

# DRV11B

DIAGNOSTIC TEST  
MD-11-DVDRA-A

EP-DVDRA-A-DL-A  
COPYRIGHT © 1976  
FICHE 1 OF 1

NOV 1976  
**digital**  
MADE IN USA

The image displays a grid of 60 small diagnostic test charts, arranged in 10 rows and 6 columns. Each chart contains various data points, including binary strings, numerical values, and graphical waveforms. The charts are organized into several sections, with some containing headers like 'TEST 1', 'TEST 2', 'TEST 3', 'TEST 4', 'TEST 5', 'TEST 6', 'TEST 7', 'TEST 8', 'TEST 9', and 'TEST 10'. The data is presented in a structured format, often with columns of binary digits (0s and 1s) and rows of numerical values. Some charts also feature graphical waveforms or plots. The overall layout is a dense grid of diagnostic information.

801

.REM!

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DVDR-A  
PRODUCT NAME: DRV11B DMA INTERFACE DIAGNOSTIC  
DATE: OCTOBER 1976  
MAINTAINER: DIAGNOSTIC ENGINEERING

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

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

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

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

TABLE OF CONTENTS

- 1.0 ABSTRACT
- 2.0 REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 STORAGE
- 3.0 LOADING PROCEDURE
- 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 DRVIIB BUS & VECTOR ADDRESS MODIFICATION
  - 7.2 XXDP/APT NOTES
  - 7.3 POWER FAIL
  - 7.4 MULTIPLE DRVIIB 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

10/11/76 15:39:39 11-OCT-76 15:39:39

1.0 ABSTRACT  
-----

THE DRV118 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. PDP11/03 COMPUTER OR LSI-11 PROCESSOR
2. DLV11 WITH I/O TYPE TERMINAL
3. DRV118 WITH LOOP BACK CABLE

## 2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING 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 (177550-LOW OR 177550-HIGH) AND CHARACTER 'L'.
4. AFTER TAPE IS LOADED, LOAD THE DRV118 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 DRV118 BINARY TAPE IN THE APPROPRIATE READER).

4.0 STARTING PROCEDURE  
-----

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 OPTIONS.

## 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 (7-3)

### 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
EXPT	DATA THAT WAS EXPECTED
RCVD	DATA THAT WAS RECEIVED
ADRS	MEMORY ADDRESS OF DATA TRANSFER ON ERROR

\*ALWAYS REPORTED

11-11-76 15:39 MACY:1 27(665) 11-OCT-76 15:39 PAGE 4  
 XEROX-11-DVCR-4 DRIVE DMA INTERFACE DIAGNOSTIC  
 A.P.11

11-0000-11-DVDRA-2 DRV11B DMA INTERFACE DIAGNOSTIC  
11-0000-11-P11

7.0 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 BK OR MCRE).  
THIS DIAGNOSTIC DOES SUPPORT "APT" AND HAS RUN UNDER IT.

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 '\$DEVN' 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  
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.

0000  
0001  
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041  
0042  
0043  
0044  
0045  
0046  
0047  
0048  
0049  
0050  
0051  
0052  
0053  
0054  
0055  
0056  
0057  
0058  
0059  
0060  
0061  
0062  
0063  
0064  
0065  
0066  
0067  
0068  
0069  
0070  
0071  
0072  
0073  
0074  
0075  
0076  
0077  
0078  
0079  
0080  
0081  
0082  
0083  
0084  
0085  
0086  
0087  
0088  
0089  
0090  
0091  
0092  
0093  
0094  
0095  
0096  
0097  
0098  
0099  
0100

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 DRV118 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 I/O PTRN
2. SINGLE 'DATO' XFER - FLOATING I/O PTRN
3. 200 'DATI' XFERS - FLOATING I/O PTRN
4. 200 'DATO' XFERS - FLOATING I/O PTRN
5. SINGLE 'DATI' XFER TO THE TTY PRINTER CSR

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



374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403

-----  
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

40  
41  
42  
43  
44  
45  
46  
47  
48  
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  
99  
00

## 1.0 ABSTRACT

THE DRV118 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. PDP11/03 COMPUTER OR LSI-11 PROCESSOR
2. DLV11 WITH I/O TYPE TERMINAL
3. DRV118 WITH LOOP BACK CABLE

### 2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

## 3.0 LOADING 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 DRV118 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 DRV118 BINARY TAPE IN THE APPROPRIATE READER).

## 4.0 STARTING PROCEDURE

1. MAKE SURE THE MAINTENANCE LOOP BACK CABLE IS INSERTED IN THE I/O CONNECTORS ON THE M7990 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 OPTIONS.

458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510

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 <7-0>

5.2 CONTROL

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

\*ALWAYS REPORTED

511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600

7.0 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.

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 '\$DEVN' 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  
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.

554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
6079.0 PROGRAM TEST DESCRIPTIONS  
-----

## 9.1 GENERAL

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE DRV118 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 I/O PTRN
2. SINGLE 'DATO' XFER - FLOATING I/O PTRN
3. 200 'DATI' XFERS - FLOATING I/O PTRN
4. 200 'DATO' XFERS - FLOATING I/O PTRN
5. SINGLE 'DATI' XFER TO THE TTY PRINTER CSR

## 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 &amp; 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

608  
 609  
 610  
 611  
 612  
 613  
 614  
 615  
 616  
 617  
 618  
 619  
 620  
 621  
 622  
 623  
 624  
 625  
 626  
 627  
 628  
 629  
 630  
 631  
 632  
 633  
 634  
 635  
 636  
 637  
 638  
 639  
 640  
 641  
 642  
 643  
 644  
 645  
 646  
 647  
 648  
 649  
 650  
 651  
 652  
 653  
 654  
 655  
 656  
 657  
 658  
 659  
 660  
 661

000001  
 160000  
 167400  
 000001

001100

000011  
 000012  
 000015  
 000200  
 177776  
 177774  
 177772  
 177570  
 177570

000000  
 000001  
 000002  
 000003

10.0 LISTING

```

!
.TITLE MAINDEC-11-DVDRA-A DRV118 DMA INTERFACE DIAGNOSTIC
;*COPYRIGHT (C) 1976
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY R. MOORE
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-CD),MAR 21, 1976.
;*
$TN=1
$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
$SWR=167400
$TN=1
.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;* SWITCH USE
;* -----
;* 15 HALT ON ERROR
;* 14 LOOP ON TEST
;* 13 INHIBIT ERROR TYPEOUTS
;* 11 INHIBIT ITERATIONS
;* 10 BELL ON ERROR
;* 9 LOOP ON ERROR
;* 8 LOOP ON TEST IN SWR<7:0>
.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS
HT= 11 ;;CODE FOR HORIZONTAL TAB
LF= 12 ;;CODE FOR LINE FEED
CR= 15 ;;CODE FOR CARRIAGE RETURN
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570 ;;HARDWARE SWITCH REGISTER
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0 ;;GENERAL REGISTER
R1= %1 ;;GENERAL REGISTER
R2= %2 ;;GENERAL REGISTER
R3= %3 ;;GENERAL REGISTER

```

662 000004  
663 000005  
664 000006  
665 000007  
666  
667  
668  
669  
670 000000  
671 000040  
672 000100  
673 000140  
674 000200  
675 000240  
676 000300  
677 000340  
678  
679  
680 100000  
681 040000  
682 020000  
683 010000  
684 004000  
685 002000  
686 001000  
687 000400  
688 000200  
689 000100  
690 000040  
691 000020  
692 000010  
693 000004  
694 000002  
695 000001  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708 100000  
709 040000  
710 020000  
711 010000  
712 004000  
713 002000  
714 001000  
715 000400

R4= %4 ;; GENERAL REGISTER  
R5= %5 ;; GENERAL REGISTER  
R6= %6 ;; GENERAL REGISTER  
R7= %7 ;; GENERAL REGISTER  
.EQUIV R6,SP ;; STACK POINTER  
.EQUIV R7,PC ;; PROGRAM COUNTER

.\*PRIORITY LEVEL DEFINITIONS  
PR0= 0 ;; PRIORITY LEVEL 0  
PR1= 40 ;; PRIORITY LEVEL 1  
PR2= 100 ;; PRIORITY LEVEL 2  
PR3= 140 ;; PRIORITY LEVEL 3  
PR4= 200 ;; PRIORITY LEVEL 4  
PR5= 240 ;; PRIORITY LEVEL 5  
PR6= 300 ;; PRIORITY LEVEL 6  
PR7= 340 ;; PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS  
SW15= 100000  
SW14= 40000  
SW13= 20000  
SW12= 10000  
SW11= 4000  
SW10= 2000  
SW09= 1000  
SW08= 400  
SW07= 200  
SW06= 100  
SW05= 40  
SW04= 20  
SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1  
.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)  
BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400

```

716      000200      BIT07= 200
717      000100      BIT06= 100
718      000040      BIT05= 40
719      000020      BIT04= 20
720      000010      BIT03= 10
721      000004      BIT02= 4
722      000002      BIT01= 2
723      000001      BIT00= 1
724      .EQUIV      BIT09,BIT9
725      .EQUIV      BIT08,BIT8
726      .EQUIV      BIT07,BIT7
727      .EQUIV      BIT06,BIT6
728      .EQUIV      BIT05,BIT5
729      .EQUIV      BIT04,BIT4
730      .EQUIV      BIT03,BIT3
731      .EQUIV      BIT02,BIT2
732      .EQUIV      BIT01,BIT1
733      .EQUIV      BIT00,BIT0
734
735      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
736      000004      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
737      000010      RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
738      000014      TBITVEC=14        ;; "T" BIT
739      000014      TRTVEC= 14         ;; TRACE TRAP
740      000014      BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
741      000020      IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
742      000024      PWRVEC= 24         ;; POWER FAIL
743      000030      EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
744      000034      TRAPVEC=34        ;; "TRAP" TRAP
745      000060      TKVEC= 60           ;; TTY KEYBOARD VECTOR
746      000064      TPVEC= 64          ;; TTY PRINTER VECTOR
747      000240      PIRQVEC=240        ;; PROGRAM INTERRUPT REQUEST VECTOR
748      172410      ABASE= 172410      ;; BASE DRV118 BUS ADRS EQUATE
749      000124      AVECT1= 000124     ;; BASE DRV118 VECTOR ADRS EQUATE -
750      000001      ADEVM= 1           ;; DEFAULT TO ONE DRV118
751      106427      MTPS=106427        ;; INSTR EQUATE THAT MOVES BYTE TO PSW
752      .SBTTL      TRAP CATCHER
753
754      000000      .=0
755      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
756      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
757      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
758      000174      .=174
759      000174      000000      DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
760      000176      000000      SWREG: .WORD 0           ;; SOFTWARE SWITCH REGISTER
761      .SBTTL      STARTING ADDRESS(ES)
762      000200      000137      001544      JMP @#START ;; JUMP TO STARTING ADDRESS OF PROGRAM
763      000100      000100      .=100
764      000100      000104      000200      000002      .WORD 104,200,2          ;; IF 'B EVENT' ON Q BUS IS CONNECTED
765      ;; IGNORE IT'S INTERRUPT - JUST DO A RTI
    
```

766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798

000046 000106  
000046 000046  
000046 007546  
000052 000052  
000052 000000  
001006 000106  
001000 001000  
  
001000 001000  
000024 000024  
000024 000200  
000044 000044  
000044 001000  
001000 001000  
  
001000 001000  
001000 000000  
001002 001174  
001004 000031  
001006 000006  
001010 000144  
001012 000052

.SBTTL ACT11 HOOKS

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

:HOOKS REQUIRED BY ACT11

SSVPC=. ;SAVE PC

=46

SENDAD ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SECP

=52

.WORD 0 ;;2)SET LOC.52 TO ZERO

SSVPC ;; RESTORE PC

=1000

.SBTTL APT PARAMETER BLOCK

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

:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

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

.SX=. ;;SAVE CURRENT LOCATION

=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM

200 ;;FOR APT START UP

=44 ;;POINT TO APT INDIRECT ADDRESS PNTR.

SAPTHDR ;;POINT TO APT HEADER BLOCK

=.SX ;;RESET LOCATION COUNTER

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

:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC

:INTERFACE SPEC.

SAPTHD:

\$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.

\$MBADR: .WORD \$MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)

\$TSTM: .WORD 25. ;;RUN TIM OF LONGEST TEST

\$PASTM: .WORD 6. ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)

\$UNITM: .WORD 100. ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT

.WORD \$ETENC-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

.SBTTL COMMON TAGS

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

799  
800  
801  
802  
803  
804  
805 001100  
806 001100 000000  
807 001100 000000  
808 001102 000  
809 001103 000  
810 001104 000000  
811 001106 000000  
812 001110 000000  
813 001112 000000  
814 001114 000  
815 001115 001  
816 001116 000000  
817 001120 000000  
818 001122 000000  
819 001124 000000  
820 001126 000000  
821 001130 000000  
822 001132 000000  
823 001134 000  
824 001135 000  
825 001136 000000  
826 001140 177570  
827 001142 177570  
828 001144 177550  
829 001146 177562  
830 001150 177564  
831 001152 177566  
832 001154 000  
833 001155 002  
834 001156 012  
835 001157 000  
836 001160 000000  
837 001162 000000  
838 001164 177607 000377  
839 001170 077  
840 001171 015  
841 001172 000012  
842  
843  
844  
845  
846  
847 001174  
848 001174 000000  
849 001176 000000  
850 001200 000000  
851 001202 000000  
852 001204 000000

SCMTAG: =1100  
: START OF COMMON TAGS  
\$TSTNM: .WORD 0 : CONTAINS THE TEST NUMBER  
\$ERFLG: .BYTE 00000000 : CONTAINS ERROR FLAG  
\$ICNT: .WORD 00000000 : CONTAINS SUBTEST ITERATION COUNT  
\$LPADR: .WORD 00000000 : CONTAINS SCOPE LOOP ADDRESS  
\$LPERR: .WORD 00000000 : CONTAINS SCOPE RETURN FOR ERRORS  
\$ERTTL: .WORD 00000000 : CONTAINS TOTAL ERRORS DETECTED  
\$ITEMB: .BYTE 00100000 : CONTAINS ITEM CONTROL BYTE  
\$ERMAX: .BYTE 00100000 : CONTAINS MAX. ERRORS PER TEST  
\$ERRPC: .WORD 00000000 : CONTAINS PC OF LAST ERROR INSTRUCTION  
\$GDADR: .WORD 00000000 : CONTAINS ADDRESS OF 'GOOD' DATA  
\$BDADR: .WORD 00000000 : CONTAINS ADDRESS OF 'BAD' DATA  
\$GDAT: .WORD 00000000 : CONTAINS 'GOOD' DATA  
\$BDAT: .WORD 00000000 : CONTAINS 'BAD' DATA  
: RESERVED--NOT TO BE USED  
\$AUTOB: .BYTE 00000000 : AUTOMATIC MODE INDICATOR  
\$INTAG: .BYTE 00000000 : INTERRUPT MODE INDICATOR  
\$SWR: .WORD 054R : ADDRESS OF SWITCH REGISTER  
\$DISPLAY: .WORD 0015F : ADDRESS OF DISPLAY REGISTER  
\$TKS: 177560 : TTY KBD STATUS  
\$TKB: 177562 : TTY KBD BUFFER  
\$TPS: 177564 : TTY PRINTER STATUS REG. ADDRESS  
\$TPB: 177566 : TTY PRINTER BUFFER REG. ADDRESS  
\$NULL: .BYTE 0 : CONTAINS NULL CHARACTER FOR FILLS  
\$FILLS: .BYTE 2 : CONTAINS # OF FILLER CHARACTERS REQUIRED  
\$FILLC: .BYTE 12 : INSERT FILL CHARS. AFTER A "LINE FEED"  
\$TFPLG: .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: : APT MAILBOX  
\$MSGTY: .WORD ANSGTY : MESSAGE TYPE CODE  
\$FATAL: .WORD AFATAL : FATAL ERROR NUMBER  
\$TESTN: .WORD ATESTN : TEST NUMBER  
\$PASS: .WORD APASS : PASS COUNT  
\$DEVCT: .WORD ADEVCT : DEVICE COUNT



602

MAINDEC-11-DVDR-A DRV:19 DMA INTERFACE DIAGNOSTIC  
OVDRAA.P11 RPT MAILBOX-ETABLE

MACY11 27(665) 11-OCT-76 15:39 PAGE 19

907  
908  
909 001320  
910

SETEND:

.SBTT\_ ERROR POINTER TABLE

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

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

911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960

001320  
001320 013372  
001322 014263  
001324 014434  
001326 000000  
  
001330 013411  
001332 014263  
001334 014434  
001336 000000  
  
001340 013433  
001342 014263  
001344 014434  
001346 000000  
  
001350 013450  
001352 014263  
001354 014434  
001356 000000  
  
001360 013512  
001362 014263  
001364 014434  
001366 000000  
  
001370 013535  
001372 014263  
001374 014434  
001376 000000

\$ERRTB:  
;ERROR 1  
EM1 :REG TIMEOUT ER  
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT  
0  
  
;ERROR 2  
EM2 :REG READ/WRITE ER  
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT  
0  
  
;ERROR 3  
EM3 :BUS RESET ER  
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT  
0  
  
;ERROR 4  
EM4 :FNCT BITS FAILED TO SET STAT BITS  
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT  
0  
  
;ERROR 5  
EM5 :READY INTR FAILURE  
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT  
0  
  
;ERROR 6  
EM6 :READY CLR OR SET ER  
DH1 :ERRPC TSTNUM BUSADR EXPCT RCVD  
DT1 :\$ERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT  
0

961			:ERROR	7							
962	001400	013561		EM7						:STATUS ER ON XFER	
963	001402	014263		DH1						:ERRPC TSTNUM BUSADR EXPCT RCVD	
964	001404	014434		DT1						:SERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
965	001406	000000		0							
966											
967			:ERROR	10							
968	001410	013603		EM10						:WORD COUNT ER ON XFER	
969	001412	014263		DH1						:ERRPC TSTNUM BUSADR EXPCT RCVD	
970	001414	014434		DT1						:SERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
971	001416	000000		0							
972											
973			:ERROR	11							
974	001420	013631		EM11						:BUFFER ADRS ER ON XFER	
975	001422	014263		DH1						:ERRPC TSTNUM BUSADR EXPCT RCVD	
976	001424	014434		DT1						:SERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
977	001426	000000		0							
978											
979			:ERROR	12							
980	001430	013660		EM12						:DATA ER FROM MEM	
981	001432	014330		DH2						:ERRPC TSTNUM BUSADR ADRS EXPCT RCVD	
982	001434	014450		DT2						:SERRPC TSTNUM \$BDADR \$GDADR \$GDDAT \$BDDAT	
983	001436	000000		0							
984											
985			:ERROR	13							
986	001440	013701		EM13						:DATA ER TO MEM	
987	001442	014330		DH2						:ERRPC TSTNUM BUSADR ADRS EXPCT RCVD	
988	001444	014450		DT2						:SERRPC TSTNUM \$BDADR \$GDADR \$GDDAT \$BDDAT	
989	001446	000000		0							
990											
991			:ERROR	14							
992	001450	013720		EM14						:SINGLE CYCLE OFF DID NOT LOCK OUT CPU	
993	001452	014405		DH3						:ERRPC TSTNUM BUSADR	
994	001454	014466		DT3						:SERRPC TSTNUM \$BDADR	
995	001456	000000		0							
996											
997			:ERROR	15							
998	001460	013766		EM15						:SINGLE CYCLE ON LOCKED OUT CPU	
999	001462	014405		DH3						:ERRPC TSTNUM BUSADR	
1000	001464	014466		DT3						:SERRPC TSTNUM \$BDADR	
1001	001466	000000		0							
1002											
1003			:ERROR	16							
1004	001470	014154		EM16						:NEX LOGIC ER	
1005	001472	014263		DH1						:ERRPC TSTNUM BUSADR EXPCT RCVD	
1006	001474	014434		DT1						:SERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
1007	001476	000000		0							
1008											
1009			:ERROR	17							
1010	001500	014171		EM17						:CYCLE FAILED TO CLK DBR (IN)	
1011	001502	014263		DH1						:ERRPC TSTNUM BDADR GDDAT BDDAT	
1012	001504	014434		DT1						:SERRPC TSTNUM \$BDADR \$GDDAT \$BDDAT	
1013	001506	000000		0							

```

1014                                     ;ERROR 20
1015 001510 014226                       EM20
1016 001512 014330                       DH2
1017 001514 014450                       DT2
1018 001516 000000                       0
1019
1020
1021                                     ;DRV118 BUS REGISTER ADDRESS POINTERS
1022
1023 001520 172410                       DRVWCR: 172410
1024 001522 172412                       DRVBAR: 172412
1025 001524 172414                       DRVCSR: 172414
1026 001526 172416                       DRVDBR: 172416
1027
1028                                     ;DRV118 VECTOR ADDRESS POINTERS
1029
1030 001530 000124                       DRVCT0: 124
1031 001532 000126                       DRVCT2: 126
1032
1033                                     ;COMMON PROGRAM LOCATION(S)
1034
1035 001534 000000                       TSTNUM: 0
1036 001536 000001                       DMAP: 1
1037 001540 000000                       CORSZ: 0
1038 001542 014476                       DBUFF: DBUF

```

```

;DATA ER FROM I/O PAGE (XCSR)
;ERRPC TSTNUM BUSADR ADRS EXPCT RCVD
;SERRPC TSTNUM SBDADR SGDADR SGDOAT SBDOAT

```

```

;WORD COUNT
;BUFFER ADDRESS
;COMMAND/STATUS
;DATA BUFFER

```

```

;READY, NEX & INCOMPLETE DATIO VECTOR
;NEW PSW ON INTR

```

```

;CONTAINS TEST NUMBER ON ERROR
;DEVICE MAP - EA BIT SAYS TEST THAT DRV118
;CONTAINS 1ST NON-EXISTANT MEM ADRS
;CONTAINS CURRENT 4K NPR BUFFER ADRS

