746 020200      MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
(1) 011676 016605 000020      MOV      20(SP),R5    ;;GET THE INPUT NUMBER
(1) 011702 100004      BPL      1$           ;;BR IF INPUT IS POS.
(1) 011704 005405      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
(1) 011706 112766 000055 000001      MOVVB   #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
(1) 011714 005000      1$:      CLR      R0           ;;ZERO THE CONSTANTS INDEX
(1) 011716 012703 012074      MOV      #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
(1) 011722 112723 000040      MOV:    #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
(1) 011726 005002      2$:      CLR      R2           ;;CLEAR THE BCD NUMBER
(1) 011730 016001 012064      MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
(1) 011734 160105      3$:      SUB      R1,R5        ;;FORM THIS BCD DIGIT
(1) 011736 002402      BLT     4$           ;;BR IF DONE
(1) 011740 005202      INC     R2           ;;INCREASE THE BCD DIGIT BY 1
(1) 011742 000774      BR      3$
(1) 011744 060105      4$:      ADD      R1,R5        ;;ADD BACK THE CONSTANT
(1) 011746 005702      TST     R2           ;;CHECK IF BCD DIGIT=0
(1) 011750 001002      BNE     5$           ;;FALL THROUGH IF 0
(1) 011752 105716      TSTB   (SP)          ;;STILL DOING LEADING 0'S?
(1) 011754 100407      BMI     7$           ;;BR IF YES
(1) 011756 106316      5$:      ASLB   (SP)          ;;MSD?
(1) 011760 103003      BCC     6$           ;;BR IF NO
(1) 011762 116663 000001 177777      MOVVB   1(SP),-1(R3)  ;;YES--SET THE SIGN
(1) 011770 052702 000060      6$:      BIS     #'0,R2        ;;MAKE THE BCD DIGIT ASCII
(1) 011774 052702 000040      7$:      BIS     #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 012000 110223      MOVVB   R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 012002 005720      TST     (R0)+        ;;JUST INCREMENTING
(1) 012004 020027 000010      CMP     R0,#10       ;;CHECK THE TABLE INDEX
(1) 012010 002746      BLT     2$           ;;GO DO THE NEXT DIGIT
(1) 012012 003002      BGT     8$           ;;GO TO EXIT
(1) 012014 010502      MOV     R5,R2        ;;GET THE LSD
```

```
(1) 012016 000764 BR 6$ ;:GO CHANGE TO ASCII
(1) 012020 105726 8$: TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
(1) 012022 100003 BPL 9$ ;:BR IF NO
(1) 012024 116663 177777 177776 MOVB -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
(1) 012032 105013 9$: CLRB (R3) ;:SET THE TERMINATOR
(3) 012034 012605 MOV (SP)+,R5 ;:POP STACK INTO R5
(3) 012036 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
(3) 012040 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
(3) 012042 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
(3) 012044 012600 MOV (SP)+,R0 ;:POP STACK INTO R0
(1) 012046 104401 012074 TYPE $DBLK ;:NOW TYPE THE NUMBER
(1) 012052 016666 000002 000004 MOV 2(SP),4(SP) ;:ADJUST THE STACK
(1) 012060 012616 MOV (SP)+,(SP) ;:RETURN TO USER
(1) 012062 000002 RTI ;:RETURN TO USER
(1) 012064 023420 $DTBL: 10000.
(1) 012066 001750 1000.
(1) 012070 000144 100.
(1) 012072 000012 10.
(1) 012074 000004 $DBLK: .BLKW 4
1455 .SBTTL ERROR HANDLER ROUTINE
(1) ;:*****
(1) ;:*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1) ;:*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) ;:*AND GO TO SWRCK ON ERROR
(1) ;:*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) ;:*SW15=1 HALT ON ERROR
(1) ;:*SW13=1 INHIBIT ERROR TYPEOUTS
(1) ;:*SW10=1 BELL ON ERROR
(1) ;:*SW09=1 LOOP ON ERROR
(1) ;:*CALL
(1) ;:* ERROR N ;:ERROR=EMT AND N=ERROR ITEM NUMBER
(1) $ERROR:
(1) 012104 CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
(1) 012104 104407 CKSWR ;:GO LOOK FOR SWR CHANGE
(2) 012106 104407 7$: INCB $ERFLG ;:SET THE ERROR FLAG
(1) 012110 105237 001103 BEQ 7$ ;:DON'T LET THE FLAG GO TO ZERO
(1) 012114 001775 MOV $TSTNM,@DISPLAY ;:DISPLAY TEST NUMBER AND ERROR FLAG
(1) 012116 013777 001102 167016 BIT #BIT10,@SWR ;:BELL ON ERROR?
(1) 012124 032777 002000 167006 BEQ 1$ ;:NO - SKIP
(1) 012132 001402 TYPE $BELL ;:RING BELL
(1) 012134 104401 001164 INC $ERTL ;:COUNT THE NUMBER OF ERRORS
(1) 012140 005237 001112 1$: MOV (SP),$ERRPC ;:GET ADDRESS OF ERROR INSTRUCTION
(1) 012144 011637 001116 SUB #2,$ERRPC
(1) 012150 162737 000002 001116 MOVB @$ERRPC,$ITEMB ;:STRIP AND SAVE THE ERROR ITEM CODE
(1) 012156 117737 166734 001114 BIT #BIT13,@SWR ;:SKIP TYPEOUT IF SET
(1) 012164 032777 020000 166746 BNE 20$ ;:SKIP TYPEOUTS
(1) 012172 001004 JSR PC,SWRCK ;:GO TO USER ERROR ROUTINE
(1) 012174 004737 012274 TYPE $CRLF
(1) 012200 104401 001171 20$: CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
(1) 012204 122737 000001 001214 BNE 2$ ;:NO SKIP APT ERROR REPORT
(1) 012212 001007 MOV $ITEMB,21$ ;:SET ITEM NUMBER AS ERROR NUMBER
(1) 012214 113737 001114 012226 JSR PC,$ATY4 ;:REPORT FATAL ERROR TO APT
(1) 012222 004737 011202 21$: .BYTE 0
(1) 012226 000
```

