

DRV11-B

DMA INTFC DIAG
CNDRAAO

AH-T450A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



Table with multiple columns and rows of data, likely representing a diagnostic or configuration table. The text is very faint and difficult to read, but appears to be organized in a grid format.

5675
5676
5677
5678
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699
5700
5701
5702
5703
5704
5705
5706
5707

5708
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5720
5721

.REM %

IDENTIFICATION

PRODUCT CODE: AC-T449A-MC
PRODUCT NAME: CNDRAAO DRV11B DMA INTFC DIAG
DATE: DEC 1982
MAINTAINER: DIAGNOSTIC SERVICES/ISS
AUTHOR: T.CHAPSKY

(C) 1982,1983

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.

5723
5724
5725
5726
5727
5728
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778

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	REVISION HISTORY

CVDRAC WAS MADE SPECIFIC TO 11/21 PROCESSOR BY CHANGING PRIORITY 7 TO 6 AND ALSO CHANGING DEFAULT ADDRESS AND VECTOR AND ALSO ADDING A CALL TO CNMAC2.SML TO INIT ODT AND LTC VECTORS. THE DIAGNOSTIC WAS RENAMMED TO CNDRAAO.

MAINDEC-11-CNDRA-A DRV11B DMA INTERFACE DIAGNOSTIC
CNDRAA.F11 14-DEC-82 11.43

MACY11 30(1046)^{D 1} 14-DEC-82 11:44 PAGE 58-1

SEQ 0003

5779

11.0 LISTING

5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5830
5831
5832
5833
5834
5835
5836

1.0 ABSTRACT

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

2.0 REQUIREMENTS

2.1 EQUIPMENT

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

2.2 STORAGE

THE PROGRAM USES THE LOWER 4K OF MEMORY.

3.0 LOADING PROCEDURE

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

4.0 STARTING PROCEDURE

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

5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5850
5851
5852
5853
5854

OPTIONS.

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

IF RUNNING UNDER XXDP+ MONITOR:

- A) DO 1., 2., 3. OF THE ABOVE.
B) IN MONITOR MODE TYPE IN 'R NDRAA?'.
C) DO 5. AND 6.

IF USING LSI-11/23B(KDF11-B) AND WANT TO TEST DMA TRANSFERS TO
I/O PAGE PUT THE ADDRESS OF YOUR I/O INTERFACE CONTROL RE-
GISTER INTO LOCATION 1544 (IOPAGE). TO DO THIS PERMANENTLY USE
LOAD-MOD-DUMP PROCEDURE DEFINED IN 7.2. FOR TEMPORARY CHANGES
JUST MODIFY LOCATIONS AFTER LOADING THE PROGRAM.

NOTE: THIS TEST IS NOT GOING TO BE PERFORMED UNLESS THE ABOVE
MENTIONED LOCATIONS ARE MODIFIED.

5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR <5-0>
***SW07=1	000200	DEVICE IS KDF11-B

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

5.2 CONTROL

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

6.0 ERROR REPORTING

6.1 ERROR COMMENT

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

6.2 ERROR DATA

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

5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967

- 7.0 *ALWAYS REPORTED
M C ANEIOUS
- - - - -
- 7.1 DRV11B BUS & VECTOR ADDRESS MODIFICATION
MODIFY LOCATION '\$BASE' IF BASE BUS ADDRESS IS NOT 174600.
MODIFY LOCATION '\$VECT1' IF VECTOR ADDRESS IS NOT 210.

*NOTE: USE THE LSI-11 ODT FACILITIES TO MODIFY THESE LOCATIONS
AFTER PROGRAM LOAD. NO VECTOR ASSIGNMENT ABOVE 774 SHOULD BE
ALLOWED.
- 7.2 XXDP/APT NOTES
THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP (REF. 7.5)(REQUIRES 8K OR MORE).
THIS DIAGNOSTIC DOES SUPPORT "APT" AND HAS RUN UNDER IT.