```

1039						.SBTTL PROGRAM START
1040	001544					START:
1041						.SBTTL INITIALIZE THE COMMON TAGS
1042						::CLEAR THE COMMON TAGS (\$CMTAG) AREA
1043	001544	012706	001100			MOV \$CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
1044	001550	005026				CLR (R6)+ ;:CLEAR MEMORY LOCATION
1045	001552	022706	001140			CMP \$SWR,R6 ;:DONE?
1046	001556	001374				ENE -6 ;:LOOP BACK IF NO
1047	001560	012706	001100			MOV \$STACK,SP ;:SETUP THE STACK POINTER
1048						::INITIALIZE A FEW VECTORS
1049	001564	012737	012064	000020		MOV \$SCOPE,\$IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
1050	001572	012737	000340	000022		MOV \$340,\$IOTVEC+2 ;:LEVEL 7
1051	001600	012737	011522	000030		MOV \$ERROR,\$EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
1052	001606	012737	000340	000032		MOV \$340,\$EMTVEC+2 ;:LEVEL 7
1053	001614	012737	013326	000034		MOV \$STRAP,\$TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
1054	001622	012737	000340	000036		MOV \$340,\$TRAPVEC+2 ;:LEVEL 7
1055	001630	012737	013122	000034		MOV \$SFWRDN,\$PWRVEC ;:POWER FAILURE VECTOR
1056	001636	012737	000340	000026		MOV \$340,\$PWRVEC+2 ;:LEVEL 7
1057	001644	005037	001160			CLR \$TIMES ;:INITIALIZE NUMBER OF ITERATIONS
1058	001650	005037	001162			CLR \$ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
1059	001654	112737	000001	001115		MOVB #1,\$ERR1X ;:ALLOW ONE ERROR PER TEST
1060	001662	012737	001662	001106		MOV #,\$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
1061	001670	012737	001670	001110		MOV #,\$SLPERP ;:SETUP THE ERROR LOOP ADDRESS
1062						::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1063						::EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1064	001676	013746	000004			MOV \$ERRVEC,-(SP) ;:SAVE ERROR VECTOR
1065	001702	012737	001736	000004		MOV \$64,\$ERRVEC ;:SET UP ERROR VECTOR
1066	001710	012737	177570	001140		MOV \$DSWR,\$SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
1067	001716	012737	177570	001142		MOV \$DDISP,\$DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
1068	001724	022777	177777	177206		CMP #-1,\$SWR ;:TRY TO REFERENCE HARDWARE SWR
1069	001732	001012				BNE 66\$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
1070						::AND THE HARDWARE SWR IS NOT = -1
1071	001734	000403				BR 65\$ ;:BRANCH IF NO TIMEOUT
1072	001736	012716	001744		64\$:	MOV \$65\$,(SP) ;:SET UP FOR TRAP RETURN
1073	001742	000002				RTI
1074	001744	012737	000176	001140	65\$:	MOV \$SWREG,\$SWR ;:POINT TO SOFTWARE SWR
1075	001752	012737	000174	001142		MOV \$DISPREG,\$DISPLAY
1076	001760	012637	000004		66\$:	MOV (SP)+,\$ERRVEC ;:RESTORE ERROR VECTOR
1077						
1078	001764	005037	001202			CLR \$PASS ;:CLEAR PASS COUNT
1079	001770	132737	000200	001215		BITB \$APTSIZE,\$ENVM ;:TEST USER SIZE UNDER APT
1080	001776	001403				BEG 67\$ ;:YES,USE NON-APT SWITCH
1081	002000	012737	001216	001140		MOV \$SSWREG,\$SWR ;:NO,USE APT SWITCH REGISTER
1082	002006				67\$:	
1083	002006	012700	001520			START1: MOV \$DRVWCR,R0 ;:SET UP REG ADRS POINTERS
1084	002012	013701	001250			MOV \$BASE,R1 ;:GET BASE ADRS
1085	002016	010120				SETUP2: MOV R1,(R0)+ ;:LOAD EM
1086	002020	062701	000002			ADD #2,R1
1087	002024	022700	001530			CMP \$DRVDBR+2,R0 ;:ALL DONE?
1088	002030	001372				BNE SETUP2 ;:BR IF NOT
1089	002032	012700	001530			MOV \$DRVCTO,R0 ;:SET UP DRV118 VECTOR ADRS POINTER
1090	002036	013701	001244			MOV \$VECT1,R1 ;:GET BASE VECTOR ADRS
1091	002042	042701	170000			BIC #170000,R1 ;:CLR OUT PRIORITY BITS
1092	002046	010120				SETUP3: MOV R1,(R0)+ ;:

```

1093 002050 062701 000002      ADD      #2,R1      ;POINT TO NEXT
1094 002054 022700 001534      CMP      #DRVCT2+2,R0 ;ALL DONE?
1095 002060 001372              BNE      SETUP3    ;BR IF NOT
1096                          .SBTTL  TYPE PROGRAM NAME
1097                          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1098 002062 005227 177777      INC      #-1       ;;FIRST TIME?
1099 002066 001052              BNE      64$       ;;BRANCH IF NO
1100 002070 104400 002136      TYPE     65$       ;;TYPE ASCIZ STRING
1101                          .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1102 002074 005737 000042      TST     2#42       ;;ARE WE RUNNING UNDER XXDP/ACT?
1103 002100 001012              BNE      66$       ;;BRANCH IF YES
1104 002102 123727 001214 000001  CMPB    $ENV,#1    ;;ARE WE RUNNING UNDER APT?
1105 002110 001406              BEQ     66$       ;;BRANCH IF YES
1106 002112 023727 001140 000176  CMP     SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
1107 002120 001005              BNE     67$       ;;BRANCH IF NO
1108 002122 104405              GTSWR                ;;GET SOFT-SWR SETTINGS
1109 002124 000403              BR      67$
1110 002126 112737 000001 001134 66$:   MOVB    #1,$AUTOB  ;;SET AUTO-MODE INDICATOR
1111 002134              67$:
1112 002134 000427              BR      64$       ;;GET OVER THE ASCIZ
1113                          ;;65$: .ASCIZ <CRLF>#MD-11-DVDRA-A DRV11B DMA INTERFACE DIAG #<CRLF>
1114                          64$:
1115 002214 005737 001540      TST     CORSZ      ;TEST IF FIRST PASS
1116 002220 001002              BNE     CORSZR     ;BR IF NOT
1117 002222 104400 014025      TYPE    WARN       ;TELL THE OPERATOR TO TURN OFF DMA REFRESH
1118                          ;;*****
1119                          ;;LET'S SEE HOW MUCH MEM WE HAVE
1120                          ;;*****
1121 002226 012700 020000 000004  CORSZR: MOV    #20000,R0 ;USE R0 TO LOOK
1122 002232 012737 002244 000004  MOV    #25,2#ERRVEC ;SET UP TIME OUT RETURN ADRS
1123 002240 005720              1$:   TST    (R0)+    ;TAKE A LOOK
1124 002242 000776              2$:   BR     1$      ;UNTIL TIMEOUT
1125 002244 042700 017777      BIC    #17777,R0   ;POINT TO 1ST NON-EXSISTANT 4K BLK
1126 002250 010037 001540      MOV    R0,CORSZ   ;SAVE FOR LATER
1127 002254 012737 000006 000004  MOV    #ERRVEC+2,2#ERRVEC ;RESTORE VECTOR
1128 002262 012737 014476 001542  MOV    #DBUF,DBUFP ;INITIALIZE TO LOWEST 4K
1129 002270 012737 000000 001206  MOV    #0,$UNIT    ;SET UP UNIT COUNT
1130 002276 013737 001252 001536  MOV    $DEVN,DMAP  ;GET THE # & POSITION OF DRV11B'S
1131 002304 042737 177400 001536  BIC    #177400,DMAP ;UP TO 8 ONLY
1132 002312 001406              BEQ    RESTRT     ;GO CONTINUE AS IF SOMETHING WAS SELECTED
1133 002314 032737 000001 001536  BIT    #1,DMAP     ;IS 1ST DRV11B SELECTED?
1134 002322 001002              BNE    RESTRT     ;BR IF SO
1135 002324 000137 007364              JMP    NXDEVI     ;NO - GO ADVANCE BASE DRV11B ADDRESSES
1136 002330 106427 000200      RESTRT: MTPS,    200 ;SET PRIORITY TO HIGHEST LEVEL
1137 002334 012706 001100      MOV    #STACK,SP  ;ALWAYS RESET STACK PTR
1138 002340 013737 001206 001204  MOV    $UNIT,$DEVCT ;LOAD APT COUNTER
1139 002346 013700 001206      MOV    $UNIT,R0   ;MAKE AN INDEX
1140 002352 006300              ASL    R0          ;VALUE
1141 002354 013760 001520 001260  MOV    DRVWCR,$DDWO(R0) ;SAVE THE BUS ADDRESS
1142 002362 000705              RESET           ;INITIALIZE DRV11B BEFORE TESTING

```

```

1143      ;*****
1144      ;*TEST 1      TEST THAT ALL DRV118 REGS ARE ACCESSIBLE
1145      ;*****
1146      002364 000240      †ST1:  <NOP>
1147      002366 012737 002402 001106      MOV      #10$, $LPADR      ;; SET SCOPE LOOP ADDRESS
1148      002374 012737 000001 001200      MOV      #1, $TESTN      ;; SET TEST NUMBER IN APT MAIL BOX.
1149      002402 112737 000001 001102      1CS:  MOVVB  #1, $STNM      ;; SET TO TEST #1
1150      002410 012737 002444 001110      MOV      #1$, $LPERR      ;; SET UP SCOPE LOOP ADRS
1151      002416 005037 001124      CLR      $GDDAT      ;; NO DATA COMPARE
1152      002422 005037 001126      CLR      $BDDAT      ;; NO DATA COMPARE
1153      002426 012737 002462 000004      MOV      #2$, @#ERRVEC      ;; SET UP TIMEOUT RETURN ADRS
1154      002434 013700 001520      MOV      DRVWCR, RO      ;; SET UP 1ST DRV11 BUS ADRS
1155      002440 012701 000004      MOV      #4, R1      ;; SET UP REG COUNT
1156      002444 010037 001122      1$:  MOV      RO, $BDADR      ;; SET UP CURRENT DRV BUS ADRS
1157      002450 005710      TST      (RO)      ;; SEE IF THERE
1158      002452 005720      TST      (RO)+      ;; BUMP TO NEXT
1159      002454 005301      DEC      R1      ;; COUNT 4 OF THEM
1160      002456 001403      BEQ      3$      ;; BR IF ALL DONE
1161      002460 000771      BR       1$      ;; TRY NEXT
1162      002462 022626      2$:  CMP      (SP)+, (SP)+      ;; FIX STACK SINCE NO RTI
1163      002464 104001      ERROR   1      ;; BUS ADRS INDICATED DID NOT RESPOND
1164      002466 012737 000006 000004      3$:  MOV      #ERRVEC+2, @#ERRVEC ; RESTORE LOC 4
1165
1166      ;*****
1167      ;*TEST 2      TEST THAT THE WORD COUNT REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
1168      ;*****
1169      002474 000004      †ST2:  SCOPE
1170      002476 012737 002522 001110      MOV      #1$, $LPERR      ;; SET UP SCOPE LOOP ADRS
1171      002504 013737 001520 001122      MOV      DRVWCR, $BDADR      ;; SET UP WC REG ADRS
1172      002512 005000      CLR      RO      ;; RO SAYS SHIFT PTRN WHEN 0
1173      002514 012737 177776 001124      MOV      #-2, $GDDAT      ;; FLOAT 0 RIGHT TO LEFT
1174      002522 013777 001124 176770      1$:  MOV      $GDDAT, @DRVWCR      ;; LD WC
1175      002530 017737 176764 001126      MOV      @DRVWCR, $BDDAT      ;; READ IT BACK
1176      002536 023737 001124 001126      CMP      $GDDAT, $BDDAT      ;; CORRECT?
1177      002544 001401      BEQ      2$      ;; BR IF SO
1178      002546 104002      ERROR   2      ;; WORD COUNT WRITE/READ FAILURE
1179      002550 005137 001124      2$:  COM      $GDDAT      ;; COMPLEMENT ZERO
1180      002554 005100      COM      RO      ;; RO SAYS SHIFT LEFT WHEN = 0
1181      002556 001361      BNE      1$      ;; TRY THE COMPLEMENT IF RO NOT 0
1182      002560 006337 001124      ASL      $GDDAT      ;; COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
1183      002564 005237 001124      INC      $GDDAT      ;; KEEP LSB SET
1184      002570 103754      BCS      1$      ;; AGAIN TILL ALL PATRNS DONE
1185
1186      ;*****
1187      ;*TEST 3      TEST THAT THE BUFFER ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
1188      ;*****
1189      002572 000004      †ST3:  SCOPE
1190      002574 012737 002620 001110      MOV      #1$, $LPERR      ;; SET UP SCOPE LOOP ADRS
1191      002602 013737 001522 001122      MOV      DRVBAR, $BDADR      ;; SET UP BA REG ADRS
1192      002610 005000      CLR      RO      ;; RO SAYS SHIFT PTRN WHEN 0
1193      002612 012737 177774 001124      MOV      #-4, $GDDAT      ;; FLOAT 0 RIGHT TO LEFT
1194      002620 013777 001124 176674      1$:  MOV      $GDDAT, @DRVBAR      ;; LD BA
1195      002626 017737 176670 001126      MOV      @DRVBAR, $BDDAT      ;; READ IT BACK
1196      002634 042737 000001 001126      BIC      #BIT00, $BDDAT      ;; DON'T WANT BIT00

```

N02

```

1197 002642 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
1198 002650 001401                      BEQ      2$                ;BR IF SO
1199 002652 104002                      ERROR   2                  ;BUS ADRS WRITE/READ FAILURE
1200 002654 005137 001124                2$:   COM      $GDDAT      ;COMPLEMENT ZERO
1201 002660 042737 000001 001124      BIC     #BIT00,$GDDAT     ;BIT 00 NOT INVOLVED
1202 002666 005100                      COM     R0                ;R0 SAYS SHIFT LEFT WHEN = 0
1203 002670 001353                      BNE     1$                ;TRY THE COMPLEMENT IF R0 NOT 0
1204 002672 006337 001124                ASL     $GDDAT            ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
1205 002676 103004                      BCC     TST4              ;NEXT TEST IF BIT 15 DONE
1206 002700 062737 000002 001124      ADD     #2,$GDDAT        ;KEEP ADDR LSB SET
1207 002706 000744                      BR      1$                ;AGAIN TILL ALL PATTERNS DONE
1208
1209                                     ;:*****
1210                                     ;:TEST 4      TEST THAT THE DATA BUFFER REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
1211                                     ;:*****
1212 002710 000004      TST4:  SCOPE
1213 002712 012737 002736 001110      MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
1214 002720 013737 001526 001122      MOV     DRVDDBR,$BDBR   ;SET UP DB REG ADRS
1215 002726 005000                      CLR     R0                ;R0 SAYS SHIFT PTRN WHEN 0
1216 002730 012737 177776 001124      MOV     #-2,$GDDAT      ;FLOAT 0 RIGHT TO LEFT
1217 002736 013777 001124 176562      1$:   MOV     $GDDAT,$DRVDDBR ;LD DB
1218 002744 017737 176556 001126      MOV     $DRVDDBR,$BDDAT ;READ IT BACK
1219 002752 023737 001124 001126      CMP     $GDDAT,$BDDAT   ;CORRECT?
1220 002760 001401                      BEQ     2$                ;BR IF SO
1221 002762 104002                      ERROR   2                  ;DATA BUFFER WRITE/READ FAILURE (LOOP BACK)
1222 002764 005137 001124                2$:   COM     $GDDAT      ;COMPLEMENT ZERO
1223 002770 005100                      COM     R0                ;R0 SAYS SHIFT LEFT WHEN = 0
1224 002772 001361                      BNE     1$                ;TRY THE COMPLEMENT IF R0 NOT 0
1225 002774 006337 001124                ASL     $GDDAT            ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
1226 003000 005237 001124                INC     $GDDAT            ;KEEP LSB SET
1227 003004 103754                      BCS     1$                ;AGAIN TILL ALL PATRNS DONE
1228
1229                                     ;:*****
1230                                     ;:TEST 5      TEST THAT THE DATA BUFFER REG IS BYTE ADDRESSABLE
1231                                     ;:*****
1232 003006 000004      TST5:  SCOPE
1233 003010 013700 001526                MOV     DRVDDBR,R0      ;GET DB REG ADRS
1234 003014 010037 001122                MOV     R0,$BDBR        ;SET UP DB REG ADRS
1235 003020 005010                      CLR     (R0)             ;ZERO DATA BUFFER REG
1236 003022 012737 177177 001124      MOV     #177177,$GDDAT  ;LD EXPECTED
1237 003030 012737 077776 001126      MOV     #77776,$BDDAT   ;SEND DATA FROM "BDDAT"
1238 003036 153760 001126 000001      BISB   $BDDAT,1(R0)     ;LOAD HI BYTE DB
1239 003044 153710 001127                BISB   $BDDAT+1,(R0)    ;LOAD LO BYTE DB
1240 003050 011037 001126                MOV     (R0),$BDDAT     ;READ IT BACK
1241 003054 023737 001124 001126      CMP     $GDDAT,$BDDAT   ;CORRECT?
1242 003062 001401                      BEQ     TST6              ;NEXT TEST IF SO
1243 003064 104002                      ERROR   2                  ;DATA ERROR ON BYTE ADDRESSING THE DATA BUFFER REG
1244
1245                                     ;:*****
1246                                     ;:TEST 6      TEST THAT RESET CLEARS WORD COUNT, BUS ADDRESS & DATA REGS
1247                                     ;:*****
1248 003066 000004      TST6:  SCOPE
1249 003070 012737 000010 001160      MOV     #10,$TIMES      ;DO 10 ITERATIONS
1250 003076 005037 001124                CLR     $GDDAT           ;LD EXPECTED

```

```

1251 003102 012777 177777 176410      MOV      #-1, @DRVWCR      ;SET ALL BITS - WC REG
1252 003110 012777 177776 176404      MOV      #-2, @DRVBAR    ;SET ALL BITS - BUS ADRS REG
1253 003116 012777 177777 176402      MOV      #-1, @DRVDBR    ;SET ALL BITS - DB OUT REG
1254 003124 000005                RESET                ;DO A BUS RESET
1255 003126 017737 176366 001126      MOV      @DRVWCR, $BDDAT ;READ WC REG
1256 003134 001404                BEQ      1$           ;BR IF CLRED
1257 003136 013737 001520 001122      MOV      DRVWCR, $BDAOR  ;SET UP WC REG ADRS
1258 003144 104003                ERROR      3           ;RESET FAILED TO CLR WC REG
1259 003146 017737 176350 001126 13:  MOV      @DRVBAR, $BDDAT ;READ BUS ADRS REG
1260 003154 042737 000001 001126      BIC      #BIT00, $BDDAT  ;DON'T WANT BIT00
1261 003162 001404                BEQ      2$           ;BR IF CLRED
1262 003164 013737 001522 001122      MOV      DRVBAR, $BDAOR  ;SET UP BA REG ADRS
1263 003172 104003                ERROR      3           ;RESET FAILED TO CLR BUS ADRS REG
1264 003174 017737 176326 001126 25:  MOV      @DRVDBR, $BDDAT ;READ DATA BUFFER REG
1265 003202 001404                BEQ      TST7         ;NEXT TEST IF CLRED
1266 003204 013737 001526 001122      MOV      DRVDBR, $BDAOR  ;SET UP DB REG ADRS
1267 003212 104003                ERROR      3           ;RESET FAILED TO CLR DATA BUFFER OUT REG

```

```

1268
1269
1270      ;*****
1271      ;*TEST 7      TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT PTRN)
1272      ;*****

```

```

1272 003214 000004      †TST7:  SCOPE
1273 003216 012737 000010 001160      MOV      #10, $TIMES    ;DO 10 ITERATIONS
1274 003224 106427 000200                MTPS,      200          ;DON'T WANT ANY INTRs
1275 003230 004537 007602                JSR      RS, SETVEC     ;SET UP INTR RETURN ADRS IN CASE
1276 003234 003332                3$           ;RETURN TO 3$ ON ILLEGAL INTR
1277 003236 013737 003256 001110      MOV      1$, $LPERR     ;SET UP SCOPE LOOP ADRS
1278 003244 013737 001524 001122      MOV      DRVCSR, $BDAOR  ;SET UP CSR ADRS
1279 003252 012700 160000                MOV      #160000, R0     ;START AT 0 - HI BITS FOR NOISE
1280 003256 010037 001124 15:  MOV      R0, $GDDAT     ;LD EXPECTED
1281 003262 042737 167201 001124      BIC      #167201, $GDDAT ;MASK TO WRITEABLE BITS
1282 003270 010077 176230                MOV      R0, @DRVCSR     ;LD CSR
1283 003274 017737 176224 001126      MOV      @DRVCSR, $BDDAT ;READ IT BACK
1284 003302 042737 007200 001126      BIC      #7200, $BDDAT  ;DON'T LOOK AT STAT & RDY BITS
1285 003310 023737 001 24 001126      CMP      $GDDAT, $BDDAT ;CORRECT?
1286 003316 001401                BEQ      2$           ;BR IF SO
1287 003320 104002                ERROR      2           ;CONTROL/STATUS REG WRITE/READ FAILURE
1288 003322 062700 000002 25:  ADD      #2, R0         ;ADVANCE COUNT PATTERN
1289 003326 001353                BNE      1$           ;WRITE NEXT PATTERN IF NOT ALL TESTED
1290 003330 000413                BR      4$           ;GO RESTORE VECTOR
1291 003332 022626 35:  CMP      (SP)+, (SP)+   ;FIX STACK - SHOULD NOT HAVE INTR'ED
1292 003334 052737 000200 001124      BIS      #200, $GDDAT   ;CORRECT EXPECTED
1293 003342 017737 176156 001126      MOV      @DRVCSR, $BDDAT ;READ CSR
1294 003350 042737 007000 001126      BIC      #7000, $BDDAT  ;DON'T WANT 'STAT' BITS
1295 003356 104005                ERROR      5           ;CPU FAILED TO LOCK OUT DRV118 INTR REQ
1296 003360 004737 007622 45:  JSR      PC, RSTVEC     ;GO RESTORE VECTOR