```

(1) 012227 000
(1) 012230 000777
(1) 012232 005777 166702
(1) 012236 100002
(1) 012240 000000
(1) 012242 104407
(1) 012244 032777 001000 166666
(1) 012252 001402
(1) 012254 013716 001110
(1) 012260 005737 001162
(1) 012264 001402
(1) 012266 013716 001162
(1) 012272
(1) 012272 000002
1456
1457
1458
1459
1460 012274 113737 001102 001534
1461 012302 004737 012312
1462 012306 104407
1463 012310 000207
1464
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 012312
(1) 012312 104401 001171
(1) 012316 010046
(1) 012320 005000
(1) 012322 153700 001114
(1) 012326 001004
(2) 012330 013746 001116
(2)
(2) 012334 104402
(1) 012336 000426
(1) 012340 005300
(1) 012342 006300
(1) 012344 006300
(1) 012346 006300
(1) 012350 062700 001320
(1) 012354 012037 012364
(1) 012360 001404
(1) 012362 104401
(1) 012364 000000
(1) 012366 104401 001171
(1) 012372 012037 012402
(1) 012376 001404
(1) 012400 104401
(1) 012402 000000
(1) 012404 104401 001171
(1) 012410 011000

        .BYTE 0
22$: BR 22$
2$: TST @SWR
        BPL 3$
        HALT
        CKSWR
3$: BIT #BIT09,@SWR
        BEQ 4$
4$: MOV $LPERR,(SP)
        TST $ESCAPE
        BEQ 5$
        MOV $ESCAPE,(SP)
5$:
        RTI
        ::RETURN
        *****
        :GO TYPE ERROR
        :GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
        *****
SWRCK: MOVB $TSTNM,TSTNUM ;SET UP TEST # ON ER
        JSR PC,$ERRTYP ;GO TYPE ERROR
        CKSWR ;GO LOOK FOR SWR CHANGE
        RTS PC ;RETURN TO ERROR HANDLER
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE
        *****
        :*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
        :*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
        :*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
$ERRTYP:
        TYPE , $CRLF ;'"CARRIAGE RETURN" & "LINE FEED"'
        MOV RO,-(SP) ;SAVE RO
        CLR RO ;PICKUP THE ITEM INDEX
        BISB @#$ITEMB,RO
        BNE 1$ ;IF ITEM NUMBER IS ZERO, JUST
        ;TYPE THE PC OF THE ERROR
        MOV $ERRPC,-(SP) ;SAVE $ERRPC FOR TYPEOUT
        ;ERROR ADDRESS
        TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
        BR 6$ ;GET OUT
1$: DEC RO ;ADJUST THE INDEX SO THAT IT WILL
        ASL RO ;WORK FOR THE ERROR TABLE
        ASL RO
        ASL RO
        ADD #$ERRTB,RO ;FORM TABLE POINTER
        MOV (RO)+,2$ ;PICKUP "ERROR MESSAGE" POINTER
        BEQ 3$ ;SKIP TYPEOUT IF NO POINTER
        TYPE ;TYPE THE "ERROR MESSAGE"
        ;"ERROR MESSAGE" POINTER GOES HERE
2$: .WORD 0 ;'"CARRIAGE RETURN" & "LINE FEED"'
        TYPE , $CRLF ;PICKUP "DATA HEADER" POINTER
3$: MOV (RO)+,4$ ;SKIP TYPEOUT IF 0
        BEQ 5$ ;TYPE THE "DATA HEADER"
        TYPE ;"DATA HEADER" POINTER GOES HERE
4$: .WORD 0 ;'"CARRIAGE RETURN" & "LINE FEED"'
        TYPE , $CRLF ;PICKUP "DATA TABLE" POINTER
5$: MOV (RO),RO
    
```



```

(1) 012574 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 012602 000446 BR $OVER
(1) 012604 105037 001103 4$: CLR $ERFLG ;;ZERO THE ERROR FLAG
(1) 012610 005037 001160 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 012614 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
(1) 012616 032777 004000 166314 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1) 012624 001011 BNE 1$ ;;BR IF YES
(1) 012626 005737 001202 TST $PASS ;;IF FIRST PASS OF PROGRAM
(1) 012632 001406 BEQ 1$ ;; INHIBIT ITERATIONS
(1) 012634 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 012640 023737 001160 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 012646 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 012650 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1) 012656 013737 012734 001160 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 012664 105237 001102 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
(1) 012670 113737 001102 001200 MOV $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(1) 012676 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(1) 012702 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(1) 012706 005037 001162 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 012712 112737 000001 001115 MOV #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 012720 013777 001102 166214 $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 012726 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(1) 012732 000002 RTI ;;FIXES PS
(1) 012734 003720 $MXCNT: 2000. ;;MAX. NUMBER OF ITERATIONS
1466 .SBTTL TTY INPUT ROUTINE
(1) ;;*****
(2) .ENABL LSB
(1) ;;*****
(2) ;;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ;;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ;;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1) ;;WHEN OPERATING IN TTY FLAG MODE.
(1) 012736 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
(1) 012744 001074 BNE 15$ ;;BRANCH IF NO
(1) 012746 105777 166172 TSTB @TKS ;;CHAR THERE?
(1) 012752 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
(1) 012754 117746 166166 MOV $TKB,-(SP) ;;SAVE THE CHAR
(1) 012760 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
(1) 012764 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
(1) 012770 001062 BNE 15$ ;;NO, RETURN TO USER
(1) 012772 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
(1) 013000 001456 BEQ 15$ ;;BRANCH IF YES
(1) 013002 104401 013473 $GTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (^G)
(1) 013006 104401 013500 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
(2) 013012 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
(2) 013016 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 013020 104401 013511 TYPE , $MNEW ;;PROMPT FOR NEW SWR
(1) 013024 005046 19$: CLR -(SP) ;;CLEAR COUNTER
(1) 013026 005046 CLR -(SP) ;;THE NEW SWR
(1) 013030 105777 166110 7$: TSTB @TKS ;;CHAR THERE?
(1) 013034 100375 BPL 7$ ;;IF NOT TRY AGAIN
(1)
(1) 013036 117746 166104 MOV $TKB,-(SP) ;;PICK UP CHAR
  
```



J 4

```
(1) 013042 042716 177600      BIC      #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1) 013046 021627 000025      9$:   CMP      (SP),#25      ;;IS IT A CONTROL-U?
(1) 013052 001005                BNE      10$               ;;BRANCH IF NOT
(1) 013054 104401 013466          TYPE    $CNTLU            ;;YES, ECHO CONTROL-U (^U)
(1) 013060 062706 000006      20$:   ADD      #6,SP          ;;IGNORE PREVIOUS INPUT
(1) 013064 000757                BR       19$               ;;LET'S TRY IT AGAIN
(1)
(1) 013066 021627 000015      10$:   CMP      (SP),#15      ;;IS IT A <CR>?
(1) 013072 001022                BNE      16$               ;;BRANCH IF NO
(1) 013074 005766 000004          TST     4(SP)            ;;YES, IS IT THE FIRST CHAR?
(1) 013100 001403                BEQ     11$               ;;BRANCH IF YES
(1) 013102 016677 000002 166030    MOV     2(SP),@SWR        ;;SAVE NEW SWR
(1) 013110 062706 000006      11$:   ADD      #6,SP          ;;CLEAR UP STACK
(1) 013114 104401 001171      14$:   TYPE    $CRLF          ;;ECHO <CR> AND <LF>
(1) 013120 123727 001135 000001    CMPB   $INTAG,#1        ;;RE-ENABLE TTY KBD INTERRUPTS?
(1) 013126 001003                BNE     15$               ;;BRANCH IF NOT
(1) 013130 012777 000100 166006    MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1) 013136 000002                RTI    ;                 ;;RETURN
(1) 013140 060737 011042      16$:   JSR     PC,$TYPEC       ;;ECHO CHAR
(1) 013144 021627 000060          CMP     (SP),#60         ;;CHAR < 0?
(1) 013150 002420                BLT    18$               ;;BRANCH IF YES
(1) 013152 021627 000067          CMP     (SP),#67         ;;CHAR > 7?
(1) 013156 003015                BGT    18$               ;;BRANCH IF YES
(1) 013160 042726 000060          BIC     #60,(SP)+        ;;STRIP-OFF ASCII
(1) 013164 005766 000002          TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1) 013170 001403                BEQ     17$               ;;BRANCH IF YES
(1) 013172 006316                ASL    (SP)              ;;NO, SHIFT PRESENT
(1) 013174 006316                ASL    (SP)              ;;CHAR OVER TO MAKE
(1) 013176 006316                ASL    (SP)              ;;ROOM FOR NEW ONE.
(1) 013200 005266 000002      17$:   INC     2(SP)           ;;KEEP COUNT OF CHAR
(1) 013204 056616 177776          BIS    -2(SP),(SP)      ;;SET IN NEW CHAR
(1) 013210 000707                BR     7$                 ;;GET THE NEXT ONE
(1) 013212 104401 001170      18$:   TYPE    $QUES          ;;TYPE ?<CR><LF>
(1) 013216 000720                BR     20$               ;;SIMULATE CONTROL-U
(1)
(1)
(1)
(2)
(1) *****
(1) *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) *CALL:
(1) *      RDCHR              ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1) *      RETURN HERE        ;;CHARACTER IS ON THE STACK
(1) *                          ;;WITH PARITY BIT STRIPPED OFF
(1)
(1)
(1) $RDCHR: MOV      (SP),-(SP)    ;;PUSH DOWN THE PC
(1) 013220 011646                MOV     4(SP),2(SP)     ;;SAVE THE PS
(1) 013222 016666 000004 000002    1$:   TSTB   @$TKS          ;;WAIT FOR
(1) 013230 105777 165710          BPL    1$                 ;;A CHARACTER
(1) 013234 100375                MOVB   @$TKB,4(SP)     ;;READ THE TTY
(1) 013236 117766 165704 000004    BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1) 013244 042766 177600 000004    CMP     4(SP),#25      ;;IS IT A CONTROL-S?
(1) 013252 026627 000004 000023
```