```

```

1297
1298
1299      ;*****
1300      ;*TEST 10     TEST THAT RESET CLEARS ALL WRITEABLE BITS & SET READY IN CSR
1301      ;*****

```

```

1301 003364 000004      †TST10: SCOPE
1302 003366 012737 000010 001160      MOV      #10, $TIMES    ;DO 10 ITERATIONS
1303 003374 013737 001524 001122      MOV      DRVCSR, $BDAOR  ;SET UP CSR ADRS
1304 003402 012737 000200 001124      MOV      #200, $GDDAT   ;LD EXPECTED

```



```

1359 003666 012777 003732 175634      MOV      #25,DRVCTO      ;SET UP EXPECTED INTR RETURN ADPS
1360 003674 052777 000100 175622      BIS      #BIT6,DRVCSR   ;ENABLE THE EXPECTED INTERRUPT
1361 003702 021616                CMP      (SP),(SP)      ;STALL
1362 003704 017737 175614 001126      MOV      DRVCSR,$BDDAT  ;GET THE CSR
1363 003712 104005                ERROR   5              ;READY FAILED TO CAUSE AN INTERRUPT
1364 003714 000417                BR       3$            ;GO RESTORE VECTOR
1365 003716 022626                CMP      (SP)+(SP)+    ;SHOULD NEVER GET HERE - IE NOT WORKING
1366 003720 017737 175600 001126      MOV      DRVCSR,$BDDAT  ;GET THE CSR
1367 003726 104005                ERROR   5              ;READY INTERRUPTED WITHOUT THE IE BIT
1368 003730 000411                BR       3$            ;GO RESTORE VECTOR
1369 003732 022626                CMP      (SP)+(SP)+    ;FIX STACK SINCE NO RETURN
1370 003734 017737 175564 001126      MOV      DRVCSR,$BDDAT  ;READ STATUS
1371 003742 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;CORRECT?
1372 003750 001401                BEQ     3$            ;BR IF SO
1373 003752 104005                ERROR   5              ;INCORRECT STATUS ON READY INTR
1374 003754 004737 007622 3$:      JSR     PC,RSTVEC      ;GO RESTORE VECTOR

```

```

*****
;TEST 14      TEST THAT GO CLRS READY & FNCT 2 WILL SET IT
*****

```

```

1375
1376
1377
1378
1379 003760 000004      TST14:  SCOPE
1380 003762 106427 000200      MTPS,   200           ;DONT WANT ANY INTRs
1381 003766 013737 001524 001122      MOV     DRVCSR,$BDAOR  ;SET UP CSR ADPS
1382 003774 005037 001124                CLR     $GDDAT         ;EXPECT 0
1383 004000 012777 000001 175516      MOV     #1,DRVCSR     ;SET GO WHICH SHOULD CLR READY
1384 004006 017737 175512 001126      MOV     DRVCSR,$BDDAT ;READ THE CSR
1385 004014 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
1386 004022 001401                BEQ     1$            ;BR IF SO
1387 004024 104006                ERROR   6              ;THE GO BIT FAILED TO CLR READY
1388 004026 052777 000004 175470 1$:      BIS     #4,DRVCSR     ;FNCT 2 SHOULD SET READY
1389 004034 012737 002204 001124      MOV     #2204,$GDDAT  ;LD EXPECTED
1390 004042 017737 175456 001126      MOV     DRVCSR,$BDDAT ;GET CSR
1391 004050 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
1392 004056 001401                BEQ     TST15         ;NEXT TEST IF SET
1393 004060 104006                ERROR   6              ;FNCT 2 (VIA ATTN) FAILED TO SET READY
1394
1395
1396
1397

```

```

*****
;TEST 15      TEST THAT READY CONTROLS 'BAR' BIT00
*****

```

```

1398 004062 000004      TST15:  SCOPE
1399 004064 012777 000004 175432      MOV     #4,DRVCSR     ;SET READY
1400 004072 013737 001522 001122      MOV     DRVBAR,$BDAOR ;SET UP BAR ADPS
1401 004100 012737 000001 001124      MOV     #1,$GDDAT     ;EXPECT LSB OF BAR
1402 004106 005077 175410                CLR     DRVBAR        ;CLR BAR
1403 004112 017737 175404 001126      MOV     DRVBAR,$BDDAT ;READ BAR
1404 004120 023737 001124 001126      CMP     $GDDAT,$BDDAT ;CORRECT?
1405 004126 001401                BEQ     1$            ;BR IF SO
1406 004130 104002                ERROR   2              ;A00 FAILED TO READ A ONE(SB TIED TO RDY)
1407 004132 012777 000001 175364 1$:      MOV     #1,DRVCSR     ;SET GO(CLRs BAR BIT00)
1408 004140 017737 175356 001126      MOV     DRVBAR,$BDDAT ;READ BAR
1409 004146 001403                BEQ     2$            ;BR IF ZERO
1410 004150 005037 001124                CLR     $GDDAT        ;EXPECTED ZERO
1411 004154 104002                ERROR   2              ;WHEN RDY CLRD-A00 FAILED TO READ A ZERO
1412 004156 012777 000004 175340 2$:      MOV     #4,DRVCSR     ;INSURE RDY SET BEFORE ADVANCING

```

E03

```

1413
1414
1415
1416
1417 004164 000004
1418 004166 013737 001526 001122
1419 004174 012737 125252 001124
1420 004202 013777 001124 175316
1421 004210 012777 000400 175306
1422 004216 012777 052525 175302
1423 004224 017737 175276 001126
1424 004232 023737 001124 001126
1425 004240 001401
1426 004242 104017
1427 004244 042777 000400 175252 1$:
1428 004252 012737 052525 001124
1429 004260 017737 175242 001126
1430 004266 023737 001124 001126
1431 004274 001401
1432 004276 104002
1433
1434
1435
1436
1437 004300 000004
1438 004302 012737 004324 001110
1439 004310 004537 007602
1440 004314 004420
1441 004316 012700 177776
1442 004322 005001
1443 004324 012777 177777 175166 1$:
1444 004332 012777 014476 175162
1445 004340 010037 014476
1446 004344 012777 000101 175152
1447 004352 052777 000400 175144
1448 004360 106427 000000
1449 004364 013737 001524 001122
1450 004372 012737 000700 001124
1451 004400 017737 175120 001126
1452 004406 042777 000100 175110
1453 004414 104005
1454 004416 000442
1455 004420 022626 2$:
1456 004422 004537 007646
1457 004426 000700
1458 004430 000001
1459 004432 104007
1460 004434 000433
1461 004436 104010
1462 004440 000431
1463 004442 104011
1464 004444 000427
1465 004446 012737 014476 001120
1466 004454 010037 001124

```

```

*****
*TEST 16 TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)
*****
↑ST16: SCOPE
MOV DRVDBR,$BDDADR ;SET UP DBR ADRS
MOV #125252,$GDDAT ;LD EXPECTED
MOV $GDDAT,DRVDBR ;LD DBR WITH #125252
MOV #400,DRVCSR ;SET CYCLE - SHOULD CLK DBR (IN)
MOV #52525,DRVDBR ;CHANGE DBR (OUT) DATA - SHOULD NOT AFFECT (IN)
MOV DRVDBR,$BDDAT ;READ DBR
CMP $GDDAT,$BDDAT ;CORRECT?
BEQ 15 ;BR IF 50
ERROR 17 ;CYCLE DID NOT LATCH DBR (IN) DATA
1$: BIC #400,DRVCSR ;REMOVE CYCLE
MOV #52525,$GDDAT ;NOW EXPECT #52525
MOV DRVDBR,$BDDAT ;READ DBR
CMP $GDDAT,$BDDAT ;CORRECT?
BEQ TST17 ;NEXT TEST IF 50
ERROR 2 ;DBR FAILED TO READ WHEN CYCLE CLEARED (NORMAL)
*****
*TEST 17 TEST SINGLE "DATA" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
*****
↑ST17: SCOPE
MOV #15,$LPERR ;SET UP SCOPE LOOP ADRS
JSR R5,SETVEC ;GO SET UP INTERRUPT RETURN
2$: ;RETURN TO 2$ ON INTR
MOV #-2,R0 ;FLOAT ZERO RIGHT TO LEFT
CLR R1 ;R1 CONTROLS DATA SHIFTING
1$: MOV #-1,DRVWCR ;DO ONE XFER
MOV #DBUF,DRVBAR ;GET DATA WORD FROM "DBUF"
MOV R0,DBUF ;SET UP MEM DATA
MOV #101,DRVCSR ;SET IE & GO
BIS #400,DRVCSR ;SET CYCLE
MTPS,0 ;ENABLE THE INTR
MOV DRVCSR,$BDDADR ;SET UP CSR ADRS
MOV #700,$GDDAT ;LD EXPECTED
MOV DRVCSR,$BDDAT ;READ THE CSR
BIC #100,DRVCSR ;CLR IE
ERROR 5 ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
BR 4$ ;GO RESTORE VECTOR
2$: CMP (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
JSR R5,CKSTAT ;GO CHECK STATUS
700 ;CSR STATUS EXPECTED
1 ;# OF XFERS
ERROR 7 ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
BR 4$ ;GO RESTORE VECTOR
ERROR 10 ;RETURN HERE IF WC ER - EXPECTED 0
BR 4$ ;GO RESTORE VECTOR
ERROR 11 ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
BR 4$ ;GO RESTORE VECTOR
MOV #DBUF,$GDADR ;RETURN HERE IF OK - SET UP XFER ADRS
MOV R0,$GDDAT ;LD EXPECTED

```

# F03

```

1457 004460 017737 175042 001126      MOV      2DRVDBR,$BDDAT ;READ DATA XFERED
1458 004466 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
1459 004474 001405                      BEQ      3$ ;BR IF 50
1470 004476 013737 001526 001122      MOV      DRVDBR,$BDADR ;SET UP DBR ADRS
1471 004504 104012                      ERROR    12 ;DATA ER - DBR CONTAINS WRONG DATA
1472 004506 000406                      BR       4$ ;GO RESTORE VECTOR
1473 004510 005100                      3$:     COM      R0 ;RETURN HERE ON GOOD DATA - NOW COM PATRN
1474 004512 005101                      COM      R1 ;KEEP TRACK OF COMPLEMENT
1475 004514 001303                      BNE      1$ ;DO COMPLEMENT OF THIS FLOATING ZERC IF C
1476 004516 006300                      ASL      R0 ;WAS DONE - NOW SHIFT ZERO LEFT
1477 004520 005200                      INC      R0 ;KEEP LSB SET
1478 004522 103700                      BCS      1$ ;AGAIN TILL ZERO BIT IN CARRY
1479 004524 004737 007622                      4$:     JSR      PC,RSTVEC ;GO RESTORE VECTOR
1480
1481 ;*****
1482 ;*TEST 20 TEST SINGLE "DATO" NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
1483 ;*****
1484 004530 000004      †ST20: SCOPE
1485 004532 012737 004554 001110      MOV      #15,$LPERR ;SET UP SCOPE LOOP ADRS
1486 004540 004537 007602                      JSR      R5,SETVEC ;GO SET UP INTERRUPT RETURN
1487 004544 004642                      2$      ;RETURN TO 2$ ON INTR
1488 004546 012700 177776                      MOV      #-2,R0 ;FLOAT ZERO RIGHT TO LEFT
1489 004552 005001                      CLR      R1 ;R1 CONTROLS DATA SHIFTING
1490 004554 012777 177777 174736 1$:     MOV      #-1,2DRVWCR ;DO ONE XFER
1491 004562 012777 014476 174732      MOV      #DBUF,2DRVBAR ;WRITE DATA WORD TO "DBUF"
1492 004570 010077 174732                      MOV      R0,2DRVDBR ;SET UP DATA IN DBR
1493 004574 012777 000103 174722      MOV      #103,2DRVCSR ;SET IE, GO & FNCT1 (C1 CONTROL)
1494 004602 052777 000400 174714      BIS      #400,2DRVCSR ;SET CYCLE
1495 004610 106427 000000                      MTPS, 0 ;ENABLE THE INTR
1496 004614 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRS
1497 004622 012737 001702 001124      MOV      #1702,$GDDAT ;LD EXPECTED
1498 004630 017737 174670 001126      MOV      2DRVCSR,$BDDAT ;READ THE CSR
1499 004636 104005                      ERROR    5 ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
1500 004640 000442                      BR       4$ ;GO RESTORE VECTOR
1501 004642 022626                      2$:     CMP      (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
1502 004644 004537 007646                      JSR      P5,CKSTAT ;GO CHECK STATUS
1503 004650 001702                      1702 ;CSR STATUS EXPECTED
1504 004652 000001                      ;# OF XFERS
1505 004654 104007                      ERROR    7 ;RETURN HERE IF STATUS ER - EXPECTED STAT C.
1506 ;CYCLE, READY, IE & FNCT 1
1507 004656 000433                      BR       4$ ;GO RESTORE VECTOR
1508 004660 104010                      ERROR    10 ;RETURN HERE IF WC ER - EXPECTED 0
1509 004662 000431                      BR       4$ ;GO RESTORE VECTOR
1510 004664 104011                      ERROR    11 ;RETURN HERE IF BAR ER - SHOULD = DBUF+2
1511 004666 000427                      BR       4$ ;GO RESTORE VECTOR
1512 004670 012737 014476 001120      MOV      #DBUF,$GDADR ;RETURN HERE IF OK - SET JP XFER ADRS
1513 004676 010037 001124                      MOV      R0,$GDDAT ;LD EXPECTED
1514 004702 013737 014476 001126      MOV      DBUF,$BDDAT ;GET DATA XFERED
1515 004710 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
1516 004716 001405                      BEQ      3$ ;BR IF 50
1517 004720 013737 001526 001122      MOV      DRVDBR,$BDADR ;SET UP DBR ADRS
1518 004726 104013                      ERROR    13 ;DATA ER - MEM CONTAINS WRONG DATA
1519 004730 000406                      BR       4$ ;GO RESTORE VECTOR
1520 004732 005100                      3$:     COM      R0 ;RETURN HERE ON GOOD DATA - NOW COM PATRN

```

G03

```

1521 004734 005101          CUM      R1          ;KEEP TRACK OF COMPLEMENT
1522 004736 001306          BNE     1$          ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
1523 004740 006300          ASL     R0          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
1524 004742 005200          INC     R0          ;KEEP LSB SET
1525 004744 103703          BCS     1$          ;AGAIN TILL ZERO BIT IN CARRY
1526 004746 004737 007622 4$:      JSR     PC,RSTVEC  ;GO RESTORE VECTOR
1527
1528
1529
1530
1531 004752 000004          ;*****
1532 004754 012737 000010 001160 ;*TEST 21      TEST 200 "DATI" NPR TRANSFERS (BURST MODE)
1533 004762 004537 007602          ;*****
1534 004766 005064          †ST21:  SCOPE
1535 004770 004737 010026          MOV     #10,$TIMES ;DO 10 ITERATIONS
1536 004774 012777 177470 174516 JSR     R5,$ETVEC  ;GO SET UP INTERRUPT RETURN
1537 005002 012777 014476 174512 1$      JSR     1$          ;RETURN TO 1$ ON INTR
1538 005010 106427 000000          IS?    PC,LDBUF    ;GO LOAD BUFFER WITH COMPLEMENTING PATRN
1539 005014 012777 000101 174502 MOV     #-200,$DRVWCR ;LOAD WC REG - WILL DO 200 XFERS
1540 005022 052777 000400 174474 MOV     #DBUF,$DRVBAR ;SET UP CURRENT ADRS
1541 005030 013737 001524 001122 MTPS,  0          ;ENABLE THE INTR
1542 005036 012737 000700 001124 MOV     #101,$DRVCSR ;SET IE & GO
1543 005044 017737 174454 001126 BIS     #400,$DRVCSR ;SET CYCLE
1544 005052 042777 000100 174444 MOV     DRVCSR,$BDDADR ;SET UP CSR ADRS
1545 005060 104005          MOV     #700,$GDDAT ;LD EXPECTED
1546 005062 000434          BIC     #100,$DRVCSR ;READ THE CSR
1547 005064 022626          ERPOR  5          ;CLR INTR ENABLE
1548 005066 004537 007646 1$:      BR     2$          ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
1549 005072 000700          CMP     (SP)+,(SP)+ ;GO RESTORE VECTOR
1550 005074 000310          JSR     R5,CKSTAT ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
1551 005076 104007          700      ;GO CHECK STATUS
1552 005100 000425          200      ;CSR STATUS EXPECTED
1553 005102 104010          ERROR   7          ;# OF XFERS
1554 005104 000423          BR     2$          ;RETURN HERE IF STATUS ER - EXPECTED CYCLE. READY & IE
1555 005106 104011          ERROR   10         ;GO RESTORE VECTOR
1556 005110 000421          BR     2$          ;RETURN HERE IF WC ER - EXPECTED 0
1557 005112 012737 015314 001120 MOV     #DBUF+616,$GDADR ;GO RESTORE VECTOR
1558 005120 012737 070707 001124 MOV     #70707,$GDDAT ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
1559 005126 017737 174374 001126 MOV     $DRVDBR,$BDDADR ;OK - SET UP LAST XFER ADRS WHERE #70707 SHOULD BE
1560 005134 023737 001124 001126 CMP     $GDDAT,$BDDAT ;LD EXPECTED
1561 005142 001404          BEQ     2$          ;DBR SHOULD HAVE LAST DATUM
1562 005144 013737 001526 001122 MOV     DRVDBR,$BDDADR ;CORRECT?
1563 005152 104012          ERROR   12         ;BR IF 50
1564 005154 004737 007622 2$:      JSR     PC,RSTVEC  ;SET UP DBR ADRS
1565
1566
1567
1568
1569 005160 000004          ;*****
1570 005162 012737 000010 001160 ;*TEST 22      TEST 200 "DATO" NPR TRANSFERS (BURST MODE)
1571 005170 004537 007602          ;*****
1572 005174 005274          †ST22:  SCOPE
1573 005176 012777 177470 174314 MOV     #10,$TIMES ;DO 10 ITERATIONS
1574 005204 012777 014476 174310 JSR     R5,$ETVEC  ;GO SET UP INTERRUPT RETURN
1$      JSR     1$          ;RETURN TO 1$ ON INTR
MOV     #-200,$DRVWCR ;WORD WC REG - WILL DO 200 XFER'S
MOV     #DBUF,$DRVBAR ;SET UP CURRENT ADRS

```

```

1575 005212 012777 177377 174306      MOV      #177377, @DRVDBR      ; THIS WILL BE WRITTEN TO MEM
1576 005220 106427 000000                MTPS,      0                ; ENABLE THE INTR
1577 005224 012777 000103 174272      MOV      #103, @DRVCSR      ; SET IE, FNCT 1 & GO
1578 005232 052777 000400 174264      BIS      #400, @DRVCSR      ; SET CYCLE
1579 005240 013737 001524 001122      MOV      DRVCSR, $BDAOR      ; SET UP CSR ADRS
1580 005246 012737 001702 001124      MOV      #1702, $GDDAT      ; LD EXPECTED
1581 005254 017737 174244 001126      MOV      @DRVCSR, $BDDAT     ; READ THE CSR
1582 005262 042777 000100 174234      BIC      #100, @DRVCSR      ; CLR INTR ENABLE
1583 005270 104005                ERROR      5                ; WCO FAILED TO INTERRUPT (SNGL CYCL' ON' COULD CAUSE THIS)
1584 005272 000416                BR         2$                ; GO RESTORE VECTOR
1585 005274 022626                :S:    CMP      (SP)+, (SP)+    ; INTR RETURNS HERE - FIX STACK SINCE NO RTI
1586 005276 004537 007646                JSR      R5, CKSTAT         ; GO CHECK STATUS
1587 005302 001702                1702                ; CSR STATUS EXPECTED
1588 005304 000310                200                ; # OF XFRS
1589 005306 104007                ERROR      7                ; RETURN HERE IF STATUS ER - EXPECTED STAT C.
1590                                ; CYCLE, READY, IE & FNCT 1
1591 005310 000407                BR         2$                ; GO RESTORE VECTOR
1592 005312 104010                ERROR      10               ; RETURN HERE IF WC ER - EXPECTED 0
1593 005314 000405                BR         2$                ; GO RESTORE VECTOR
1594 005316 104011                ERROR      11               ; RETURN HERE IF BAR ER - SHOULD = DBUF+620
1595 005320 000403                BR         2$                ; GO RESTORE VECTOR
1596 005322 004737 010150                JSR      PC, CKDAT          ; RETURN HERE IF OK - NOW GO CHECK DATA
1597 005326 104013                ERROR      13               ; RETURN HERE IF DATA ER - DBR CONTAINS WRONG DATA
1598 005330 004737 007622                :S:    JSR      PC, RSTVEC     ; RETURN HERE IF DATA CHECK OK - GO RESTORE VECTOR
1599
1600
1601                                ; *****
1602                                ; *TEST 23      TEST THAT THE CPU IS LOCKED OUT WITH SINGLE CYCLE OFF
1603                                ; *****
1604                                ; *ST23:  SCOPE
1605                                ;
1606                                ; DO 10 ITERATIONS
1607                                ; SET UP CSR ADRS
1608                                ; GO SET UP INTR RETURN
1609                                ; RETURN TO 3$ ON INTR
1610                                ; DO EIGHT 200 WORD XFER'S
1611                                ; USR $BDDAT AS A COUNTER
1612                                ; DO 200 XFRS (DATI'S)
1613                                ; FROM DBUF
1614                                ; ALLOW AN INTR
1615                                ; SET IE & GO
1616                                ; SET CYCLE
1617                                ; FREEBEE
1618                                ;
1619                                ; START COUNTING - SHOULD NEVER GET HERE
1620                                ; UNTIL 64K
1621                                ; LD EXPECTED
1622                                ; READ STATUS
1623                                ; CLR IE
1624                                ; NO INTERRUPT ON 200 DATI'S
1625                                ; GO RESTORE VECTOR
1626                                ; FIX STACK SINCE NO RTI
1627                                ; DONE 8 TIMES?
1628                                ; BR IF NOT
1629                                ; SHOULD STILL BE ZERO

```

```

1603 005334 000004                :S:    MOV      #10, $TIMES    ; DO 10 ITERATIONS
1604 005336 012737 000010 001160      MOV      DRVCSR, $BDAOR      ; SET UP CSR ADRS
1605 005344 013737 001524 001122      JSR      R5, SETVEC         ; GO SET UP INTR RETURN
1606 005352 004537 007602                3$                ; RETURN TO 3$ ON INTR
1607 005356 005466                MOV      #10, R0            ; DO EIGHT 200 WORD XFER'S
1608 005360 012700 000010                CLR      $BDDAT            ; USR $BDDAT AS A COUNTER
1609 005364 005037 001126                :S:    MOV      #-200, @DRVWCR ; DO 200 XFRS (DATI'S)
1610 005370 012777 177470 174122      MOV      #DBUF, @DRVBAR     ; FROM DBUF
1611 005376 012777 014476 174116      MTPS,      0                ; ALLOW AN INTR
1612 005404 106427 000000                MOV      #101, @DRVCSR      ; SET IE & GO
1613 005410 012777 000101 174106      BIS      #400, @DRVCSR      ; SET CYCLE
1614 005416 052777 000400 174100      NOP                ; FREEBEE
1615 005424 000240                NOP
1616 005426 000240                NOP
1617 005430 000240                NOP
1618 005432 005237 001126                :S:    INC      $BDDAT         ; START COUNTING - SHOULD NEVER GET HERE
1619 005436 001375                BNE      2$                ; UNTIL 64K
1620 005440 012737 000700 001124      MOV      #700, $GDDAT       ; LD EXPECTED
1621 005446 017737 174052 001126      MOV      @DRVCSR, $BDDAT     ; READ STATUS
1622 005454 042777 000100 174042      BIC      #100, @DRVCSR      ; CLR IE
1623 005462 104005                ERROR      5                ; NO INTERRUPT ON 200 DATI'S
1624 005464 000407                BR         4$                ; GO RESTORE VECTOR
1625 005466 022626                :S:    CMP      (SP)+, (SP)+    ; FIX STACK SINCE NO RTI
1626 005470 005300                DEC      R0                ; DONE 8 TIMES?
1627 005472 001336                BNE      1$                ; BR IF NOT
1628 005474 005737 001126                TST      $BDDAT            ; SHOULD STILL BE ZERO

```

```

1629 005500 001401 BEQ 4$ ;BR IF 50
1630 005502 104014 ERROR 14 ;BURST MD (SINGLE CYCLE=0) FAILS TO LOCK OUT CPU
1631 005504 004737 007622 4$: JSR PC,RSTVEC ;GO RESTORE VECTOR
1632
1633 ;*****
1634 ;*TEST 24 TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON
1635 ;*****
1636 005510 000004 †S-24: SCOPE
1637 005512 012737 000010 001160 MOV #10,$TIMES ;DO 10 ITERATIONS
1638 005520 013737 001524 00112E MOV DRVCSR,$BDADR ;SET UP CSR ADRS
1639 005526 004537 007602 JSR R5,SETVEC ;GO SET UP INTR RETURN
1640 005532 005640 3$ ;RETURN TO 3$ ON INTR
1641 005534 012700 000010 MOV #10,R0 ;DO EIGHT 200 WORD XFER'S
1642 005540 012737 000000 001126 MOV #0,$BDDAT ;USE $BDDAT AS A COUNTER
1643 005546 012777 177470 173744 1$: MOV #-200,$DRVWCR ;DO 200 XFERS (DATI'S)
1644 005554 012777 014476 173740 MOV $DBUF,$DRVBAR ;FROM DBUF
1645 005562 106427 000000 MTPS, 0 ;ALLOW AN INTR
1646 005566 012777 000111 173730 MOV #111,$DRVCSR ;SET IE, FNCT3 & GO
1647 005574 052777 000400 173722 BIS #400,$DRVCSR ;SET CYCLE
1648 005602 000240 NOP ;FREEBEE
1649 005604 005237 001126 2$: INC $BDDAT ;START COUNTING
1650 005610 001375 BNE 2$ ;UNTIL 64K - SHOULD INTR BEFORE OVERFLOW
1651 005612 012737 004710 001124 MOV #4710,$GDDAT ;LD EXPECTED
1652 005620 017737 173700 001126 MOV $DRVCSR,$BDDAT ;READ STATUS
1653 005626 042777 000100 173670 BIC #100,$DRVCSR ;CLR IE
1654 005634 104005 ERROR 5 ;NO INTERRUPT ON 200 DATI'S (WITH SINGLE CYCLE)
1655 005636 000423 BR 5$ ;GO RESTORE VECTOR
1656 005640 02262E 3$: CMP (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
1657 005642 005300 DEC R0 ;DONE 8 TIMES?
1658 005644 001340 BNE 1$ ;BR IF NOT
1659 005646 022737 000000 001126 CMP #0,$BDDAT ;$BDDAT SHOULD HAVE BEEN COUNTED
1660 005654 103401 BCS 4$ ;BR IF 50
1661 005656 104015 ERROR 15 ;CPU APPEARED LOCKED OUT WITH SINGLE CYCLE SET
1662 005660 017737 173640 001126 4$: MOV $DRVCSR,$BDDAT ;READ STATUS
1663 005666 012737 004710 001124 MOV #4710,$GDDAT ;LD EXPECTED
1664 005674 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1665 005702 001401 BEQ 5$ ;BR IF 50
1666 005704 104007 EPROR 7 ;STATUS INCORRECT ON XFER WITH SINGLE CYCLE SET
1667 005706 004737 007622 5$: JSR PC,RSTVEC ;GO RESTORE VECTOR
1668
1669 ;*****
1670 ;*TEST 25 TEST THAT MAINT MODE CONTROLS FNCT BITS, XFER DIR & SINGLE CYCLE
1671 ;*****
1672 005712 000004 †S25: SCOPE
1673 005714 012737 000200 001160 MOV #200,$TIMES ;DO 200 ITERATIONS
1674 005722 004537 007602 JSR R5,SETVEC ;GO SET UP INTR RETURN
1675 005726 006044 2$ ;RETURN TO 2$ ON INTR
1676 005730 004737 010100 JSR FC,LDBUF1 ;GO SET UP DBUF (SPECIAL COM PATTERN)
1677 005734 012737 011702 006052 MOV #11702,3$ ;3$ CONTAINS EXPECTED STATUS
1678 005742 012737 000001 006054 MOV #1,4$ ;4$ CONTAINS THE CURRENT XFER NO # (MAX 8)
1679 005750 106427 000000 1$: MTPS, 0 ;ALLOW INTR
1680 005754 012777 014476 173740 MOV $DBUF,$DRVBAR ;SET UP CURRENT ADRS
1681 005762 013777 006054 173530 MOV 4$,$DRVWCR ;GET XFER #
1682 005770 005477 173524 NEG $DRVWCR ;NEGATE FOR WC

```

```

1683 005774 012777 010101 173522      MOV      #10101,DRVCSR ;SET UP MAINT, IE & CO
1684 006002 052777 000400 173514      BIS      #400,DRVCSR ;SET CYCLE
1685 006010 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRS
1686 006016 013737 006052 001124      MOV      3,$GDDAT ;LD EXPECTED
1687 006024 017737 173474 001126      MOV      @DRVCSR,$BDDAT ;READ STATUS
1688 006032 042777 000100 173464      BIC      #100,DRVCSR ;DISABLE IE
1689 006040 104005      ERROR 5 ;NO INTR ON XFER (IN MAINT MD)
1690 006042 000440      BR      6$ ;GO RESTORE VECTOR
1691 006044 022626      2$: CMP      (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1692 006046 004537 007646      JSR      R5,CKSTAT ;GO CHECK STATUS
1693 006052 011702      3$:      11702 ;THIS LOCATION WILL CONTAIN EXPECTED STATUS
1694 006054 000001      4$:      ! ;THIS LOCATION WILL CONTAIN CURRENT XFER # (MAX 8)
1695 006056 104007      ERROR 7 ;RETURN HERE IF STATUS ER - WILL EXPECT
1696 ; ;MAINT, COUNT INCREASE OF FNCT & STAT BITS,
1697 ; ;CYCLE, READY & IE
1698 006060 000431      BR      6$ ;GO RESTORE VECTOR
1699 006062 104010      ERROR 10 ;RETURN HERE IF WC ER - SHOULD BE 0
1700 006064 000427      BR      6$ ;GO RESTORE VECTOR
1701 006066 104011      ERROR 11 ;RETURN HERE IF BAR ER-
1702 006070 000425      BR      6$ ;GO RESTORE VECTOR
1703 006072 062737 001002 006052      ADD      #1002,3$ ;RETURN HERE IF OK - ADVANCE EXPECTED STATUS LOC
1704 006100 032737 020000 006052      BIT      #BIT13,3$ ;LOOK FOR OVERFLOW
1705 006106 001403      BEQ     5$ ;BR IF NOT
1706 006110 012737 010700 006052      MOV      #10700,3$ ;FNCT & STAT BITS SHOULD BE ZERO THIS TIME
1707 006116 005237 006054      5$: INC      4$ ;ADVANCE CURRENT XFER #
1708 006122 022737 000011 006054      CMP      #11,4$ ;HAVE 10 XFERS BEEN DONE
1709 006130 001307      BNE     1$ ;BR IF NOT
1710 006132 004537 010216      JSR      R5,CKDAT1 ;NOW GO CHECK DATA
1711 006136 000010      10 ;# OF XFER'S TO CHECK
1712 006140 104013      ERROR 13 ;RETURN HERE IF DATA ER - (WITH MAINT SET)
1713 006142 000240      NOP ;RESTORE VECTOR NEXT
1714 006144 004737 007622      6$: JSR      PC,RSTVEC ;GO RESTORE VECTOR
1715
1716 ;*****
1717 ;*TEST 26 TEST THAT A DATA FROM A NON-EXISTANT BUS ADRS SETS 'NEX'
1718 ;*****
1719 006150 000004      1ST26: SCOPE
1720 006152 012737 000100 001160      MOV      #100,$TIMES ;DO 100 ITERATIONS
1721 006160 012700 000002      MOV      #2,R0 ;R0 WHEN ZERO SAYS CLR 'NEX' WITH RESET
1722 006164 004537 007602      1$: JSR      R5,SETVEC ;GO SET UP INTERRUPT RETURN
1723 006170 006262      2$ ;RETURN TO 2$ ON TIMEOUT INTR
1724 006172 012777 177777 173320      MOV      #-1,DRVWCR ;SET UP FOR ONE XFER'S
1725 006200 012777 160000 173314      MOV      #160000,DRVBAR ;SET UP CA TO 160000 (RESERVED)
1726 006206 106427 000000      MTPS, 0 ;ALLOW INTR
1727 006212 012777 000161 173304      MOV      #161,DRVCSR ;SET IE, XAD 17,16 & GO
1728 006220 052777 000400 173276      BIS      #400,DRVCSR ;SET CYCLE
1729 006226 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRS - SHOULD NEVER GET HERE
1730 006234 012737 140760 001124      MOV      #140760,$GDDAT ;LD EXPECTED
1731 006242 017737 173256 001126      MOV      @DRVCSR,$BDDAT ;GIVE THEM THE STATUS
1732 006250 042777 000100 173246      BIC      #100,DRVCSR ;CLR IE
1733 006256 104016      ERROR 16 ;NEX FAILED TO CAUSE AN INTERRUPT
1734 006260 000521      BR      7$ ;GO RESTORE VECTOR
1735 006262 022626      2$: CMP      (SP)+,(SP)+ ;SHOULD INTR RETURN HERE - FIX STACK
1736 006264 017737 173234 001126      MOV      @DRVCSR,$BDDAT ;READ THE CSR

```

# K03

1737	006272	012737	140760	001124		MOV	#140760,\$GDDAT	;LD EXPECTED
1738	006300	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;CORRECT?
1739	006306	001405				BEQ	3\$	;BR IF SO
1740	006310	013737	001524	001122		MOV	DRVCSR,\$BDADR	;SET UP CSR ADRES
1741	006316	104016				ERROR	16	;STATUS ER - EXPECTED
1742								;ER, NEX, CYCLE, READY, IE, XAD17 & XAD16
1743	006320	000501				BR	7\$	;GO RESTORE VECTOR
1744	006322	017737	173172	001126	3\$:	MOV	QDRVWCR,\$BDDAT	;READ WORD COUNT
1745	006330	012737	177777	001124		MOV	#-1,\$GDDAT	;SHOULD STILL HAVE -1
1746	006336	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;CORRECT?
1747	006344	001405				BEQ	4\$	;BR IF SO
1748	006346	013737	001520	001122		MOV	DRVWCR,\$BDADR	;SET UP WCR ADRES
1749	006354	104010				ERROR	10	;WC INCREMENTED ON A TIMEOUT ER
1750	006356	000462				BR	7\$	;GO RESTORE VECTOR
1751	006360	017737	173136	001126	4\$:	MOV	QDRVBAR,\$BDDAT	;READ BUFFER ADRES
1752	006366	012737	160000	001124		MOV	#160000,\$GDDAT	;SHOULD NOT HAVE INCREMENTED
1753	006374	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;CORRECT?
1754	006402	001405				BEQ	5\$	;BR IF SO
1755	006404	013737	001522	001122		MOV	DRVBAR,\$BDADR	;SET UP BAR ADRES
1756	006412	104011				ERROR	11	;BAR INCREMENTED ON A TIMEOUT ER
1757	006414	000443				BR	7\$	;GO RESTORE VECTOR
1758	006416	005300			5\$:	DEC	RD	;KEEP TRACK ON HOW TO CLR
1759	006420	001422				BEQ	6\$	;BR IF CLR BY RESET
1760	006422	042777	040000	173074		BIC	#40000,QDRVCSR	;WRITE NEX TO ZERO
1761	006430	017737	173070	001126		MOV	QDRVCSR,\$BDDAT	;READ CSR
1762	006436	012737	000760	001124		MOV	#760,\$GDDAT	;LD EXPECTED
1763	006444	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;CORRECT?
1764	006452	001644				BEQ	1\$	;BR IF SO + REPEAT TEST FOR RESET TEST
1765	006454	013737	001524	001122		MOV	DRVCSR,\$BDADR	;SET UP CSR ADRES
1766	006462	104016				ERROR	16	; 'NEX' FAILED TO WRITE TO ZERO
1767	006464	000417				BR	7\$	;GO RESTORE VECTOR
1768	006466	000005			6\$:	RESET		;ISSUE BUS RESET
1769	006470	017737	173030	001126		MOV	QDRVCSR,\$BDDAT	;READ THE CSR
1770	006476	012737	000200	001124		MOV	#200,\$GDDAT	;EXPECT ONLY READY
1771	006504	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;CORRECT?
1772	006512	001404				BEQ	7\$	;BR IF SO
1773	006514	013737	001524	001122		MOV	DRVCSR,\$BDADR	;SET UP CSR ADRES
1774	006522	104016				ERROR	16	;RESET FAILED TO CLR 'NEX'
1775	006524	004737	007622		7\$:	JSR	PC,RSTVEC	;GO RESTORE VECTOR
1776								*****
1777								;*TEST 27 TEST 200 NPR TRANSFERS IN MAINT MODE
1778								*****
1779	006530	000004				†ST27:	SCOPE	
1780	006532	012737	000010	001160		MOV	#10,\$TIMES	;DO 10 ITERATIONS
1781	006540	004537	007602			JSR	RS,SETVEC	;GO SET UP INTR RETURN
1782	006544	006656				2\$		;RETURN TO 2\$ ON INTR
1783	006546	004737	010100			JSR	PC,LDBUF1	;GO SET UP DBUF (SPECIAL COM PATTERN)
1784	006552	012777	014476	172742		MOV	#DBUF,QDRVBAR	;SET UP CURRENT ADRES
1785	006560	012777	177470	172732		MOV	#-200.,QDRVWCR	;SET UP FOR 200 XFER'S
1786	006566	106427	000000			MTPS,	0	;ALLOW INTR
1787	006572	012777	010101	172724		MOV	#10101,QDRVCSR	;SET MAINT, IE & GO
1788	006600	052777	000400	172716		BIS	#400,QDRVCSR	;SET CYCLE
1789	006606	012737	000000	001126		MOV	#0,\$BDDAT	;SET UP A COUNTER
1790	006614	005237	001126		1\$:	INC	\$BDDAT	;COUNT AWAY

```

1791 006620 001375          BNE      1$      ;WAIT TILL DONE - SHOULD INTR BEFORE OVFL0
1792 006622 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRS
1793 006630 012737 010700 001124      MOV      #10700,$GDDAT ;LD EXPECTED
1794 006636 017737 172662 001126      MOV      @DRVCSR,$BDDAT ;READ STATUS
1795 006644 042777 000100 172652      BIC      #100,@DRVCSR ;DISABLE IE
1796 006652 104005          ERROR    5      ;NO INTR AFTER 200 MAINT MODE XFER'S
1797 006654 000420          BR       3$      ;GO RESTORE VECTOR
1798 006656 022626          2$:      CMP      (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1799 006660 004537 007646          JSR      R5,CKSTAT ;GO CHECK STATUS
1800 006664 010700          10700 ;EXPECTED STATUS
1801 006666 000310          200. ;# OF XFER'S
1802 006670 104007          ERROR    7      ;RETURN HERE IF STATUS ER - EXPECTED MAINT,
1803                                     ;CYCLE, READY & IE
1804 006672 000411          BR       3$      ;GO RESTORE VECTOR
1805 006674 104010          ERROR    10     ;RETURN HERE IF WC ER - SHOULD BE 0
1806 006676 000407          BR       3$      ;GO RESTORE VECTOR
1807 006700 104011          ERROR    11     ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
1808 006702 000405          BR       3$      ;GO RESTORE VECTOR
1809 006704 004537 010216          JSR      R5,CKDAT1 ;RETURN HERE IF OK - NOW GO CHECK DATA
1810 006710 000310          200. ;# OF XFER'S TO CHECK
1811 006712 104013          ERROR    13     ;RETURN HERE IF DATA ER - (WITH MAINT SET)
1812 006714 000240          NOP ;RESTORE VECTOR NEXT
1813 006716 004737 007622          3$:      JSR      PC,RSTVEC ;GO RESTORE VECTOR
1814                                     ;*****
1815                                     ;*TEST 30 TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILABLE 4K
1816                                     ;*****
1817 006722 000004          TST30:  SCOPE
1818 006724 012737 000005 001160      MOV      #5,$TIMES ;DO 5 ITERATIONS
1819 006732 004537 007602          JSR      R5,SETVEC ;GO SET UP INTR RETURN
1820 006736 007072          3$      ;RETURN TO 3$ ON INTR
1821 006740 005037 001542          CLR      DBUFP ;GET LOWEST BUFFER ADRS
1822 006744 062737 020000 001542          1$:      ADD      #20000,DBUFP ;POINT TO NEXT 4K
1823 006752 023737 001540 001542          CMP      CORSZ,DBUFP ;IS THE 4K THERE?
1824 006760 001465          BEQ     4$      ;BR IF NOT
1825 006762 004737 010100          JSR      PC,LDBUF1 ;GO SET UP SPECIAL COMPEMENT PATTERN
1826 006766 013777 001542 172526      MOV      DBUFP,@DRVBAR ;SET UP BUFFER ADRS
1827 006774 012777 177470 172516      MOV      #-200.,@DRVWCR ;SET UP FOR 200 XFER'S
1828 007002 106427 000000          MTPS, 0 ;ALLOW INTR
1829 007006 012777 010101 172510      MOV      #10101,@DRVCSR ;SET MAINT, IE & GO
1830 007014 052777 000400 172502      BIS      #400,@DRVCSR ;SET CYCLE
1831 007022 012737 000000 001126      MOV      #0,$BDDAT ;SET UP A COUNTER
1832 007030 005237 001126          2$:      INC      $BDDAT ;COUNT AWAY
1833 007034 001375          BNE     2$      ;SHOULD ALWAYS INTR FROM THIS LOOP
1834 007036 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRA
1835 007044 012737 010700 001124      MOV      #10700,$GDDAT ;LD EXPECTED
1836 007052 017737 172446 001126      MOV      @DRVCSR,$BDDAT ;READ CSR
1837 007060 042777 000100 172436      BIC      #100,@DRVCSR ;DISABLE IE
1838 007066 104005          ERROR    5      ;NO INTR AFTER 200 MAINT MODE XFER'S
1839 007070 000421          BR       4$      ;GO RESTORE VECTOR
1840 007072 022626          3$:      CMP      (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
1841 007074 004537 007646          JSR      R5,CKSTAT ;GO CHECK STATUS
1842 007100 010700          10700 ;EXPECTED STATUS
1843 007102 000310          200. ;# OF XFER'S
1844 007104 104007          ERROR    7      ;RETURN HERE IF STATUS ER - EXPECTED MAINT,