```

(1) 013260 001013          BNE      3$          ;;BRANCH IF NO
(1) 013262 105777 165656 2$:  TSTB    @STKS      ;;WAIT FOR A CHARACTER
(1) 013266 100375          BPL      2$          ;;LOOP UNTIL ITS THERE
(1) 013270 117746 165652  MOVB    @STKB,-(SP)  ;;GET CHARACTER
(1) 013274 042716 177600  BIC     #'C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 013300 022627 000021  CMP     (SP)+,#21   ;;IS IT A CONTROL-Q?
(1) 013304 001366          BNE      2$          ;;IF NOT DISCARD IT
(1) 013306 000750          BR       1$          ;;YES, RESUME
(1) 013310 026627 000004 000021 3$:  CMP     4(SP),#$XON  ;;IS IT A RANDOM XON?          ;RAN001
(1) 013316 001744          BEQ      1$          ;;BRANCH IF YES                ;RAN001
(1) 013320 026627 000004 000140  CMP     4(SP),#140   ;;IS IT UPPER CASE?
(1) 013326 002407          BLT     4$          ;;BRANCH IF YES
(1) 013330 026627 000004 000175  CMP     4(SP),#175   ;;IS IT A SPECIAL CHAR?
(1) 013336 003003          BGT     4$          ;;BRANCH IF YES
(1) 013340 042766 000040 000004  BIC     #40,4(SP)   ;;MAKE IT UPPER CASE
(1) 013346 000002          4$:  RTI          ;;GO BACK TO USER
(2)                                     ;*****
(1)                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*
(1)                                     ;*   RDLIN          ;;INPUT A STRING FROM THE TTY
(1)                                     ;*   RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;*                                     ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1) 013350 010346          $RDLIN: MOV     R3,-(SP) ;;SAVE R3
(1) 013352 012703 013456 1$:  MOV     #$TTYIN,R3  ;;GET ADDRESS
(1) 013356 022703 013466 2$:  CMP     #$TTYIN+8.,R3 ;;BUFFER FULL?
(1) 013362 101405          BLOS    4$          ;;BR IF YES
(1) 013364 104410          RDCHR   ;;GO READ ONE CHARACTER FROM THE TTY
(1) 013366 112613          MOVB    (SP)+,(R3)  ;;GET CHARACTER
(1) 013370 122713 000177 10$:  CMPB   #177,(R3)   ;;IS IT A RUBOUT
(1) 013374 001003          BNE     3$          ;;SKIP IF NOT
(1) 013376 104401 001170 4$:  TYPE   $QUES      ;;TYPE A '?'
(1) 013402 000763          BR      1$          ;;CLEAR THE BUFFER AND LOOP
(1) 013404 111337 013454 3$:  MOVB   (R3),9$     ;;ECHO THE CHARACTER
(1) 013410 104401 013454   TYPE   ,9$
(1) 013414 122723 000015   CMPB   #15,(R3)+   ;;CHECK FOR RETURN
(1) 013420 001356          BNE     2$          ;;LOOP IF NOT RETURN
(1) 013422 105063 177777   CLRB   -1(R3)     ;;CLEAR RETURN (THE 15)
(1) 013426 104401 001172   TYPE   ,SLF      ;;TYPE A LINE FEED
(1) 013432 012603          MOV     (SP)+,R3   ;;RESTORE R3
(1) 013434 011646          MOV     (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 013436 016666 000004 000002  MOV     4(SP),2(SP) ;;FIRST ASCII CHARACTER ON IT
(1) 013444 012766 013456 000004  MOV     #$TTYIN,4(SP)
(1) 013452 000002          RTI          ;;RETURN
(1) 013454 000          9$:  .BYTE  0          ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 013455 000          .BYTE  0          ;;TERMINATOR
(1) 013456 000010          $TTYIN: .BLKB  8.  ;;RESERVE 8 BYTES FOR TTY INPUT
(1) 013466 052536 005015 000   $CNTLU: .ASCIZ  /'U/'<15><12> ;;CONTROL 'U'
(1) 013473 0136 006507 000012  $CNTLG: .ASCIZ  /'G/'<15><12> ;;CONTROL 'G'
(1) 013500 005015 053523 020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1) 013506 020075 000
(1) 013511 040 047040 053505  $MNEW:  .ASCIZ  / NEW = /
(1) 013516 036440 000040
1467
(1) .SBTTL READ AN OCTAL NUMBER FROM THE TTY
(2) ;*****

```

```

(1) ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1) ;*CHANGE IT TO BINARY.
(1) ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
(1) ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
(1) ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
(1) ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
(1) ;*CALL:
(1) ;*      RDOCT          ;;READ AN OCTAL NUMBER
(1) ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1) ;*                  ;;HIGH ORDER BITS ARE IN $HIOCT
(1) $RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1) 013522 011646      MOV      4(SP),2(SP)  ;;INPUT NUMBER
(1) 013524 016666 000004 000002  MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 013532 010046      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 013534 010146      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(3) 013536 010246      1$: RDLIN      ;;READ AN ASCII LINE
(1) 013540 104411      MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
(1) 013542 012600      MOV      R0,5$      ;;AND SAVE IT
(1) 013544 010037 013650  CLR      R1      ;;CLEAR DATA WORD
(1) 013550 005001      CLR      R2
(1) 013552 005002      2$: MOVB     (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
(1) 013554 112046      BEQ      3$      ;;IF ZERO GET OUT
(1) 013556 001420      CMPB     #'0,(SP)  ;;MAKE SURE THIS CHARACTER
(1) 013560 122716 000060  BGT      4$      ;;IS AN OCTAL DIGIT
(1) 013564 003026      CMPB     #'7,(SP)
(1) 013566 122716 000067  BLT      4$
(1) 013572 002423      ASL      R1      ;;*2
(1) 013574 006301      ROL      R2
(1) 013576 006102      ASL      R1      ;;*4
(1) 013600 006301      ROL      R2
(1) 013602 006102      ASL      R1      ;;*8
(1) 013604 006301      ROL      R2
(1) 013606 006102      BIC      #'C7,(SP)  ;;STRIP THE ASCII JUNK
(1) 013610 042716 177770  ADD      (SP)+,R1  ;;ADD IN THIS DIGIT
(1) 013614 062601      BR       2$      ;;LOOP
(1) 013616 000756      3$: TST      (SP)+  ;;CLEAN TERMINATOR FROM STACK
(1) 013620 005726      MOV      R1,12(SP) ;;SAVE THE RESULT
(1) 013622 010166 000012  MOV      R2,$HIOCT
(1) 013626 010237 013660  MOV      (SP)+,R2  ;;POP STACK INTO R2
(3) 013632 012602      MOV      (SP)+,R1  ;;POP STACK INTO R1
(3) 013634 012601      MOV      (SP)+,R0  ;;POP STACK INTO R0
(3) 013636 012600      RTI      ;;RETURN
(1) 013640 000002      4$: TST      (SP)+  ;;CLEAN PARTIAL FROM STACK
(1) 013642 005726      CLRB     (R0)      ;;SET A TERMINATOR
(1) 013644 105010      TYPE     0        ;;TYPE UP THRU THE BAD CHAR.
(1) 013646 104401      .WORD    0
(1) 013650 000000      5$: TYPE     $QUES  ;; "?" "CR" & "LF"
(1) 013652 104401 001170  BR       1$      ;;TRY AGAIN
(1) 013656 000730      $HIOCT: .WORD    0  ;;HIGH ORDER BITS GO HERE
(1) 013660 000000      .SBTTL  POWER DOWN AND UP ROUTINES

(1) ;*****
(1) ;POWER DOWN ROUTINE
(1) 013662 012737 014026 000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 013670 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;;PRIO:7
  
```