```

```

1845                                     ;CYCLE, READY & IE
1846 007106 000412 BR 4$ ;GO RESTORE VECTOR
1847 007110 104010 ERROR 10 ;RETURN HERE IF WC ER - SHOULD = 0
1848 007112 000410 BR 4$ ;GO RESTORE VECTOR
1849 007114 104011 ERROR 11 ;RETURN HERE IF BAR ER - SHOULD = DBUFF+620
1850 007116 000406 BR 4$ ;GO RESTORE VECTOR
1851 007120 004537 010216 JSR R5,CKDAT1 ;RETURN HERE IF OK - NOW GO CK DATA
1852 007124 000310 200. ;# OF XFER'S TO CK
1853 007126 104013 ERROR 13 ;RETURN HERE IF DATA ER
1854 007130 000401 BR 4$ ;GO RESTORE VECTOR
1855 007132 000704 BR 1$ ;TRY NEXT BANK
1856 007134 012737 014476 001542 4$: MOV #DBUF,DBUFF ;RESTORE BUFFER ADRS TO LOWEST 4K
1857 007142 004737 007622 JSR PC,RSTVEC ;GO RESTORE VECTOR
1858                                     ;*****
1859 ;*TEST 31 TEST THE ADDRESS (I/O) ABILITY TO THE TTY PRINTER CSR
1860 ;*****
1861 007146 000004 TST31: SCOPE
1862 007150 004537 007602 JSR R5,SETVEC ;GO SET UP INTR RETURN
1863 007154 007260 1$ ;RETURN TO 1$ ON INTR
1864 007156 012777 177777 172334 MOV #-1,@DRVWCR ;SET UP WC - 1 XFER
1865 007164 013737 001150 001542 MOV $STPS,DBUFF ;SET UP BUFFER ADRS TO PRINTER CSR ADRS
1866 007172 013777 001150 172322 MOV $STPS,@DRVBAR ;SET UP BUFFER ADRS - FROM PRINTER CSR
1867 007200 106427 000000 MTPS, 0 ;ALLOW INTR
1868 007204 005077 172316 CLR @DRVDBR ;ZERO THE DBR
1869 007210 012777 000161 172306 MOV #161,@DRVCSP ;SET IE, XAD17 & XAD16, & GO
1870 007216 052777 000400 172300 BIS #400,@DRVCSP ;SET CYCLE
1871 007224 013737 001524 001122 MOV DRVCSP,$BDADR ;SET UP CSR ADRS
1872 007232 012737 000760 001124 MOV #760,$GDDAT ;LD EXPECTED STATUS
1873 007240 017737 172260 001126 MOV @DRVCSP,$BDDAT ;READ THE CSR
1874 007246 042777 000100 172250 BIC #100,@DRVCSP ;DISABLE IE
1875 007254 104005 ERROR 5 ;NO INTR ON 1 WD XFER FROM XCSR
1876 007256 000434 BR 2$ ;GO RESTORE VECTOR
1877 007260 022626 007646 1$: CMP (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STK SINCE NO RTI
1878 007262 004537 JSR R5,CKSTAT ;GO CK STATUS
1879 007266 000760 760 ;CSR EXPECTED STATUS
1880 007270 000001 1 ;# OF XFER'S
1881 007272 104007 ERROR 7 ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
1882                                     ;XAD17 & XAD16, RDY & IE
1883 007274 000425 BR 2$ ;GO RESTORE VECTOR
1884 007276 104010 ERROR 10 ;RETURN HERE IF WC ER - EXPECTED 0
1885 007300 000423 BR 2$ ;GO RESTORE VECTOR
1886 007302 104011 ERROR 11 ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
1887 007304 000421 BR 2$ ;GO RESTORE VECTOR
1888 007306 017737 171636 001124 MOV @STPS,$GDDAT ;RETURN HERE IF OK - GET XCSR CONTENTS
1889 007314 017737 172206 001126 MOV @DRVDBR,$BDDAT ;READ DBR
1890 007322 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
1891 007330 001407 BEQ 2$ ;BR IF SO
1892 007332 013737 001526 001122 MOV DRVDBR,$BDADR ;SET UP DBR ADRS
1893 007340 013737 001150 001120 MOV $STPS,$GDADR ;GET ADRS OF DATA (XCSR)
1894 007346 104020 ERROR 20 ;DATA ER FROM TTY PRINTER CSR
1895 007350 012737 014476 001542 2$: MOV #DBUF,DBUFF ;RESTORE BUFFER ADRS TO LOWEST 4K
1896 007356 004737 007622 JSR PC,RSTVEC ;GO RESTORE VECTOR
1897
1898                                     ;*****

```

```

1899
1900
1901 007362 000004
1902 007364 000241
1903 007366 006037 001536
1904 007372 001432
1905 007374 062737 000010 001520
1906 007402 062737 000010 001522
1907 007410 062737 000010 001524
1908 007416 062737 000010 001526
1909 007424 062737 000004 001530
1910 007432 062737 000004 001532
1911 007440 005237 001206
1912 007444 032737 000001 001536
1913 007452 001744
1914 007454 000137 002330
1915
1916
1917
1918
1919
1920
1921
1922
1923 007460
1924 007460 000240
1925 007462 005037 001102
1926 007466 005037 001160
1927 007472 005237 001202
1928 007476 042737 100000 001202
1929 007504 005327
1930 007506 000001
1931 007510 003022
1932 007512 012737
1933 007514 000001
1934 007516 007506
1935 007520 104400 007565
1936 007524 013746 001202
1937 007530 104404
1938 007532 104400 007562
1939 007536 013700 000042
1940 007542 001405
1941 007544 000005
1942 007546 004710
1943 007550 000240
1944 007552 000240
1945 007554 000240
1946 007556
1947 007556 000137
1948 007560 002006
1949 007562 377 377 000
1950 007565 015 042412 042116
1951 007572 050040 051501 020123
1952 007600 000043

```

```

: DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11'S HAVE BEEN TESTED
: *****
NXDEV: SCOPE
NXDEV1: CLC ; CLR CARRY
ROR DMAP ; LOOK FOR NEXT
BEQ $EOP ; BR IF ALL TESTED
ADD #10,DRVWCR ; OFFSET BASE BUS ADRS TO NEXT DRV11B
ADD #10,DRVBAR
ADD #10,DRVCSR
ADD #10,DRVDBR
ADD #4,DRVCT0 ; OFFSET VECTOR ADRS TO NEXT
ADD #4,DRVCT2
INC $UNIT ; COUNT DEVICE
BIT #1,DMAP ; IS IT SELECTED?
BEQ ,NXDEV1 ; BR IF NOT
JMP RESTR ; TEST NEXT

.SBTTL END OF PASS ROUTINE

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

$EOP:
NOP
CLR $STNM ; ZERO THE TEST NUMBER
CLR $TIMES ; ZERO THE NUMBER OF ITERATIONS
INC $PASS ; INCREMENT THE PASS NUMBER
BIC #100000,$PASS ; DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ; LOOP?

$EOPCT: .WORD 1
BGT $DOAGN ; YES
MOV (PC)+,2(PC)+ ; RESTORE COUNTER

$ENDCT: .WORD 1
TYPE $ENDMG ; TYPE "END PASS #"
MOV $PASS,-(SP) ; SAVE $PASS FOR TYPEOUT
TYPDS ; GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $NULL ; TYPE A NULL CHARACTER
$GET42: MOV 2#42,R0 ; GET MONITOR ADDRESS
BEQ $DOAGN ; BRANCH IF NO MONITOR
RESET ; CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ; GO TO MONITOR
NOP ; SAVE ROOM
NOP ; FOR
NOP ; ACT11

$DOAGN: JMP 2(PC)+ ; RETURN

$RTNAD: .WORD START1
$NULL: .BYTE -1,-1,0 ; NULL CHARACTER STRING
$ENDMG: .ASCIZ <15><12>/END PASS #/

```

.SBTTL PROGRAM SUBROUTINES

```

1953
1954
1955
1956
1957
1958
1959
1960
1961 007602 106427 000200
1962 007606 012577 171716
1963 007612 012777 000200 171712
1964 007620 000205
1965
1966
1967
1968
1969
1970 007622 005077 171676
1971 007626 013777 001532 171674
1972 007634 005077 171672
1973 007640 106127 000200
1974 007644 000207
1975
1976
1977
1978
1979
1980
1981
1982 007646 017737 171652 001126
1983 007654 042777 000100 171642
1984 007662 012537 001124
1985 007666 023737 001124 001126
1986 007674 001406
1987 007676 013737 001524 001122
1988 007704 062705 000002
1989 007710 000205
1990 007712 017737 171602 001126 15:
1991 007720 001410
1992 007722 013737 001520 001122
1993 007730 005037 001124
1994 007734 062705 000006
1995 007740 000205
1996 007742 011537 001124 25:
1997 007746 006337 001124
1998 007752 063737 001542 001124
1999 007760 017737 171536 001126
2000 007766 042737 000001 001126
2001 007774 023737 001124 001126
2002 010002 001406
2003 010007 013737 001522 001122
2004 010012 062705 000012
2005 010016 000205
2006 010020 062705 000016 35:

```

```

*****
THIS ROUTINE SETS THE PRIORITY LEVEL FOR NO INTERRUPT -
SETS UP THE DRV118 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 ON INTR
RTS R5 ;EXIT
*****
THIS ROUTINE CLEARS THE DRV118 CSR - RESTORES THE DRV118
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
*****
CHKSTAT: MOV DRVCSR,$BDADR ;READ THE STATUS
BIC #100,DRVCSR ;DISABLE THE IE BIT
MOV (R5)+,$GDDAT ;SET UP EXPECTED STATUS
CMP $GDDAT,$BDAT ;CORRECT?
BEQ 15 ;BR IF SO
MOV DRVCSR,$BDADR ;SET UP CSR ADRS
ADD #2,R5 ;POINT TO THE CSR ER
RTS R5 ;EXIT HERE ON STATUS ERROR
MOV DRVWCR,$BDADR ;GET WC
BEQ 25 ;BR IF ZERO
MOV DRVWCR,$BDADR ;SET UP WCR ADRS
CLR $GDDAT ;EXPECTED 0
ADD #6,R5 ;POINT TO THE WCR ER
RTS R5 ;EXIT HERE ON WCR ER
MOV (R5),$GDDAT ;GET XFER #
ASL $GDDAT ;CONVERT TO WORD
ADD DBUFF,$GDDAT ;POINT TO LAST XFER +2
MOV DRVBAR,$BDADR ;GET BA
BIC #BIT00,$BDAT ;DON'T WANT BIT00
CMP $GDDAT,$BDAT ;CORRECT?
BEQ 35 ;BR IF SO
MOV DRVBAR,$BDADR ;SET UP BAR ADRS
ADD #12,R5 ;POINT TO BAR ER
RTS R5 ;EXIT HERE ON BAR ER
ADD #16,R5 ;ALL OK - POINT TO GOOD EXIT

```

```

2007 010024 000205          RIS  R5          ;EXIT HERE IF NO ERRORS
2008
2009
2010
2011
2012
2013
2014
2015 010026 012703 014476  LDBUF: MOV      #DBUF,R3          ;GET BUFFER ADRS
2016 010032 012704 177776  1S:  MOV      #177776,R4        ;SET UP FLOATING ZERO PATRN
2017 010036 010423          2S:  MOV      R4,(R3)+          ;LOAD IT (FLOATING 0)
2018 010040 005104          COM      R4                    ;MAKE INTO FLOATING 1
2019 010042 022703 015314  CMP      #DBUF+616,R3        ;AT END OF BUFFER?
2020 010046 001003          BNE      4S                    ;BR IF NOT
2021 010050 012713 070707  3S:  MOV      #70707,(R3)       ;LOAD LAST DATVAR (SPECIAL)
2022 010054 000207          RTS      PC                    ;GET OUT
2023 010056 010423          4S:  MOV      R4,(R3)+          ;LOAD IT (FLOATING 1)
2024 010060 022703 015314  CMP      #DBUF+616,R3        ;AT END OF BUFFER?
2025 010064 001771          BEQ      3S                    ;BR IF SO
2026 010066 005104          COM      R4                    ;BACK TO FLOATING ZERO
2027 010070 006304          ASL      R4                    ;SHIFT LEFT
2028 010072 005204          INC      R4                    ;KEEP LSB SET
2029 010074 103356          BCC      1S                    ;GO RESET FLOATING PATRN
2030 010076 000757          BR       2S                    ;GO LOAD NEXT PATRN
2031
2032
2033
2034
2035
2036
2037
2038 010100 013703 001542  LDBUF1: MOV     DBUF,R3          ;GET BUFFER ADRS
2039 010104 010305          MOV     R3,R5                 ;SAVE IN R5
2040 010106 062705 000620  ADD     #620,R5                ;POINT TO END OF BUFFER
2041 010112 012704 177776  1S:  MOV     #177776,R4          ;SET UP FLOATING ZERO PATRN
2042 010116 010423          2S:  MOV     R4,(R3)+          ;LOAD IT (FLOATING 0)
2043 010120 005023          CLR     (R3)+                 ;ZERO NEXT
2044 010122 005104          COM     R4                    ;SET UP FLOATING 1
2045 010124 010423          MOV     R4,(R3)+             ;LOAD IT
2046 010126 005023          CLR     (R3)+                 ;ZERO NEXT
2047 010130 020503          CMP     R5,R3                 ;200 LOCS DONE?
2048 010132 001001          BNE     3S                    ;BR IF NOT
2049 010134 000207          RTS     PC                    ;GET OUT
2050 010136 005104          3S:  COM     R4                    ;BACK TO FLOATING ZERO
2051 010140 006304          ASL     R4                    ;SHIFT LEFT
2052 010142 005204          INC     R4                    ;KEEP LSB SET
2053 010144 103362          BCC     1S                    ;GO RESET FLOATING PATRN
2054 010146 000763          BR      2S                    ;GO FLOAT NEXT PATRN
2055
2056
2057
2058
2059
2060

```

```

:*****
: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
:*****
:*****
: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 (DATA/DATA SEQUENCE) - 200 LOCS
:ARE LOADED WITH THIS PATTERN
:*****
:*****
:THIS ROUTINE CHECKS 200 LOCATIONS IN "DBUF" FOR GOOD TRANSFERED
:DATA (#177377) ON 'DATA' TRANSFERS - IF AN ERROR IS DETECTED
:THE RETURN IS TO CALL +2 - IF NO ERROR THE RETURN IS TO CALL +4
:*****

```

```

2061 010150 012701 014476      CKDAT:  MOV      #DBUF,R1      ;GET BUFFER ADRS
2062 010154 022721 177377      1$:    CMP      #177377,(R1)+    ;DATA OK?
2063 010160 001410              BEQ      2$                    ;BR IF SO
2064 010162 013737 001526 001122  MOV      DRVDBR,$BDAOR      ;SET UP DBR ADRS
2065 010170 014137 001126      MOV      -(R1),$BDDAT      ;GET ACTUAL DATA XFERED
2066 010174 010137 001120      MOV      R1,$GDAOR        ;GET MEMORY ADRS
2067 010200 000207              RTS      PC                  ;RETURN TO ERROR
2068 010202 022701 015316      2$:    CMP      #DBUF+520,R1     ;AT END OF 'DBUF'?
2069 010206 001362              SNE     1$                    ;BR IF MORE
2070 010210 062716 000002      ADD     #2,(SP)             ;ADJUST STACK FOR GOOD RETURN
2071 010214 000207              RTS      PC                  ;GET OUT

```

```

:*****
: 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
:*****

```

```

2080 010216 012500      CKDAT1: MOV      (R5)+,R0      ;GET # OF CHECKS
2081 010220 013702 001542      MOV      DBUF,R2          ;GET BUFFER ADRS
2082 010224 012701 177776      1$:    MOV      #177776,R1     ;SET UP FLOATING ZERO PATRN
2083 010230 005003              CLR      R3                ;R3 SAYS WHEN TO SHIFT PATRN
2084 010232 020122      2$:    CMP      R1,(R2)+      ;DATA OK?
2085 010234 001010              BNE     3$                    ;BR IF NOT
2086 010236 020122              CMP      R1,(R2)+      ;DATA WRITTEN OK?
2087 010240 001006              BNE     3$                    ;BR IF NOT
2088 010242 162700 000002      SUB     #2,R0              ;ACCOUNT FOR TWO ADRS'S
2089 010246 003015              BGT     4$                    ;BR IF MORE
2090 010250 062705 000004      ADD     #4,R5              ;ADJUST FOR GOOD RETURN
2091 010254 000205              RTS     R5                  ;EXIT
2092 010256 014237 001126      3$:    MOV      -(R2),$BDDAT    ;GET BAD DATA
2093 010262 010237 001120      MOV      R2,$GDAOR        ;GET MEM ADRS
2094 010266 010137 001124      MOV      R1,$GDDAT        ;LD EXPECTED DATA
2095 010272 013737 001526 001122  MOV      DRVDBR,$BDAOR    ;SET UP DBR ADRS
2096 010300 000205              RTS     R5                  ;RETURN TO ERROR
2097 010302 005101      4$:    COM     R1              ;NOW EXPECT COMPLEMENT
2098 010304 005103              COM     R3                ;TIME TO SHIFT?
2099 010306 001351              BNE     2$                    ;BR IF NOT
2100 010310 006301              ASL     R1                  ;SHIFT LEFT
2101 010312 005201              INC     R1                  ;KEEP LSB SET
2102 010314 103343              BCC     1$                    ;GC RESET FLOATING PATRN
2103 010316 000745              BR     2$                    ;DO NEXT

```

.SBTTL SYSMAC ROUTINES

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157

```

010320 105737 001157 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
010324 100002 BPL 15 ;; BR IF YES
010326 000000 HALT ;; HALT HERE IF NO TERMINAL
010330 000430 BR 3$ ;; LEAVE
010332 010046 1$: MOV RO, -(SP) ;; SAVE RO
010334 017600 000002 MOV 32(SP), RO ;; GET ADDRESS OF ASCIZ STRING
010340 122737 000001 001214 CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
010346 001011 BNE 62$ ;; NO GO CHECK FOR APT CONSOLE
010350 132737 000100 001215 BITB #APTSPOOL, $ENVM ;; SPOOL MESSAGE TO APT
010356 001405 BEQ 62$ ;; NO GO CHECK FOR CONSOLE
010360 010037 010370 MOV RO, 61$ ;; SETUP MESSAGE ADDRESS FOR APT
010364 004737 010610 JSR PC, $ATY3 ;; SPOOL MESSAGE TO APT
010370 000000 61$: .WORD 0 ;; MESSAGE ADDRESS
010372 132737 000040 001215 62$: BITB #APTCSUP, $ENVM ;; APT CONSOLE SUPPRESSED
010400 001003 BNE 60$ ;; YES, SKIP TYPE OUT
010402 112046 2$: MOVB (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
010404 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
010406 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
010410 012600 60$: MOV (SP)+, RO ;; RESTORE RO
010412 062716 000002 3$: ADD #2, (SP) ;; ADJUST RETURN PC
010416 000002 RTI ;; RETURN
010420 122716 000011 4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
010424 001430 BEQ 8$
010426 122716 000200 CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
010432 001006 BNE 5$
010434 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
010436 104400 TYPE ;; TYPE A CR AND LF
010440 001171 $CRLF
010442 105037 010576 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
010446 000755 BR 2$ ;; GET NEXT CHARACTER
010450 004737 010532 5$: JSR PC, $TYPEC ;; GO TYPE THIS CHARACTER
010454 123726 001156 6$: CMPB $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
010460 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
010462 013746 001154 MOV $NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
;; AND THE NULL CHAR.