```

(3) 013676 010046     MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 013700 010146     MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 013702 010246     MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(3) 013704 010346     MOV      R3,-(SP)      ;;PUSH R3 ON STACK
(3) 013706 010446     MOV      R4,-(SP)      ;;PUSH R4 ON STACK
(3) 013710 010546     MOV      R5,-(SP)      ;;PUSH R5 ON STACK
(3) 013712 017746     MOV      @SWR,-(SP)    ;;PUSH @SWR ON STACK
(1) 013716 010637     MOV      SP,$SAVR6     ;;SAVE SP
(1) 013722 012737     MOV      #SPWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 013730 000000     HALT
(1) 013732 000776     BR       .-2          ;;HANG UP
(1)
(2)
(1)
(1) 013734 012737     014026 000024  $PWRUP: MOV     #SILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
(1) 013742 013706     014032         MOV     $SAVR6,SP          ;;GET SP
(1) 013746 005037     014032         CLR    $SAVR6            ;;WAIT LOOP FOR THE TTY
(1) 013752 005237     014032 1$:      INC    $SAVR6            ;;WAIT FOR THE INC
(1) 013756 001375         BNE   1$                ;;OF WORD
(3) 013760 012677     165154         MOV   (SP)+,@SWR        ;;POP STACK INTO @SWR
(3) 013764 012605         MOV   (SP)+,R5         ;;POP STACK INTO R5
(3) 013766 012604         MOV   (SP)+,R4         ;;POP STACK INTO R4
(3) 013770 012603         MOV   (SP)+,R3         ;;POP STACK INTO R3
(3) 013772 012602         MOV   (SP)+,R2         ;;POP STACK INTO R2
(3) 013774 012601         MOV   (SP)+,R1         ;;POP STACK INTO R1
(3) 013776 012600         MOV   (SP)+,R0         ;;POP STACK INTO R0
(1) 014000 012737     013662 000024  MOV     #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 014006 012737     000340 000026  MOV     #340,@#PWRVEC+2  ;;PRIO:7
(1) 014014 104401         TYPE                                ;;REPORT THE POWER FAILURE
(1) 014016 014034     $PWRMG: .WORD   PWRMSG      ;;POWER FAIL MESSAGE POINTER
(1) 014020 012716         MOV   (PC)+,(SP)       ;;RESTART AT RESTR
(1) 014022 002406     $PWRAD: .WORD   RESTR     ;;RESTART ADDRESS
(1) 014024 000002         RTI
(1) 014026 000000     $SILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
(1) 014030 000776         BR    .-2              ;;BEFORE THE POWER DOWN WAS COMPLETE
(1) 014032 000000     $SAVR6: 0              ;;PUT THE SP HERE
1469 014034 005015     042522 052123  PWRMSG: .ASCII <15><12>/RESTARTED FROM PWR FAIL/
      014042 051101     042524 020104
      014050 051106     046517 050040
      014056 051127     043040 044501
      014064 000114
1470 .EVEN
1471 .SBITL TRAP DECODER
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 014066 010046     $TRAP: MOV     R0,-(SP)    ;;SAVE R0
(1) 014070 016600     000002     MOV     2(SP),R0        ;;GET TRAP ADDRESS
(1) 014074 005740         TST    -(R0)            ;;BACKUP BY 2
(1) 014076 111000         MOVB  (R0),R0          ;;GET RIGHT BYTE OF TRAP
(1) 014100 006300         ASL   R0                ;;POSITION FOR INDEXING
(1) 014102 016000     014122     MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE

```

```

(1) 014106 000200          RTS      R0          ;;GO TO ROUTINE
(1)
(1)
(1)          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1) 014110 011646          $TRAP2: MOV      (SP),-(SP)      ;;MOVE THE PC DOWN
(1) 014112 016666 000004 000002  MOV      4(SP),2(SP)      ;;MOVE THE PSW DOWN
(1) 014120 000002          RTI          ;;RESTORE THE PSW
(1)
(3)          .SBTTL  TRAP TABLE
(3)
(3)          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)          ;*BY THE "TRAP" INSTRUCTION.
(3)
(3)          :          ROUTINE
(3)          :          -----
(3) 014122 014110          $TRPAD: .WORD  $TRAP2
(3) 014124 010630          $TYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3) 014126 011456          $TYPOC ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 014130 011432          $TYPOS ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 014132 011472          $TYPON ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 014134 011660          $TYPDS ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
(1)
(3) 014136 013006          $GTSWR ;;CALL=GTSWR     TRAP+6(104406)  GET SOFT-SWR SETTING
(1)
(3) 014140 012736          $CKSWR ;;CALL=CKSWR     TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
(3) 014142 013220          $RDCHR ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3) 014144 013350          $RDLIN ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3) 014146 013522          $RDOCT ;;CALL=RDOCT     TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
1472
1473
1474 014150 042522 020107 044524 .SBTTL  ASCII MESSAGES
1474 014156 042515 052517 020124 EM1:   .ASCIZ  /REG TIMEOUT ER/
1475 014164 051105 000          EM2:   .ASCIZ  'REG READ/WRITE ER'
1475 014167 122 043505 051040
1475 014174 040505 027504 051127
1475 014202 052111 020105 051105
1475 014210 000
1476 014211 102 051525 051040 EM3:   .ASCIZ  /BUS RESET ER/
1476 014216 051505 052105 042440
1476 014224 000122
1477 014226 047106 052103 041040 EM4:   .ASCIZ  /FNCT BITS FAILED TO SET STAT BITS/
1477 014234 052111 020123 040506
1477 014242 046111 042105 052040
1477 014250 020117 042523 020124
1477 014256 052123 052101 041040
1477 014264 052111 000123
1478 014270 042522 042101 020131 EM5:   .ASCIZ  /READY INTR FAILURE/
1478 014276 047111 051124 043040
1478 014304 044501 052514 042522
1478 014312 000
1479 014313 122 040505 054504 EM6:   .ASCIZ  /READY CLR OR SET ER/
1479 014320 041440 051114 047440
1479 014326 020122 042523 020124
1479 014334 051105 000
1480 014337 123 040524 052524 EM7:   .ASCIZ  /STATUS ER ON XFER/
  
```

	014344	020123	051105	047440		
	014352	020116	043130	051105		
	014360	000				
1481	014361	127	051117	020104	EM10:	.ASCIZ /WORD COUNT ER ON XFER/
	014366	047503	047125	020124		
	014374	051105	047440	020116		
	014402	043130	051105	000		
1482	014407	102	043125	042506	EM11:	.ASCIZ /BUFFER ADRS ER ON XFER/
	014414	020122	042101	051522		
	014422	042440	020122	047117		
	014430	054040	042506	000122		
1483	014436	040504	040524	042440	EM12:	.ASCIZ /DATA ER FROM MEM/
	014444	020122	051106	046517		
	014452	046440	046505	000		
1484	014457	104	052101	020101	EM13:	.ASCIZ /DATA ER TO MEM/
	014464	051105	052040	020117		
	014472	042515	000115			
1485	014476	044523	043516	042514	EM14:	.ASCIZ /SINGLE CYCLE OFF DID NOT LOCK OUT CPU/
	014504	041440	041531	042514		
	014512	047440	043106	042040		
	014520	042111	047040	052117		
	014526	046040	041517	020113		
	014534	052517	020124	050103		
	014542	000125				
1486	014544	044523	043516	042514	EM15:	.ASCIZ /SINGLE CYCLE ON LOCKED OUT CPU/
	014552	041440	041531	042514		
	014560	047440	020116	047514		
	014566	045503	042105	047440		
	014574	052125	041440	052520		
	014602	000				
1487	014603	015	050012	042514	WARN:	.ASCII <15><12>/PLEASE DISABLE 'REV11' MEMORY REFRESH OPTION/
	014610	051501	020105	044504		
	014616	040523	046102	020105		
	014624	051042	053105	030461		
	014632	020042	042515	047515		
	014640	054522	051040	043105		
	014646	042522	044123	047440		
	014654	052120	047511	116		
1488	014661	015	040412	042116		.ASCIZ <15><12>/AND ENABLE PROCESSOR MEMORY REFRESH /
	014666	042440	040516	046102		
	014674	020105	051120	041517		
	014702	051505	047523	020122		
	014710	042515	047515	054522		
	014716	051040	043105	042522		
	014724	044123	020040	000040		
1489	014732	042516	020130	047514	EM16:	.ASCIZ /NEX LOGIC ER/
	014740	044507	020103	051105		
	014746	000				
1490	014747	103	041531	042514	EM17:	.ASCIZ /CYCLE FAILED TO CLK DBR (IN)/
	014754	043040	044501	042514		
	014762	020104	047524	041440		
	014770	045514	042040	051102		
	014776	024040	047111	000051		
1491	015004	040504	040524	042440	EM20:	.ASCIZ "DATA ER FROM I/O PAGE (XCSR)"
	015012	020122	051106	046517		
	015020	044440	047457	050040		