```

```

2158 010466 105366 000001 7S:   DECB   1(SP)   ;; DOES A NULL NEED TO BE TYPED?
2159 010472 002770          BLT     5S      ;; BR IF NO--GO POP THE NULL OFF OF STACK
2160 010474 004737 010532   JSR    PC,$TYPEC  ;; GO TYPE A NULL
2161 010500 105337 010576   DECB   $CHARCNT  ;; DO NOT COUNT AS A COUNT
2162 010504 000770          BR      7S      ;; LOOP
2163
2164          ;HORIZONTAL TAB PROCESSOR
2165
2166 010506 112716 000040   9S:   MOVB   #' (SP)   ;; REPLACE TAB WITH SPACE
2167 010512 004737 010532   9S:   JSR    PC,$TYPEC  ;; TYPE A SPACE
2168 010516 132737 000007 010576   BITB   #',$CHARCNT  ;; BRANCH IF NOT AT
2169 010524 001372          SNE     9S      ;; TAB STOP
2170 010526 005726          TST    (SP)+    ;; POP SPACE OFF STACK
2171 010530 000724          BR     2S      ;; GET NEXT CHARACTER
2172 010532 105777 170412   $TYPEC: TSTB   2$TPS  ;; WAIT UNTIL PRINTER IS READY
2173 010536 100375          BPL    $TYPEC
2174 010540 116677 000002 170404   MOVB   2(SP),2$TPB  ;; LOAD CHAR TO BE TYPED INTO DATA REG.
2175 010546 122766 000015 000002   CMPB   #CR,2(SP)   ;; IS CHARACTER A CARRIAGE RETURN?
2176 010554 001003          BNE     1S      ;; BRANCH IF NO
2177 010556 105037 010576   CLRB   $CHARCNT  ;; YES--CLEAR CHARACTER COUNT
2178 010562 000406          BR     $TYPEX  ;; EXIT
2179 010564 122766 000012 000002   1S:   CMPB   #LF,2(SP)  ;; IS CHARACTER A LINE FEED?
2180 010572 001402          BEQ    $TYPEX  ;; BRANCH IF YES
2181 010574 105227          INCB   (PC)+    ;; COUNT THE CHARACTER
2182 010576 000000   $CHARCNT: .WORD 0  ;; CHARACTER COUNT STORAGE
2183 010600 000207   $TYPEX: RTS     PC
2184
2185          .SBTTL  APT COMMUNICATIONS ROUTINE
2186
2187          ;*****
2188 010602 112737 000001 011046   $ATY1: MOVB   #1,$FFLG  ;; TO REPORT FATAL ERROR
2189 010610 112737 000001 011044   $ATY3: MOVB   #1,$MFLG  ;; TO TYPE A MESSAGE
2190 010616 000403          BR     $ATYC
2191 010620 112737 000001 011046   $ATY4: MOVB   #1,$FFLG  ;; TO ONLY REPORT FATAL ERROR
2192 010626          $ATYC:
2193 010626 010046          MOV    R0,-(SP)  ;; PUSH R0 ON STACK
2194 010630 010146          MOV    R1,-(SP)  ;; PUSH R1 ON STACK
2195 010632 105737 011044          TSTR   $MFLG    ;; SHOULD TYPE A MESSAGE?
2196 010636 001450          BEQ    5S      ;; IF NOT: BR
2197 010640 122737 000001 001214   CMPB   #APTENV,$ENV  ;; OPERATING UNDER APT?
2198 010646 001031          BNE     3S      ;; IF NOT: BR
2199 010650 132737 000100 001215   BITB   #APTPOOL,$ENVM  ;; SHOULD SPOOL MESSAGES?
2200 010656 001425          BEQ    3S      ;; IF NOT: BR
2201 010660 017600 000004          MOV    24(SP),R0  ;; GET MESSAGE ADDR.
2202 010664 062766 000002 000004          ADD    #2,4(SP)   ;; BUMP RETURN ADDR.
2203 010672 005737 001174          1S:   TST    $MSGTYPE  ;; SEE IF DONE W/ LAST XMISSION?
2204 010676 001375          BNE     1S      ;; IF NOT: WAIT
2205 010700 010037 001210          MOV    R0,$MSGAD  ;; PUT ADDR IN MAILBOX
2206 010704 105720          2S:   TSTB   (R0)+    ;; FIND END OF MESSAGE
2207 010706 001376          BNE     2S
2208 010710 163700 001210          SJB   $MSGAD,R0  ;; SUB START OF MESSAGE
2209 010714 006200          ASR   R0        ;; GET MESSAGE LNGTH IN WORDS
2210 010716 010037 001212          MOV    R0,$MSGGLT  ;; PUT LENGTH IN MAILBOX
2211 010722 012737 000004 001174          MOV    #4,$MSGTYPE  ;; TELL APT TO TAKE MSG.

```

```

2212 010730 000413          BR          SS
2213 010732 017637 000004 010756 3$:  MOV      24(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
2214 010740 062766 000002 000004      ADD      2,4(SP)      ;;BUMP RETURN ADDRESS
2215 010746 013746 177776      MOV      177776,-(SP)  ;;PUSH 177776 ON STACK
2216 010752 004737 010320      JSR      PC,$TYPE     ;;CALL TYPE MACRO
2217 010756 000000          .WORD    0
2218 010760
2219 010760 105737 011046          4$:     TSTB     $FFLG      ;;SHOULD REPORT FATAL ERROR?
2220 010764 001416          5$:     BEQ      12$          ;;IF NOT: BR
2221 010766 005737 001214          10$:    TST      $ENV        ;;RUNNING UNDER APT?
2222 010772 001413          11$:    BEQ      12$          ;;IF NOT: BR
2223 010774 005737 001174          TST      $MSGTYPE     ;;FINISHED LAST MESSAGE?
2224 011000 001375          SNE     11$          ;;IF NOT: WAIT
2225 011002 017637 000004 001176      MOV      24(SP),$FATAL ;;GET ERROR #
2226 011010 062766 000002 000004      ADD      2,4(SP)      ;;BUMP RETURN ADDR.
2227 011016 005237 001174          INC     $MSGTYPE     ;;TELL APT TO TAKE ERROR
2228 011022 105037 011046          12$:    CLRB     $FFLG     ;;CLEAR FATAL FLAG
2229 011026 105037 011045          CLRB     $LFLG     ;;CLEAR LOG FLAG
2230 011032 105037 011044          CLRB     $MFLG     ;;CLEAR MESSAGE FLAG
2231 011036 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
2232 011040 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0
2233 011042 000207          RTS      PC          ;;RETURN
2234 011044          000      $MFLG: .BYTE 0      ;;MESSG. FLAG
2235 011045          000      $LFLG: .BYTE 0      ;;LOG FLAG
2236 011046          000      $FFLG: .BYTE 0      ;;FATAL FLAG
2237          011050          .EVEN
2238          000200          APTSIZE=200
2239          000001          APTENV=001
2240          000100          APTSPool=100
2241          000040          APTCSUP=040
2242          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
2243
2244 *****
2245 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2246 *OCTAL (ASCII) NUMBER AND TYPE IT.
2247 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2248 *CALL:
2249 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2250 *      TYPOS      ;;CALL FOR TYPEOUT
2251 *      .BYTE    N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2252 *      .BYTE    M              ;;M=1 OR 0
2253 *                                  ;;1=TYPE LEADING ZEROS
2254 *                                  ;;0=SUPPRESS LEADING ZEROS
2255
2256 *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2257 *$TYPOS OR $TYPOC
2258 *CALL:
2259 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2260 *      TYPON     ;;CALL FOR TYPEOUT
2261
2262 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2263 *CALL:
2264 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2265 *      TYPOC     ;;CALL FOR TYPEOUT

```

2266											
2267	011050	017646	000000		\$TYPOS:	MOV	3(SP),-(SP)	::	PICKUP THE MODE		
2268	011054	116637	000001	011273		MOVB	1(SP),%SOFILL	::	LOAD ZERO FILL SWITCH		
2269	011062	112637	011275			MOVB	(SP)+,%SOMODE+1	::	NUMBER OF DIGITS TO TYPE		
2270	011066	062716	0C0002			ADD	%2,(SP)	::	ADJUST RETURN ADDRESS		
2271	011072	000406				BR	\$TYPON				
2272	011074	112737	000001	011273	\$TYPOC:	MOVB	%1,%SOFILL	::	SET THE ZERO FILL SWITCH		
2273	011102	112737	000006	011275		MOVB	%6,%SOMODE+1	::	SET FOR SIX(6) DIGITS		
2274	011110	112737	000005	011272	\$TYPON:	MOVB	%5,%SOCNT	::	SET THE ITERATION COUNT		
2275	011116	010346				MOV	R3,-(SP)	::	SAVE R3		
2276	011120	010446				MOV	R4,-(SP)	::	SAVE R4		
2277	011122	010546				MOV	R5,-(SP)	::	SAVE R5		
2278	011124	113704	011275			MOVB	%SOMODE+1,R4	::	GET THE NUMBER OF DIGITS TO TYPE		
2279	011130	005404				NEG	R4				
2280	011132	062704	000006			ADD	%6,R4	::	SUBTRACT IT FOR MAX. ALLOWED		
2281	011136	110437	011274			MOVB	R4,%SOMODE	::	SAVE IT FOR USE		
2282	011142	113704	011273			MOVB	%SOFILL,R4	::	GET THE ZERO FILL SWITCH		
2283	011146	016605	000012			MOV	12(SP),R5	::	PICKUP THE INPUT NUMBER		
2284	011152	005003				CLR	R3	::	CLEAR THE OUTPUT WORD		
2285	011154	006105			1\$:	ROL	R5	::	ROTATE MSB INTO "C"		
2286	011156	000404				BR	3\$	::	GO DO MSB		
2287	011160	006105			2\$:	ROL	R5	::	FORM THIS DIGIT		
2288	011162	006105				ROL	R5				
2289	011164	006105				ROL	R5				
2290	011166	010503				MOV	R5,R3				
2291	011170	006103			3\$:	ROL	R3	::	GET LSB OF THIS DIGIT		
2292	011172	105337	011274			DECB	%SOMODE	::	TYPE THIS DIGIT?		
2293	011176	100016				BPL	7\$	::	BR IF NO		
2294	011200	042703	177770			BIC	%177770,R3	::	GET RID OF JUNK		
2295	011204	001002				BNE	4\$	::	TEST FOR 0		
2296	011206	005704				TST	R4	::	SUPPRESS THIS 0?		
2297	011210	001403				BEQ	5\$	::	BR IF YES		
2298	011212	005204			4\$:	INC	R4	::	DON'T SUPPRESS ANYMORE 0'S		
2299	011214	052703	000060			BIS	%'0,R3	::	MAKE THIS DIGIT ASCII		
2300	011220	052703	000040		5\$:	BIS	%'1,R3	::	MAKE ASCII IF NOT ALREADY		
2301	011224	110337	011270			MOVB	R3,%\$	::	SAVE FOR TYPING		
2302	011230	104400	011270			TYPE	%8\$	::	GO TYPE THIS DIGIT		
2303	011234	105337	011272		7\$:	DECB	%SOCNT	::	COUNT BY 1		
2304	011240	003347				BGT	2\$	::	BR IF MORE TO DO		
2305	011242	002402				BLT	6\$	::	BR IF DONE		
2306	011244	005204				INC	R4				
2307	011246	000744				BR	2\$	::	INSURE LAST DIGIT ISN'T A BLANK		
2308	011250	012605			6\$:	MOV	(SP)+,R5	::	GO DO THE LAST DIGIT		
2309	011252	012604				MOV	(SP)+,R4	::	RESTORE R5		
2310	011254	012603				MOV	(SP)+,R3	::	RESTORE R4		
2311	011256	016666	000002	000004		MOV	(SP)+,R3	::	RESTORE R3		
2312	011264	012616				MOV	2(SP),4(SP)	::	SET THE STACK FOR RETURNING		
2313	011266	000002				RTI		::	RETURN		
2314	011270	000			8\$:	.BYTE	0	::	STORAGE FOR ASCII DIGIT		
2315	011271	000				.BYTE	0	::	TERMINATOR FOR TYPE ROUTINE		
2316	011272	000			\$SOCNT:	.BYTE	0	::	OCTAL DIGIT COUNTER		
2317	011273	000			\$SOFILL:	.BYTE	0	::	ZERO FILL SWITCH		
2318	011274	000000			\$SOMODE:	.WORD	0	::	NUMBER OF DIGITS TO TYPE		
2319					.SBTTL	CONVERT	BINARY TO DECIMAL AND TYPE ROUTINE				

```

2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331 011276
2332 011276 010046
2333 011300 010146
2334 011302 010246
2335 011304 010346
2336 011306 010546
2337 011310 012746 020200
2338 011314 016605 000020
2339 011320 100004
2340 011322 005405
2341 011324 112766 000055 000001
2342 011332 005000 1$:
2343 011334 012703 011512
2344 011340 112723 000040
2345 011344 005002 2$:
2346 011346 016001 011502
2347 011352 160105 3$:
2348 011354 002402
2349 011356 005202
2350 011360 000774
2351 011362 060105 4$:
2352 011364 005702
2353 011366 001002
2354 011370 105716
2355 011372 100407
2356 011374 106316 5$:
2357 011376 103003
2358 011400 116663 000001 177777
2359 011406 052702 000060 6$:
2360 011412 052702 000040 7$:
2361 011416 110223
2362 011420 005720
2363 011422 020027 000010
2364 011426 002746
2365 011430 003002
2366 011432 010502
2367 011434 000764
2368 011436 105726 8$:
2369 011440 100003
2370 011442 116663 177777 177776
2371 011450 105013 9$:
2372 011452 012605
2373 011454 012603

```

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.
*CALL:
*   MOV   NUM,-(SP)   ;;PUT THE BINARY NUMBER ON THE STACK
*   TYPDS   ;;GO TO THE ROUTINE
$TYPDS:
MOV   R0,-(SP)   ;;PUSH R0 ON STACK
MOV   R1,-(SP)   ;;PUSH R1 ON STACK
MOV   R2,-(SP)   ;;PUSH R2 ON STACK
MOV   R3,-(SP)   ;;PUSH R3 ON STACK
MOV   R5,-(SP)   ;;PUSH R5 ON STACK
MOV   #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
MOV   20(SP),R5   ;;GET THE INPUT NUMBER
BPL   1$         ;;BR IF INPUT IS POS.
NEG   R5         ;;MAKE THE BINARY NUMBER POS.
MOVB  #'-,1(SP)  ;;MAKE THE ASCII NUMBER NEG.
CLR   R0         ;;ZERO THE CONSTANTS INDEX
MOV   #SDBLK,R3  ;;SETUP THE OUTPUT POINTER
MOVB  #' ,(R3)+  ;;SET THE FIRST CHARACTER TO A BLANK
CLR   R2         ;;CLEAR THE BCD NUMBER
MOV   $DTBL(R0),R1 ;;GET THE CONSTANT-
SUB   R1,R5      ;;FORM THIS BCD DIGIT
BLT   4$         ;;BR IF DONE
INC   R2         ;;INCREASE THE BCD DIGIT BY 1
BR    3$
4$:   ADD   R1,R5   ;;ADD BACK THE CONSTANT
TST   R2         ;;CHECK IF BCD DIGIT=0
BNE   5$         ;;FALL THROUGH IF 0
TSTB  (SP)       ;;STILL DOING LEADING 0'S?
BMI   7$         ;;BR IF YES
ASLB  (SP)       ;;MSD?
BCC   6$         ;;BR IF NO
MOVB  1(SP),-1(R3) ;;YES--SET THE SIGN
BIS   #'0,R2     ;;MAKE THE BCD DIGIT ASCII
BIS   #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB  R2,(R3)+   ;;PUT THIS CHARACTER-IN THE OUTPUT BUFFER
TST   (R0)+     ;;JUST INCREMENTING
CMP   R0,#10    ;;CHECK THE TABLE INDEX
BLT   2$         ;;GO DO THE NEXT DIGIT
BGT   8$         ;;GO TO EXIT
MOV   R5,R2     ;;GET THE LSD
BR    6$         ;;GO CHANGE TO ASCII
TSTB  (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
BPL   9$         ;;BR IF NO
MOVB  -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
CLRB  (R3)      ;;SET THE TERMINATOR
MOV   (SP)+,R5  ;;POP STACK INTO R5
MOV   (SP)+,R3  ;;POP STACK INTO R3

```

```

2374 011456 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
2375 011460 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
2376 011462 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
2377 011464 104400 011512 000004  TYPE      $DBLK      ;;NOW TYPE THE NUMBER
2378 011470 016666 000002 000004  MOV      2(SP),4(SP)  ;;ADJUST THE STACK
2379 011476 012616      MOV      (SP)+,(SP)
2380 011500 000002      RTI                          ;;RETURN TO USER
2381 011502 023420      $C^BL: 10000.
2382 011504 001750      1000.
2383 011506 000144      100.
2384 011510 000012      10.
2385 011512 000004      $DBLK: .BLKW 4
2386                          .SBTTL  ERROR HANDLER ROUTINE
2387
2388
2389      ;;*****
2390      ;;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2391      ;;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2392      ;;AND GO TO SWRCK ON ERROR
2393      ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2394      ;;SW15=1      HALT ON ERROR
2395      ;;SW13=1      INHIBIT ERROR TYPEOUTS
2396      ;;SW10=1      BELL ON ERROR
2397      ;;SW09=1      LOOP ON ERROR
2398      ;;CALL
2399      ;;      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2400
2401 011522      $ERROR:
2402 011522 104406      CKSWR      CKSWR      ;;TEST FOR CHANGE IN 'SOFT-SWR
2403 011524 104406      CKSWR      CKSWR      ;GO LOOK FOR SWR CHANGE
2404 011526 105237 001103 7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
2405 011532 001775      BEQ      7$          ;;DON'T LET THE FLAG GO TO ZERO
2406 011534 013777 001102 167400  MOV      $STNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
2407 011542 032777 002000 167370  BIT      #BIT10,$SWR    ;;BELL ON ERROR?
2408 011550 001402      BEQ      1$          ;;NO - SKIP
2409 011552 104400 001164      TYPE      $BELL      ;;RING BELL
2410 011556 005237 001112 1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
2411 011562 011637 001116      MOV      (SP),$ERRPC  ;;GET ADDRESS OF ERROR INSTRUCTION
2412 011566 162737 000002 001116  SUB      #2,$ERRPC
2413 011574 117737 167316 001114  MOVB     2,$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
2414 011602 032777 020000 167330  BIT      #BIT13,$SWR    ;;SKIP TYPEOUT IF SET
2415 011610 001004      BNE      20$         ;;SKIP TYPEOUTS
2416 011612 004737 011712      JSR      PC,$SWRCK    ;;GO TO USER ERROR ROUTINE
2417 011616 104400 001171      TYPE      $CRLF
2418 011622      20$:      CMPB     #APTENV,$ENV  ;;RUNNING IN APT MODE
2419 011630 001007      BNE      2$          ;;NO SKIP APT ERROR REPORT
2420 011632 113737 001114 011644  MOVB     $ITEMB,21$    ;;SET ITEM NUMBER AS ERROR NUMBER
2421 011640 004737 010620      JSR      PC,$ATY4     ;;REPORT FATAL ERROR TO APT
2422 011644      000      21$:      .BYTE    0
2423 011645      000      .BYTE    0
2424 011646 000777      22$:      BR       22$         ;;APT ERROR LOOP
2425 011650 005777 167264 2$:      TST     2$SWR        ;;HALT ON ERROR
2426 011654 100002      BPL     3$          ;;SKIP IF CONTINUE
2427 011656 000000      HALT     3$         ;;HALT ON ERROR!

```

```

2428 011660 104406          CKSWR          ;; TEST FOR CHANGE IN SOFT-SWR
2429 011662 032777 001000 167250 3$: BIT      #BIT09,DSWR  ;; LOOP ON ERROR SWITCH SET?
2430 011670 001402          BEQ      4$      ;; BR IF NO
2431 011672 013716 001110  MOV      $LPERR,(SP)  ;; FUDGE RETURN FOR LOOPING
2432 011676 005737 001162  4$: TST      $ESCAPE  ;; CHECK FOR AN ESCAPE ADDRESS
2433 011702 001402          BEQ      5$      ;; BR IF NONE
2434 011704 013716 001162  MOV      $ESCAPE,(SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE
2435 011710          5$:
2436 011710 000002          RTI          ;; RETURN
2437          ;; *****
2438          ;; GO TYPE ERROR
2439          ;; GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
2440          ;; *****
2441 011712 113737 001102 001534 SWRCK: MOVB   $STNM,TSTNUM  ;; SET UP TEST # ON ER
2442 011720 004737 011730  JSR      PC,$ERRTYP  ;; GO TYPE ERROR
2443 011724 104406          CKSWR          ;; GO LOOK FOR SWR CHANGE
2444 011726 000207          RTS      PC      ;; RETURN TO ERROR HANDLER
2445          .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
2446
2447          ;; *****
2448          ;; *THIS ROUTINE USES THE "ITEM CONTROL BYTE"- ($ITEMB) TO DETERMINE WHICH
2449          ;; *ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" ($ERRTB),
2450          ;; *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
2451
2452          $ERRTYP:
2453 011730 011730 104400 001171  TYPE     $SCRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
2454 011734 010046          MOV      RO,-(SP)   ;; SAVE RO
2455 011736 005000          CLR      RO        ;; PICKUP THE ITEM INDEX
2456 011740 153700 001114  BISB    2*$ITEMB,RO
2457 011744 001004          BNE     1$
2458          ;; IF ITEM NUMBER IS ZERO, JUST
2459 011746 013746 001116  MOV      $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
2460          ;; ERROR ADDRESS
2461 011752 104401          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2462 011754 000426          BR      6$
2463 011756 005300 1$: DEC      RO
2464 011760 006300          ASL     RO
2465 011762 006300          ASL     RO
2466 011764 006300          ASL     RO
2467 011766 062700 001320  ADD     #$ERRTB,RO  ;; FORM TABLE POINTER
2468 011772 012037 012002  MOV     (RO)+,2$   ;; PICKUP "ERROR MESSAGE" POINTER
2469 011776 001404          BEQ     3$
2470 012000 104400          TYPE          ;; TYPE THE "ERROR MESSAGE"
2471 012002 000000 2$: .WORD   0        ;; "ERROR MESSAGE" POINTER GOES HERE
2472 012004 104400          TYPE     $SCRLF  ;; "CARRIAGE RETURN" & "LINE FEED"
2473 012010 012037 012020 3$: MOV     (RO)+,4$  ;; PICKUP "DATA HEADER" POINTER
2474 012014 001404          BEQ     5$
2475 012016 104400          TYPE          ;; TYPE THE "DATA HEADER"
2476 012020 000000 4$: .WORD   0        ;; "DATA HEADER" POINTER GOES HERE
2477 012022 104400 001171  TYPE     $SCRLF  ;; "CARRIAGE RETURN" & "LINE FEED"
2478 012026 011000 5$: MOV     (RO),RO   ;; PICKUP "DATA TABLE" POINTER
2479 012030 001004          BNE     7$
2480 012032 012600 6$: MOV     (SP)+,RO  ;; GO TYPE THE DATA
2481 012034 104400 001171  TYPE     $SCRLF  ;; RESTORE RO
                ;; "CARRIAGE RETURN" & "LINE FEED"

```

```

2482 012040 000207          RIS      PC          ;;RETURN
2483 012042          7$:          MOV      2(R0)+,-(SP)      ;;SAVE 2(R0)+ FOR TYPEOUT
2484 012042 013046          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2485 012044 104401          TST      (R0)          ;;IS THERE ANOTHER NUMBER?
2486 012046 005710          BEQ      6$           ;;BR IF NO
2487 012050 001770          TYPE     8$           ;;TYPE TWO(2) SPACES
2488 012052 104400 012060          BR       7$           ;;LOOP
2489 012056 000771          .ASCIZ  / /          ;;TWO(2) SPACES
2490 012060 020040 000          .EVEN
2491 012064 012064          .SBTTL SCOPE HANDLER ROUTINE
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506 012064          $SCOPE:
2507 012064 104406          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
2508 012066 104406          CKSWR          ;;GO LOOK FOR SWR CHANGE
2509 012070 032777 040000 167042 1$:      BIT      #BIT14,2SWR      ;;LOOP ON PRESENT TEST?
2510 012076 001114          BNE      $OVER          ;;YES IF SW14=1
2511
2512 012100 000416          $XTSTR: BR      6$           *****START OF CODE FOR THE XOR TESTER*****
2513
2514 012102 013746 000004          MOV      2#ERRVEC,-(SP)      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
2515 012106 012737 012126 000004          MOV      #55,2#ERRVEC      THIS INSTRUCTION TO A "NOP" (NOP=240)
2516 012114 005737 177060          TST      2#177060          SAVE THE CONTENTS OF THE ERROR VECTOR
2517 012120 012637 000004          MOV      (SP)+,2#ERRVEC      SET FOR TIMEOUT
2518 012124 000463          BR      $SVLAD          TIME OUT ON XOR?
2519 012126 022626          5$:      CMP      (SP)+,(SP)+      RESTORE THE ERROR VECTOR
2520 012130 012637 000004          MOV      (SP)+,2#ERRVEC      GO TO THE NEXT TEST
2521 012134 000423          BR       7$           CLEAR THE STACK AFTER A TIME OUT
2522 012136          6$:; *****END OF CODE FOR THE XOR TESTER*****      RESTORE THE ERROR VECTOR
2523 012136 032777 000400 166774          BIT      #BIT08,2SWR      LOOP ON SPEC. TEST?
2524 012144 001404          BEQ      2$           BR IF NO
2525 012146 127737 166766 001102          CMPB    2SWR,$STNM      ON THE RIGHT TEST? SWR<7:0>
2526 012154 001465          BEQ      $OVER          BR IF YES
2527 012156 105737 001103          2$:      TSTB    $ERFLG          HAS AN ERROR OCCURRED?
2528 012162 001421          BEQ      3$           BR IF NO
2529 012164 123737 001115 001103          CMPB    $ERMAX,$ERFLG      MAX. ERRORS FOR THIS TEST OCCURRED?
2530 012172 101015          BHI      3$           BR IF NO
2531 012174 032777 001000 166736          BIT      #BIT09,2SWR      LOOP ON ERROR?
2532 012202 001404          BEQ      4$           BR IF NO
2533 012204 013737 001110 001106 7$:      MOV      $LPERR,$LPADR      SET LOOP ADDRESS TO LAST SCOPE
2534 012212 000446          BR       $OVER
2535 012214 105037 001103          4$:      CLRB    $ERFLG          ;;ZERO THE ERROR FLAG

```

```

2536 012220 005037 001160 CLR $TIMES ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
2537 012224 000415 BR 1$ ;; ESCAPE TO THE NEXT TEST
2538 012226 032777 004000 166704 3$: BIT #BIT11,$SWR ;; INHIBIT ITERATIONS?
2539 012234 001011 BNE 1$ ;; BR IF YES
2540 012236 005737 001202 TST $PASS ;; IF FIRST PASS OF PROGRAM
2541 012242 001406 BEQ 1$ ;; INHIBIT ITERATIONS
2542 012244 005237 001104 INC $ICNT ;; INCREMENT ITERATION COUNT
2543 012250 023737 001160 001104 CMP $TIMES,$ICNT ;; CHECK THE NUMBER OF ITERATIONS MADE
2544 012256 002024 BGE $OVER ;; BR IF MORE ITERATION REQUIRED
2545 012260 012737 000001 001104 1$: MOV #1,$ICNT ;; REINITIALIZE THE ITERATION COUNTER
2546 012266 013737 012344 001160 MOV $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
2547 012274 105237 001102 $SVLAD: INCB $STNM ;; COUNT TEST NUMBERS
2548 012300 113737 001102 001200 MOVB $STNM,$TESTN ;; SET TEST NUMBER IN APT MAILBOX
2549 012306 011637 001106 MOV (SP),$LPADR ;; SAVE SCOPE LOOP ADDRESS
2550 012312 011637 001110 MOV (SP),$LPERR ;; SAVE ERROR LOOP ADDRESS
2551 012316 005037 001162 CLR $ESCAPE ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2552 012322 112737 000001 001115 MOVB #1,$ERMAX ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2553 012330 013777 001102 166604 $OVER: MOV $STNM,$DISPLAY ;; DISPLAY TEST NUMBER
2554 012336 013716 001106 MOV $LPADR,(SP) ;; FUDGE RETURN ADDRESS
2555 012342 000002 RTI ;; FIXES PS
2556 012344 003720 $MXCNT: 2000. ;; MAX. NUMBER OF ITERATIONS
2557 .SBTTL TTY INPUT ROUTINE
2558
2559 ;:*****
2560 .ENABL LSB
2561
2562 ;:*****
2563 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2564 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2565 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2566 ;*WHEN OPERATING IN TTY FLAG MODE.
2567 012346 022737 000176 001140 $CKSWR: CMP #SWREG,$SWR ;; IS THE SOFT-SWR SELECTED?
2568 012354 001074 BNE 15$ ;; BRANCH IF NO
2569 012356 105777 166562 TSTB $TKS ;; CHAR THERE?
2570 012362 100071 BPL 15$ ;; IF NO, DON'T WAIT AROUND
2571 012364 117746 166556 MOVB $TKB,-(SP) ;; SAVE THE CHAR
2572 012370 042716 177600 BIC #C177,(SP) ;; STRIP-OFF THE ASCII
2573 012374 022726 000007 CMP #7,(SP)+ ;; IS IT A CONTROL G?
2574 012400 001062 BNE 15$ ;; NO, RETURN TO USER
2575 012402 123727 001134 000001 CMPB $AUTOB,#1 ;; ARE WE RUNNING IN AUTO-MODE?
2576 012410 001456 BEQ 15$ ;; BRANCH IF YES
2577
2578 012412 104400 013073 $GTSWR: TYPE , $CNTLG ;; ECHO THE CONTROL-G (↑G)
2579 012416 104400 013100 TYPE $MSWR ;; TYPE CURRENT CONTENTS
2580 012422 013746 000176 MOV $SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
2581 012426 104401 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2582 012430 104400 013111 TYPE , $MNEW ;; PROMPT FOR NEW SWR
2583 012434 005046 19$: CLR -(SP) ;; CLEAR COUNTER
2584 012436 005046 CLR -(SP) ;; THE NEW SWR
2585 012440 105777 166500 7$: TSTB $TKS ;; CHAR THERE?
2586 012444 100375 BPL 7$ ;; IF NOT TRY AGAIN
2587
2588 012446 117746 166474 MOVB $TKB,-(SP) ;; PICK UP CHAR
2589 012452 042716 177600 BIC #C177,(SP) ;; MAKE IT 7-BIT ASCII

```

```

2590
2591
2592
2593 012456 021627 000025      9$:    CMP      (SP),#25      ;; IS IT A CONTROL-U?
2594 012462 001005              BNE      10$      ;; BRANCH IF NOT
2595 012464 104400 013066      TYPE    $CNTLU    ;; YES, ECHO CONTROL-U (↑U)
2596 012470 062706 000006      20$:   ADD      #6,SP    ;; IGNORE PREVIOUS INPUT
2597 012474 000757              BR       19$      ;; LET'S TRY IT AGAIN
2598
2599
2600 012476 021627 000015      10$:   CMP      (SP),#15    ;; IS IT A <CR>?
2601 012502 001022              BNE      16$      ;; BRANCH IF NO
2602 012504 005766 000004      TST     4(SP)     ;; YES, IS IT THE FIRST CHAR?
2603 012510 001403              BEQ      11$      ;; BRANCH IF YES
2604 012512 016677 000002 166420  MOV     2(SP),@SWR ;; SAVE NEW SWR
2605 012520 062706 000006      11$:   ADD      #6,SP    ;; CLEAR UP STACK
2606 012524 104400 001171      14$:   TYPE    $CRLF    ;; ECHO <CR> AND <LF>
2607 012530 123727 001135 000001  CMPB   $INTAG,#1  ;; RE-ENABLE TTY KBD INTERRUPTS?
2608 012536 001003              BNE      15$      ;; BRANCH IF NOT
2609 012540 012777 000100 166376  MOV     #100,@$TKS ;; RE-ENABLE TTY KBD INTERRUPTS
2610 012546 000002              RTI                    ;; RETURN
2611 012550 004737 010532      16$:   JSR     PC,$TYPEC   ;; ECHO CHAR
2612 012554 021627 000060      CMP     (SP),#60   ;; CHAR < 0?
2613 012560 002420              BLT     18$      ;; BRANCH IF YES
2614 012562 021627 000067      CMP     (SP),#67   ;; CHAR > 7?
2615 012566 003015              BGT     18$      ;; BRANCH IF YES
2616 012570 042726 000060      BIC     #60,(SP)+  ;; STRIP-OFF ASCII
2617 012574 005766 000002      TST     2(SP)     ;; IS THIS THE FIRST CHAR
2618 012600 001403              BEQ     17$      ;; BRANCH IF YES
2619 012602 006316              ASL     (SP)      ;; NO, SHIFT PRESENT
2620 012604 006316              ASL     (SP)      ;; CHAR OVER TO MAKE
2621 012606 006316              ASL     (SP)      ;; ROOM FOR NEW ONE.
2622 012610 005266 000002      17$:   INC     2(SP)     ;; KEEP COUNT OF CHAR
2623 012614 056616 177776      BIS     -2(SP),(SP) ;; SET IN NEW CHAR
2624 012620 000707              BR      7$        ;; GET THE NEXT ONE
2625 012622 104400 001170      18$:   TYPE    $QUES    ;; TYPE ?<CR><LF>
2626 012626 000720              BR      20$      ;; SIMULATE CONTROL-U
2627 .DSABL  LSB
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638 012630 011646 000004 000002 $RDCHR: MOV     (SP),-(SP) ;; PUSH DOWN THE PC
2639 012632 016666 000004 000002 1$:    MOV     4(SP),2(SP) ;; SAVE THE PS
2640 012640 105777 166300      TSTB   @$TKS     ;; WAIT FOR
2641 012644 100375              BPL     1$        ;; A CHARACTER
2642 012646 117766 166274 000004  MOVB   @$TKB,4(SP) ;; READ THE TTY
2643 012654 042766 177600 000004  BIC     #↑C<177>,4(SP) ;; GET RID OF JUNK IF ANY

```

```

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

```

```

2644 012662 026627 000004 000023      CMP      4(SP),#23      :: IS IT A CONTROL-S?
2645 012670 001013                    BNE      3$           :: BRANCH IF NO
2646 012672 105777 166246          2$:      TSTB      2$TKS      :: WAIT FOR A CHARACTER
2647 012676 100375                    BPL      2$           :: LOOP UNTIL ITS THERE
2648 012700 117746 166242      MOVB     2$TKB,-(SP)    :: GET CHARACTER
2649 012704 042716 177600      BIC     #1C177,(SP)    :: MAKE IT 7-BIT ASCII
2650 012710 022627 000021      CMP     (SP)+,#21     :: IS IT A CONTROL-Q?
2651 012714 001366                    BNE     2$           :: IF NOT DISCARD IT
2652 012716 000750                    BR      1$           :: YES, RESUME
2653 012720 026627 000004 000140      3$:      CMP     4(SP),#140  :: IS IT UPPER CASE?
2654 012726 002407                    BLT     4$           :: BRANCH IF YES
2655 012730 026627 000004 000175      CMP     4(SP),#175   :: IS IT A SPECIAL CHAR?
2656 012736 003003                    BGT     4$           :: BRANCH IF YES
2657 012740 042766 000040 000004      BIC     #40,4(SP)    :: MAKE IT UPPER CASE
2658 012746 000002          4$:      RTI              :: GO BACK TO USER
2659                                     :: *****
2660                                     :: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2661                                     :: *CALL:
2662                                     :: *      RDLIN          :: INPUT A STRING FROM THE TTY
2663                                     :: *      RETURN HERE   :: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2664                                     :: *                                     :: TERMINATOR WILL BE A BYTE OF ALL C'S
2665
2666 012750 010346          $RDLIN: MOV     R3,-(SP)    :: SAVE R3
2667 012752 012703 013056      1$:      MOV     #STTYIN,R3  :: GET ADDRESS
2668 012756 022703 013056      2$:      CMP     #STTYIN+8.,R3 :: BUFFER FULL?
2669 012762 101405                    BLOS    4$           :: BR IF YES
2670 012764 104407                    RDCHR   :: GO READ ONE CHARACTER FROM THE TTY
2671 012766 112613      MOVB     (SP)+,(R3)  :: GET CHARACTER
2672 012770 122713 000177      10$:     CMPB    #177,(R3)    :: IS IT A RUBOUT
2673 012774 001003                    BNE     3$           :: SKIP IF NOT
2674 012776 104400 001170      4$:      TYPE    $QUES      :: TYPE A '?'
2675 013002 000763                    BR      1$           :: CLEAR THE BUFFER AND LOOP
2676 013004 111337 013054      3$:      MOVB     (R3),9$     :: ECHO THE CHARACTER
2677 013010 104400 013054                    TYPE    9$
2678 013014 122723 000015      CMPB    #15,(R3)+    :: CHECK FOR RETURN
2679 013020 001356                    BNE     2$           :: LOOP IF NOT RETURN
2680 013022 105063 177777      CLRB    -1(R3)       :: CLEAR RETURN (THE 15)
2681 013026 104400 001172      TYPE    $LF          :: TYPE A LINE FEED
2682 013032 012603      MOV     (SP)+,R3     :: RESTORE R3
2683 013034 011546      MOV     (SP)-,(SP)   :: ADJUST THE STACK AND PUT ADDRESS OF THE
2684 013036 016666 000004 000002      MOV     4(SP),2(SP)  :: FIRST ASCII CHARACTER ON IT
2685 013044 012766 013056 000004      MOV     #STTYIN,4(SP)
2686 013052 000002          RTI
2687 013054          9$:      .BYTE    0          :: RETURN
2688 013055          .BYTE    0          :: STORAGE FOR ASCII CHAR. TO TYPE
2689 013056 000010          .BLKB   8          :: TERMINATOR
2690 013066 052536 005015          $TTYIN: .ASCIZ  /?U/(15)<12> :: RESERVE 8 BYTES FOR TTY INPUT
2691 013073          136 006507 000012      $CNTLG: .ASCIZ  /?G/(15)<12> :: CONTROL "U"
2692 013100 005015 053523 020122      $MSWR:  .ASCIZ  <15><12>/SWR = / :: CONTROL "G"
2693 013106 020075          000
2694 013111          040 047040 053505      $MNEW:  .ASCIZ  / NEW =
2695 013116 036440 000040
2696
2697 .SBTTL POWER DOWN AND UP ROUTINES

```

```

2598
2699
2700 013122 012737 013266 000024
2701 013130 012737 000340 000026
2702 013136 010046
2703 013140 010146
2704 013142 010246
2705 013144 010346
2706 013146 010446
2707 013150 010546
2708 013152 017746 165762
2709 013156 010637 013272
2710 013162 012737 013174 000024
2711 013170 000000
2712 013172 000776
2713
2714
2715
2716 013174 012737 013266 000024
2717 013202 013706 013272
2718 013206 005037 013272
2719 013212 005237 013272
2720 013216 001375
2721 013220 012677 165714
2722 013224 012605
2723 013226 012604
2724 013230 012603
2725 013232 012602
2726 013234 012601
2727 013236 012600
2728 013240 012737 013122 000024
2729 013246 012737 000340 000026
2730 013254 104400
2731 013256 013274
2732 013260 012716
2733 013262 002330
2734 013264 000002
2735 013266 000000
2736 013270 000776
2737 013272 000000
2738 013274 005015 042522 052123
2739 013302 051101 042524 020104
2740 013310 051106 046517 050040
2741 013316 051127 043040 044501
2742 013324 000114
2743
2744
2745
2746
2747
2748
2749
2750
2751

```

\*\*\*\*\*

POWER DOWN ROUTINE

```

$PWRDN: MOV $SILLUP, @PWRVEC ;; SET FOR FAST UP
MOV @340, @PWRVEC+2 ;; PRIO:7
MOV R0, -(SP) ;; PUSH R0 ON STACK
MOV R1, -(SP) ;; PUSH R1 ON STACK
MOV R2, -(SP) ;; PUSH R2 ON STACK
MOV R3, -(SP) ;; PUSH R3 ON STACK
MOV R4, -(SP) ;; PUSH R4 ON STACK
MOV R5, -(SP) ;; PUSH R5 ON STACK
MOV @SWR, -(SP) ;; PUSH @SWR ON STACK
MOV SP, $SAVR6 ;; SAVE SP
MOV @SPWRUP, @PWRVEC ;; SET UP VECTOR
HALT
BR -2 ;; HANG UP

```

\*\*\*\*\*

POWER UP ROUTINE

```

$PWRUP: MOV $SILLUP, @PWRVEC ;; SET FOR FAST DOWN
MOV $SAVR6, SP ;; GET SP
CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
IS: INC $SAVR6 ;; WAIT FOR THE INC
BNE IS OF WORD
MOV (SP)+, @SWR ;; POP STACK INTO @SWR
MOV (SP)+, R5 ;; POP STACK INTO R5
MOV (SP)+, R4 ;; POP STACK INTO R4
MOV (SP)+, R3 ;; POP STACK INTO R3
MOV (SP)+, R2 ;; POP STACK INTO R2
MOV (SP)+, R1 ;; POP STACK INTO R1
MOV (SP)+, R0 ;; POP STACK INTO R0
MOV @SPWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
MOV @340, @PWRVEC+2 ;; PRIO:7
TYPE PWRMSG ;; REPORT THE POWER FAILURE
MOV (PC)+, (SP) ;; POWER FAIL MESSAGE POINTER
$PWRAD: .WORD RESTRY ;; RESTART AT RESTRY
RTI ;; RESTART ADDRESS
$SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0 ;; PUT THE SP HERE
PWRMSG: .ASCIZ (<15><12>)/RESTARTED FROM PWR FAIL/

```

.EVEN  
.SBTTL TRAP DECODER

```

*****
*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
*GO TO THAT ROUTINE.

```

```

2753 013326 010046
2754 013330 016600 000002
2755 013334 005740
2756 013336 111000
2757 013340 006300
2758 013342 016000 013350
2759 013346 000200
    
```

```

$TRAP:  MOV  RD, -(SP)      ;; SAVE RD
        MOV  2(SP), RD     ;; GET TRAP ADDRESS
        TST  -(RD)        ;; BACKUP BY 2
        MOVB (RD), RD     ;; GET RIGHT BYTE OF TRAP
        ASL  RD           ;; POSITION FOR INDEXING
        MOV  $TRPAD(RD), RD ;; INDEX TO TABLE
        RTS  RD           ;; GO TO ROUTINE
    
```

.SBTTL TRAP TABLE

;\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
 ;\*BY THE "TRAP" INSTRUCTION.

ROUTINE

```

2760
2761
2762
2763
2764
2765
2766
2767 013350
2768 013350 010320
2769 013352 011074
2770 013354 011050
2771 013356 011110
2772 013360 011276
2773
2774 013362 012416
2775
2776 013364 012346
2777 013366 012630
2778 013370 012750
2779
    
```

```

$TRPAD:
$TYPE   ;; CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
$TYPOC  ;; CALL=TYPOC     TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS  ;; CALL=TYPOS     TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON  ;; CALL=TYPON     TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS  ;; CALL=TYPDS     TRAP+4(104404)  TYPE DECIMAL NUMBER (WITH SIGN)

$GTSWR  ;; CALL=GTSWR     TRAP+5(104405)  GET SOFT-SWR SETTING

$CKSWR  ;; CALL=CKSWR     TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
$RDCHR  ;; CALL=RDCHR     TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
$RDLIN  ;; CALL=RDLIN     TRAP+10(104410) TTY TYPEIN STRING ROUTINE
    
```

.SBTTL ASCII MESSAGES

```

2781 013372 042522 020107 044524
2782 013400 042515 052517 020124
2783 013406 051105 000
2784 013411 122 043505 051040
2785 013416 040505 027504 051127
2786 013424 052111 020105 051105
2787 013432 000
2788 013433 102 051525 051040
2789 013440 051505 052105 042440
2790 013446 000122
2791 013450 047106 052103 041040
2792 013456 052111 020123 040506
2793 013464 046111 042105 052040
2794 013472 020117 042523 020124
2795 013500 052123 052101 041040
2796 013506 052111 000123
2797 013512 042522 042101 020131
2798 013520 047111 051124 043040
2799 013526 044501 052514 042522
2800 013534 000
2801 013535 122 040505 054504
2802 013542 041440 051114 047440
2803 013550 020122 042523 020124
2804 013556 051105 000
2805 013561 123 040524 052524
    
```

```

EM1:  .ASCIZ  /REG TIMEOUT ER/
EM2:  .ASCIZ  'REG READ/WRITE ER'
EM3:  .ASCIZ  /BUS RESET ER/
EM4:  .ASCIZ  /FNCT BITS FAILED TO SET STAT BITS/
EM5:  .ASCIZ  /READY INTR FAILURE/
EM6:  .ASCIZ  /READY CLR OR SET ER/
EM7:  .ASCIZ  /STATUS ER ON XFER/
    