```

1492 015026 043501 020105 054050
      015034 051503 024522 000
      015041 105 051122 041520 DH1: .ASCIZ /ERRPC TSTNUM BUSADR EXPCT RCVD/
      015046 020040 052040 052123
      015054 052516 020115 041040
      015062 051525 042101 020122
      015070 042440 050130 052103
      015076 020040 051040 053103
      015104 000104
1493 015106 051105 050122 020103 DH2: .ASCIZ /ERRPC TSTNUM BUSADR ADRS EXPCT RCVD/
      015114 020040 051524 047124
      015122 046525 020040 052502
      015130 040523 051104 020040
      015136 042101 051522 020040
      015144 020040 054105 041520
      015152 020124 020040 041522
      015160 042126 000
1494 015163 105 051122 041520 DH3: .ASCIZ /ERRPC TSTNUM BUSADR/
      015170 020040 052040 052123
      015176 052516 020115 041040
      015204 051525 042101 000122
1495
1496
1497 015212 001116 001534 001122 DT1: .EVEN
      015220 001124 001126 000000 $ERRPC,TSTNUM,$BDADR,$GDDAT,$BDDAT,0
1498 015226 001116 001534 001122 DT2: $ERRPC,TSTNUM,$BDADR,$GDADR,$GDDAT,$BDDAT,0
      015234 001120 001124 001126
      015242 000000
1499 015244 001116 001534 001122 DT3: $ERRPC,TSTNUM,$BDADR,0
      015252 000000
1500 015254 005015 051511 052040 KDF11B: .ASCIZ <15><12>/IS THE PROCESSOR KDF11-B? /
      015262 042510 050040 047522
      015270 042503 051523 051117
      015276 045440 043104 030461
      015304 041055 020077 000040
1501 015312 005015 051124 047101 IOTEST: .ASCIZ <15><12>.TRANSFERS FOR I/O PAGE? .
      015320 043123 051105 020123
      015326 047506 020122 027511
      015334 020117 040520 042507
      015342 020077 000040
1502 015346 005015 042101 051104 IOADR: .ASCIZ <15><12>\ADDRESS OF I/O PAGE? \
      015354 051505 020123 043117
      015362 044440 047457 050040
      015370 043501 037505 020040
      015376 000
1503 015400 000000
1504 000001
1505
1506
1507
1508 015400 000000
1509 000001

```