```

2806	013566	020123	051105	047440		
2807	013574	020116	043130	051105		
2808	013602	000				
2809	013603	127	051117	020104	EM10:	.ASCIZ /WORD COUNT ER ON XFER/
2810	013610	047503	047125	020124		
2811	013616	051105	047440	020116		
2812	013624	043130	051105	000		
2813	013631	102	043125	042506	EM11:	.ASCIZ /BUFFER ADRS ER ON XFER/
2814	013636	020122	042101	051522		
2815	013644	042440	020122	047117		
2816	013652	054040	042506	000122		
2817	013660	040504	040524	042440	EM12:	.ASCIZ /DATA ER FROM MEM/
2818	013666	020122	051106	046517		
2819	013674	046440	046505	000		
2820	013701	104	052101	020101	EM13:	.ASCIZ /DATA ER TO MEM/
2821	013706	051105	052040	020117		
2822	013714	042515	000115			
2823	013720	044523	043516	042514	EM14:	.ASCIZ /SINGLE CYCLE OFF DID NOT LOCK OUT CPU/
2824	013726	041440	041531	042514		
2825	013734	047440	043106	042040		
2826	013742	042111	047040	052117		
2827	013750	046040	041517	020113		
2828	013756	052517	020124	050103		
2829	013764	000125				
2830	013766	044523	043516	042514	EM15:	.ASCIZ /SINGLE CYCLE ON LOCKED OUT CPU/
2831	013774	041440	041531	042514		
2832	014002	047440	020116	047514		
2833	014010	045503	042105	047440		
2834	014016	052125	041440	052520		
2835	014024	000				
2836	014025	015	050012	042514	WARN:	.ASCII (15)(12)/PLEASE DISABLE "REV11" MEMORY REFRESH OPTION/
2837	014032	051501	020105	044504		
2838	014040	040523	046102	020105		
2839	014046	051042	053105	030461		
2840	014054	020042	042515	047515		
2841	014062	054522	051040	043105		
2842	014070	042522	044123	047440		
2843	014076	052120	047511	116		
2844	014103	015	040412	042116	.ASCIZ	(15)(12)/AND ENABLE PROCESSOR MEMORY REFRESH /
2845	014110	042440	040516	046102		
2846	014116	020105	051120	041517		
2847	014124	051505	047523	020122		
2848	014132	042515	047515	054522		
2849	014140	051040	043105	042522		
2850	014146	044123	020040	000040		
2851	014154	042516	020130	047514	EM16:	.ASCIZ /NEX LOGIC ER/
2852	014162	044507	020103	051105		
2853	014170	000				
2854	014171	103	041531	042514	EM17:	.ASCIZ /CYCLE FAILED TO CLK DBR (IN)/
2855	014176	043040	044501	042514		
2856	014204	020104	047524	041440		
2857	014212	045514	042040	051102		
2858	014220	024040	047111	000051		
2859	014226	040504	040524	042440	EM20:	.ASCIZ "DATA ER FROM I/O PAGE (XCSR)"

```

2860 014234 020122 051106 046517
2861 014242 044440 047457 050040
2862 014250 043501 020105 054050
2863 014256 051503 024522 000
2864 014263 105 051122 041520
2865 014270 020040 052040 052123
2866 014276 052516 020115 041040
2867 014304 051525 042101 020122
2868 014312 042440 050130 052103
2869 014320 020040 051040 053103
2870 014326 000104
2871 014330 051105 050122 020103
2872 014336 020040 051524 047124
2873 014344 046525 020040 052502
2874 014352 040523 051104 020040
2875 014360 042101 051522 020040
2876 014366 020040 054105 041520
2877 014374 020124 020040 041522
2878 014402 042126 000
2879 014405 105 051122 041520
2880 014412 020040 052040 052123
2881 014420 052516 020115 041040
2882 014426 051525 042101 000122
2883
2884
2885 014434 001116 001534 001122
2886 014442 001124 001126 000000
2887 014450 001116 001534 001122
2888 014456 001120 001124 001126
2889 014464 000000
2890 014466 001116 001534 001122
2891 014474 000000
2892
2893
2894
2895
2896 014476 000000
2897 000001

```

DM1: .ASCIZ ERRPC TSTNUM BUSADR EXPCT RCVD/

DM2: .ASCIZ /ERRPC TSTNUM BUSADR ADRS EXPCT RCVD/

DM3: .ASCIZ /ERRPC TSTNUM BUSADR/

DT1: .EVEN \$ERRPC, TSTNUM, \$BDADR, \$GDDAT, \$BDDAT, 0

DT2: \$ERRPC, TSTNUM, \$BDADR, \$GDADR, \$GDDAT, \$BDDAT, 0

DT3: \$ERRPC, TSTNUM, \$BDADR, 0

```

*****
; '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	748#	846	887				
ACDW1 = 000000	846	889					
ACDW2 = 000000	846	890					
ACPUOP = 000000	846	861					
ADDW0 = 000000	846	891					
ADDW1 = 000000	846	892					
ADDW10 = 000000	846	901					
ADDW11 = 000000	846	902					
ADDW12 = 000000	846	903					
ADDW13 = 000000	846	904					
ADDW14 = 000000	846	905					
ADDW15 = 000000	846	906					
ADDW2 = 000000	846	893					
ADDW3 = 000000	846	894					
ADDW4 = 000000	846	895					
ADDW5 = 000000	846	896					
ADDW6 = 000000	846	897					
ADDW7 = 000000	846	898					
ADDW8 = 000000	846	899					
ADDW9 = 000000	846	900					
ADEVCT = 000000	846	852					
ADEVN = 000001	750#	846	888				
RENV = 000000	846	857					
REVM = 000000	846	858					
RFATAL = 000000	846	849					
RMADR1 = 000000	846	874					
RMADR2 = 000000	846	878					
RMADR3 = 000000	846	881					
RMADR4 = 000000	846	884					
RMAMS1 = 000000	846	868					
RMAMS2 = 000000	846	876					
RMAMS3 = 000000	846	879					
RMAMS4 = 000000	846	882					
RMSGAD = 000000	846	854					
RMSGLG = 000000	846	855					
RMSGTY = 000000	846	848					
AMTYP1 = 000000	846	869					
AMTYP2 = 000000	846	877					
AMTYP3 = 000000	846	880					
AMTYP4 = 000000	846	883					
APASS = 000000	846	851					
APRIOR = 000000	846						
APTC SU = 000040	2136	2241#					
APTENV = 000001	2129	2197	2239#	2416			
APTSIZ = 000200	1079	2238#					
APTSPO = 000100	2131	2199	2240#				
ASWREG = 000000	846	859					
ATESTN = 000000	846	850					
AUNIT = 000000	846	853					
AUSWR = 000000	846	860					
AVECT1 = 000124	749#	846	885				
AVECT2 = 000000	846	886					
BIT0 = 000001	733#						
BIT00 = 000001	723#	733	1196	1201	1260	2000	





FR7 = 000340	677*												
PS = 177776	650*	651											
PSW = 177776	651*												
PWRMSG 013274	2731	2738*											
PWRVEC= 000024	742*	1055*	1056*	2700*	2701*	2710*	2716*	2728*	2729*				
ROCHR = 104407	2670	2777*											
RDLIN = 104410	2778*												
RESTRT 002330	1132	1134	1136*	1914	2733								
RESVEC= 000010	737*												
RSTVEC 007622	1296	1374	1479	1526	1564	1598	1631	1667	1714	1775	1813	1857	1896
	1970*												
R0 =%000000	658*	1083*	1085*	1087	1089*	1092*	1094	1121*	1123	1125*	1126	1139*	1140*
	1141*	1154*	1156	1157	1158	1172*	1180*	1192*	1202*	1215*	1223*	1233*	1234
	1235*	1238*	1239*	1240	1279*	1280	1282	1288*	1316*	1317	1318*	1321*	1322*
	1323	1327*	1336*	1337	1344*	1441*	1445	1466	1473*	1476*	1477*	1488*	1492
	1513	1520*	1523*	1524*	1608*	1626*	1641*	1657*	1721*	1758*	1939*	1942	2080*
	2088*	2127	2128*	2133	2138	2141*	2193	2201*	2205	2206	2208*	2209*	2210
	2232*	2332	2342*	2346	2362	2363	2376*	2454	2455*	2456*	2463*	2464*	2465*
	2466*	2467*	2468	2473	2478*	2480*	2484	2486	2702	2727*	2752	2753*	2754
	2755*	2756*	2757*	2758*									
R1 =%000001	659*	1084*	1085	1086*	1090*	1091*	1092	1093*	1155*	1159*	1442*	1474*	1489*
	1521*	2061*	2062	2065	2066	2068	2082*	2084	2086	2094	2097*	2100*	2101*
	2194	2231*	2333	2346*	2347	2351	2375*	2703	2726*				
R2 =%000002	660*	2081*	2084	2086	2092	2093	2334	2345*	2349*	2352	2359*	2360*	2361
	2366*	2374*	2704	2725*									
R3 =%000003	661*	2015*	2017*	2019	2021*	2023*	2024	2038*	2039	2042*	2043*	2045*	2046*
	2047	2083*	2098*	2275	2284*	2290*	2291*	2294*	2299*	2300*	2301	2310*	2335
	2343*	2344*	2358*	2361*	2370*	2371*	2373*	2666	2667*	2668	2671*	2672	2676
	2678	2680*	2682*	2705	2724*								
R4 =%000004	662*	2016*	2017	2018*	2023	2026*	2027*	2028*	2041*	2042	2044*	2045	2050*
	2051*	2052*	2276	2278*	2279*	2280*	2281	2282*	2296	2298*	2306*	2309*	2706
	2723*												
R5 =%000005	663*	1275*	1439*	1456*	1486*	1502*	1533*	1548*	1571*	1586*	1606*	1639*	1674*
	1692*	1710*	1722*	1781*	1799*	1809*	1819*	1841*	1851*	1862*	1878*	1962	1964*
	1984	1988*	1989*	1994*	1995*	1996	2004*	2005*	2006*	2007*	2039*	2040*	2047
	2080	2090*	2091*	2096*	2277	2283*	2285*	2287*	2288*	2289*	2290	2308*	2336
	2338*	2340*	2347*	2351*	2366	2372*	2707	2722*					
R6 =%000006	664*	666	1043*	1044*	1045								
R7 =%000007	665*	667											
SETUP2 002016	1085*	1088											
SETUP3 002046	1092*	1095											
SETVEC 007602	1275	1439	1486	1533	1571	1606	1639	1674	1722	1781	1819	1862	1961*
SP =%000006	666*	1047*	1064*	1072*	1076	1137*	1162	1291	1358	1361	1365	1369	1455
	1501	1547	1585	1625	1656	1691	1735	1798	1840	1877	1936*	2070*	2127*
	2128	2138*	2140	2141	2142*	2144	2146	2148	2154	2156*	2158*	2166*	2170
	2174	2175	2179	2193*	2194*	2201	2202*	2213	2214*	2215*	2225	2226*	2231
	2232	2267*	2268	2269	2270*	2275*	2276*	2277*	2283	2308	2309	2310	2311*
	2312*	2332*	2333*	2334*	2335*	2336*	2337*	2338	2341*	2354	2356*	2358	2368
	2370	2372	2373	2374	2375	2376	2378*	2379*	2410	2431*	2434*	2454*	2459*
	2480	2484*	2514*	2517	2519	2520	2549	2550	2554*	2571*	2572*	2573	2580*
	2583*	2584*	2588*	2589*	2593	2596*	2600	2602	2604	2605*	2612	2614	2616*
	2617	2619*	2620*	2621*	2622*	2623*	2638*	2639*	2642*	2643*	2644	2648*	2649*
	2650	2653	2655	2657*	2666*	2671	2682	2683*	2684*	2685*	2702*	2703*	2704*
	2705*	2706*	2707*	2708*	2709	2717*	2721	2722	2723	2724	2725	2726	2727













..SAMI	10	6130	627
..SALO	6380		
..SALI	10	6130	766
..SAPB	10	6130	843
..SAPH	10	6130	733
..SAPY	10	6130	2185
..SASTA	10		
..SCATC	10	6130	752
..SCTA	10	6130	799
..SOB20	10		
..SOB20	10		
..SOBY	10		
..SOOP	10	6130	1915
..SOOR	10	6130	2386
..SOPT	10	6130	2445
..SOPL	10		
..SOPM	10	6130	2696
..SOPND	10		
..SOPDR	10		
..SPOOC	10	6130	
..SPOED	10	6130	2557
..SPOZ	10		
..SSAVE	10		
..SSVCU	10		
..SSB20	10		
..SSCOP	10	6130	2492
..SSIZE	10		
..SSUPR	10		
..STRAP	10	6130	2744
..STYPB	10		
..STYPD	10	6130	2319
..STYPE	10	6130	2106
..STYPO	10	6130	2242
..S40CA	10		
..S	10		





.ASCIZ	838	841	1114	1950	2490	2690	2691	2692	2694	2738	2781	2784	2788	2791	2797
	2801	2805	2809	2813	2817	2820	2823	2830	2844	2851	2854	2859	2864	2871	2879
.BLKB	2589														
.BLKW	2385														
.BYTE	809	809	814	815	823	824	832	833	834	835	857	858	868	869	876
	877	879	880	882	883	1949	2234	2235	2236	2314	2315	2316	2317	2422	2423
	2687	2688													
.DSABL	2627														
.ENABL	1	613	2560												
.END	2897														
.ENDC	618	634	636	637	638	642	734	748	763	769	773	775	780	782	789
	802	806	808	836	837	838	839	843	846	868	876	879	892	885	886
	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901
	902	903	904	905	906	907	911	1039	1047	1048	1051	1053	1055	1057	1058
	1060	1062	1083	1100	1106	1112	1114	1119	1121	1144	1145	1146	1147	1148	1149
	1167	1168	1169	1170	1187	1188	1189	1190	1206	1210	1211	1212	1213	1230	1231
	1232	1233	1243	1246	1247	1248	1249	1250	1266	1270	1271	1272	1273	1274	1299
	1300	1301	1302	1303	1310	1313	1314	1315	1316	1330	1331	1332	1333	1348	1349
	1350	1351	1352	1377	1378	1379	1380	1393	1396	1397	1398	1399	1415	1416	1417
	1418	1432	1435	1436	1437	1438	1482	1483	1484	1485	1529	1530	1531	1532	1533
	1567	1568	1569	1570	1571	1601	1602	1603	1604	1605	1634	1635	1636	1637	1638
	1670	1671	1672	1673	1674	1717	1718	1719	1720	1721	1777	1778	1779	1780	1781
	1815	1816	1817	1818	1819	1859	1860	1861	1862	1899	1901	1918	1919	1920	1922
	1925	1931	1934	1935	1939	1941	1947	1949	1950	1953	1957	1961	1967	1970	1977
	1982	2010	2015	2033	2038	2057	2061	2074	2080	2109	2138	2188	2189	2192	2219
	2234	2245	2322	2389	2392	2403	2410	2415	2416	2417	2425	2436	2437	2438	2441
	2448	2463	2492	2495	2498	2503	2509	2511	2522	2525	2526	2527	2529	2531	2538
	2542	2547	2549	2553	2556	2557	2560	2561	2563	2591	2627	2631	2659	2660	2667
	2669	2672	2674	2690	2696	2699	2708	2709	2715	2721	2722	2732	2734	2736	2747
	2753	2756	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2893
	2896														
.EQUIV	642	643	651	666	667	696	697	698	699	700	701	702	703	704	705
	724	725	726	727	728	729	730	731	732	733					
.EVEN	846	1114	2237	2491	2743	2884									
.FF	614	634	635	636	637	638	640	706	734	761	768	771	773	779	781
	798	801	805	807	836	837	838	842	843	845	868	876	879	882	885
	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
	901	902	903	904	905	906	907	911	1039	1042	1047	1049	1051	1053	1055
	1057	1058	1060	1078	1099	1100	1101	1104	1113	1118	1120	1143	1145	1147	1148
	1166	1168	1170	1186	1188	1190	1205	1209	1211	1213	1229	1231	1233	1242	1245
	1247	1249	1250	1265	1269	1271	1273	1274	1298	1300	1302	1303	1309	1312	1314
	1316	1329	1331	1333	1347	1349	1351	1352	1376	1378	1380	1392	1395	1397	1399
	1414	1416	1418	1431	1434	1436	1438	1481	1483	1485	1528	1530	1532	1533	1566
	1568	1570	1571	1600	1602	1604	1605	1633	1635	1637	1638	1669	1671	1673	1674
	1716	1718	1720	1721	1776	1778	1780	1781	1814	1816	1818	1819	1858	1860	1862
	1898	1900	1917	1918	1919	1920	1921	1922	1924	1930	1933	1935	1939	1941	1947
	1949	1950	1956	1960	1966	1969	1976	1981	2009	2014	2032	2037	2056	2060	2073
	2079	2108	2129	2187	2189	2192	2219	2234	2244	2321	2388	2391	2402	2406	2413
	2415	2416	2418	2425	2429	2436	2437	2440	2447	2462	2478	2494	2497	2502	2508
	2509	2521	2523	2524	2525	2527	2528	2529	2538	2540	2548	2550	2555	2556	2557
	2559	2561	2562	2563	2591	2630	2631	2659	2667	2668	2672	2673	2689	2690	2696
	2698	2708	2709	2714	2721	2722	2730	2732	2734	2738	2746	2752	2756	2760	2769
	2770	2771	2772	2773	2774	2776	2777	2778	2779	2892	2895				
.FF	634	636	637	638	640	769	773	775	780	782	789	802	805	808	836

	843	846	1047	1099	1100	1119	1121	1143	1144	1145	1146	1147	1167	1168	1169
	1170	1187	1188	1189	1190	1206	1210	1211	1212	1213	1230	1231	1232	1233	1243
	1246	1247	1248	1249	1266	1270	1271	1272	1273	1299	1300	1301	1302	1310	1313
	1314	1315	1316	1330	1331	1332	1333	1348	1349	1350	1351	1377	1378	1379	1380
	1393	1396	1397	1399	1399	1415	1416	1417	1418	1432	1435	1436	1437	1438	1482
	1483	1484	1485	1529	1530	1531	1532	1567	1568	1569	1570	1601	1602	1603	1604
	1634	1635	1636	1637	1670	1671	1672	1673	1717	1718	1719	1720	1777	1778	1779
	1780	1815	1816	1817	1818	1859	1860	1861	1862	1899	1901	1918	1921	1924	1931
	1934	1949	1957	1961	1967	1970	1977	1982	2010	2015	2033	2038	2057	2061	2074
	2080	2109	2188	2245	2322	2389	2391	2406	2436	2437	2438	2441	2448	2463	2492
	2495	2522	2525	2526	2529	2556	2557	2550	2563	2631	2633	2638	2659	2660	2669
	2673	2690	2699	2715	2730	2747	2753	2893	2896						
.FT	1114	2416	2537	2633	2638										
.IFF	1114	2415	2535	2578	2631	2634									
.IIF	613	618	623	624	631	632	633	634	637	638	758	842	846	1048	1051
	1057	1058	1060	1061	1100	1919	1925	1926	1937	1949	1953	2185	2392	2393	2394
	2395	2396	2401	2428	2436	2437	2460	2485	2498	2499	2500	2501	2502	2503	2507
	2536	2537	2553	2556	2557	2560	2581	2682	2690	2696	2768	2769	2770	2771	2772
	2774	2776	2777	2778											
.IRP	1039	1143	1148	1166	1186	1209	1229	1245	1269	1298	1312	1329	1347	1376	1395
	1414	1434	1481	1528	1566	1600	1633	1669	1716	1776	1814	1858	1924	2193	2194
	2215	2231	2232	2332	2372	2402	2508	2702	2708	2721	2722				
.LIST	1	613	637	748	758	836	843	846	1039	1062	1100	1101	1114	1143	1147
	1166	1170	1186	1190	1209	1213	1229	1233	1245	1249	1269	1273	1298	1302	1312
	1316	1329	1333	1347	1351	1376	1380	1395	1399	1414	1418	1434	1438	1481	1485
	1528	1532	1566	1570	1600	1604	1633	1637	1669	1673	1716	1720	1776	1780	1814
	1818	1858	1862	1925	1941	2436	2502	2659	2760	2768	2769	2770	2771	2772	2773
	2774	2775	2776	2777	2778	2779									
.MACRO	1	638	799	1014	1078	2760									
.MCALL	613	748	843	1062	1101										
.LIST	1	613	637	748	758	836	843	846	1039	1062	1100	1101	1114	1143	1147
	1166	1170	1186	1190	1209	1213	1229	1233	1245	1249	1269	1273	1298	1302	1312
	1316	1329	1333	1347	1351	1376	1380	1395	1399	1414	1418	1434	1438	1481	1485
	1528	1532	1566	1570	1600	1604	1633	1637	1669	1673	1716	1720	1776	1780	1814
	1818	1858	1862	1925	1941	2436	2502	2659	2760	2768	2769	2770	2771	2772	2773
	2774	2775	2776	2777	2778	2779									
.PAGE	766	799	911	961	1014	1039	1143	1953	2104						
.REM	1	307													
.REPT	758														
.SBTTL	627	638	752	761	766	777	799	843	911	1039	1041	1096	1101	1143	1166
	1186	1209	1229	1245	1269	1298	1312	1329	1347	1376	1395	1414	1434	1481	1528
	1566	1600	1633	1669	1716	1776	1814	1858	1915	1953	2104	2106	2185	2242	2319
	2386	2445	2492	2557	2696	2744	2760	2780							
.TITLE	613														
.WORD	758	759	760	764	774	793	794	795	796	797	798	807	810	811	812
	813	816	817	818	819	820	821	822	825	826	827	848	849	850	851
	852	853	854	855	859	860	861	874	878	881	884	895	886	887	888
	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903
	904	905	906	1930	1933	1948	2135	2182	2217	2318	2471	2476	2731	2733	

ERRORS DETECTED: 0

\*.DVDRAA.SEQ.SOL/CRF/NL:TOC=DVDRAA.SML.DVDRAA.P11  
RUN-TIME: 49 58 5 SECONDS  
CORE USED: 32K