```

.EVEN
:*****
: 'DBUF' IS THE WORKING AREA IN EACH 4K MEM FOR ALL
: NPR OPERATIONS - IT IS 200 WORDS LONG
:*****
DBUF: 0 ;1ST ADRS OF DATA BUFFER
.END

```

ABASE = 172410	377#	390			
ACDW1 = 000000	390				
ACDW2 = 000000	390				
ACPUOP= 000000	390				
ADDW0 = 000000	390				
ADDW1 = 000000	390				
ADDW10= 000000	390				
ADDW11= 000000	390				
ADDW12= 000000	390				
ADDW13= 000000	390				
ADDW14= 000000	390				
ADDW15= 000000	390				
ADDW2 = 000000	390				
ADDW3 = 000000	390				
ADDW4 = 000000	390				
ADDW5 = 000000	390				
ADDW6 = 000000	390				
ADDW7 = 000000	390				
ADDW8 = 000000	390				
ADDW9 = 000000	390				
ADEVCT= 000000	390				
ADEVM = 000001	379#	390			
AENV = 000000	390				
AENVM = 000000	390				
AFATAL= 000000	390				
ALTERN 007452	1199	1240#			
AMADR1= 000000	390				
AMADR2= 000000	390				
AMADR3= 000000	390				
AMADR4= 000000	390				
AMAMS1= 000000	390				
AMAMS2= 000000	390				
AMAMS3= 000000	390				
AMAMS4= 000000	390				
AMSGAD= 000000	390				
AMSGLG= 000000	390				
AMSGTY= 000000	390				
AMTYP1= 000000	390				
AMTYP2= 000000	390				
AMTYP3= 000000	390				
AMTYP4= 000000	390				
APASS = 000000	390				
APRIOR= 000000	390				
APTC SU= 000040	1451	1452#	1455		
APTENV= 000001	1451	1452#			
APTSIZ= 000200	514	1452#			
APTSPO= 000100	1451	1452#			
ASWREG= 000000	390				
ATESTN= 000000	390				
AUNIT = 000000	390				
AUSWR = 000000	390				
AVECT1= 000124	378#	390			
AVECT2= 000000	390				
BIT0 = 000001	376#				
BIT00 = 000001	376#	610	615	664	1344
BIT01 = 000002	376#				















\$SETUP=	000117	512#	514	528	1295	1455	1465	1466										
\$STUP =	177777	512#																
\$SVLAD	012654	1465#																
\$SVPC =	000106	387#																
\$SWR =	167400	371#	372#	375	390	514	568	586	603	623	640	653	673	698				
		708	722	737	762	778	794	811	855	899	933	963	992	1024				
		1067	1123	1157	1197	1295	1455	1465	1468									
\$SWREG	001216	390#	514															
\$SWRMK=	000300	373#	375	1465														
\$TESTN	001200	390#	568*	1465*														
\$TIMES	001160	390#	514*	653*	673*	698*	737*	899*	933*	963*	992*	1024*	1067*	1123*				
		1157*	1295*	1465*														
\$TKB	001146	390#	1451	1466														
\$TKS	001144	390#	1451	1466*														
\$TN =	000032	371#	374#	568#	586#	603#	619	623#	640#	650	653#	669	673#	698#				
		705	708#	722#	737#	762#	775	778#	794#	808	811#	855#	899#	933#				
		963#	992#	1024#	1067#	1123#	1157#	1197#										
\$TPB	001152	390#	1451*															
\$TPFLG	001157	390#	1451															
\$TPS	001150	390#	1203	1204	1226	1231	1451											
\$TRAP	014066	514	1471#															
\$TRAP2	014110	1471#																
\$TRP =	000013	1471#																
\$TRPAD	014122	1471#																
\$TSTM	001004	389#																
\$TSTM	001102	390#	569*	1295*	1455	1460	1465*											
\$TTYIN	013456	1466#																
\$TYPBN=	***** U	1471																
\$TYPDS	011660	1454#	1471															
\$TYPE	010630	1451#	1452	1471														
\$TYPEC	011042	1451#	1466															
\$TYPEX	011162	1451#																
\$TYPOC	011456	1453#	1471															
\$TYPON	011472	1453#	1471															
\$TYPOS	011432	1453#	1471															
\$UNIT	001206	390#	554*	563	564	1291*												
\$UNITM	001010	389#																
\$USWR	001220	390#																
\$VECT1	001244	390#	522															
\$VECT2	001246	390#																
\$XOFF =	000023	1451																
\$XON =	000021	1451	1466															
\$XTSTR	012462	1465#																
\$GET4=	000000	1295#																
\$OFIL	011655	1453#*																
\$4OCAT=	***** U	1455	1465															
.	= 015402	382#	383#	387#	388#	389#	390#	514	528#	1295	1451	1452#	1454#	1455				
		1464#	1465	1466#	1467	1468	1503#											
.\$ASTA=	***** U	1452																
.\$X =	001000	389#																

. ABS. 015402 000

ERRORS DETECTED: 0

CVDRAB, CVDRAB/CRF:SYM/NL:TOC=CVDRAB.P11  
RUN-TIME: 19 9 .9 SECONDS  
RUN-TIME RATIO: 116/29=3.8  
CORE USED: 27K (54 PAGES)