IF THE PROCESSOR USED IS KDF11-B:
FOR APT SET \$SWREG BIT07 TO 1 (000200)
FOR XXDP CHAINS SET LOCATION 176 (SWR) BIT07 TO 1
TO DO THIS UNDER XXDP:
1. R UPD2
2. LOAD NDRAA?.BIC
3. MOD 176
THE TERMINAL WILL RESPOND: 176/000000
4. NOW TYPE IN 200
5. BE CAREFUL: AT THIS POINT IT IS NECESSARY TO DELETE THE FILE
FROM THE DISK.
DEL DLO:NDRAA?.BIC (IF MEDIA IS RL ON DRIVE C)
6. DUMP NDRAA0.BIC
TO BE SAFE, SKIP 5. AND DUMP THE FILE UNDER DIFFERENT NAME
WITH .BIC EXTENSION.
NOTE: THIS PROCEDURE ASSUMES THAT DIAGNOSTIC IS ON THE DISK
FROM WHICH THE SYSTEM IS BOOTED. IF THIS IS UNTRUE
IN 2. AND 6. THE OPERATOR HAVE TO SPECIFY THE DRIVE BEFORE
THE NAME.
- 7.3 POWER FAIL
A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT
WHICH TIME THE PROGRAM IS RESTARTED (ONLY ON SYSTEMS WITH
NON-VOLATILE MEMORY AND WITH APPROPRIATE HARDWARE).
- 7.4 MULTIPLE DRV11B INTERFACE TESTING
THIS PROGRAM DOES NOT "AUTO-SIZE" THE NUMBER OF DRV11B'S CONNECTED.
THIS DIAGNOSTIC WILL TEST SEQUENTIALLY UP TO 8 DRV11B INTERFACES
WITH CONTIGUOUS BUS AND VECTOR ADDRESSES. THIS IS ACCOMPLISHED
BY THE OPERATOR SETTING UP LOCATION '\$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 'REV1?' (DMA REFRESH), THE DMA

5968
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6020
6021
6022
6023

- REFRESH MUST BE DISABLED AND CPU REFRESH MUST BE ENABLED.
- 8.0 EXECUTION TIME

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

- 9.1 GENERAL
- THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS AND DATA PATHS OF THE DRV11B DMA INTERFACE. A HIGH DEGREE OF TESTING IS ACCOMPLISHED WITH THE AID OF THE MAINTENANCE LOOP BACK CABLE PROVIDED FOR DIAGNOSTIC TESTING. A COMPLETE LIST OF TESTS IS AVAILABLE IN THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN EACH TEST CAN BE BENEFICIAL IN TEST UNDERSTANDING.
- 9.2 REGISTER TESTS
- THE FOLLOWING REGISTERS ARE READ/WRITE & RESET TESTED:
1. WORD COUNT
 2. BUFFER ADDRESS
 3. COMMAND/STATUS
 4. DATA BUFFER
- 9.3 BYTE ADDRESSING TESTS
1. COMMAND/STATUS
 2. DATA BUFFER
- 9.4 'FNCT' TO 'STAT' WRAP AROUND TEST
- 9.5 READY INTERRUPT TEST
- 9.6 NPR DATA TRANSFER TESTS
- THE FOLLOWING NPR XFERS ARE CHECKED FOR CORRECT STATUS, WORD COUNT, BUFFER ADDRESS & DATA:
1. SINGLE 'DATI' XFER - FLOATING 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 FOR KDF11-B PROCESSOR THE USER HAVE TO SELECT CSR OF A PARTICULAR INTERFACE (DISK INTERFACES ARE SUGGESTED). SOME OF THEM ARE:
RXV11 (RX01): 177170
RXV21 (RX02): 177170
RKV11-D (RK05): 177404

6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042

- 9.7 MAINT MODE NPR DATA TRANSFER TESTS
 - 1. THAT MAINT MODE CONTROLS 'FNCT' BITS
 - 2. 200 MAINT MODE XFERS - CHECKING STATUS & DATA
 - 3. 200 MAINT MODE XFERS TO EACH 4K AVAILABLE MEM
- 9.8 BURST & NON-BURST MODE TESTS
 - 1. THAT CPU IS LOCKED OUT IN BURST MODE
 - 2. THAT CPU IS NOT LOCKED OUT IN NON-BURST MODE
- 9.9 'NEX' ERROR CONDITION TEST
- 10.0 REVISION HISTORY
CVDRAC DIAGNOSTIC WAS MADE SPECIFIC TO 11/21 PROCESSOR AND
IS CALLED CNDRAAO.
- 11.0 LISTING

- %

```

6054 .TITLE MAINDEC-11-CNDRA-A DRV11B DMA INTERFACE DIAGNOSTIC
(1)  ;*COPYRIGHT (C) 1982
(1)  ;*DIGITAL EQUIPMENT CORP.
(1)  ;*MAYNARD, MASS. 01754
(1)  ;*
(1)  ;*PROGRAM BY R. MOORE
(1)  ;*
(1)  ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)  ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)  ;*
(1)  000001 $TN=1
(1)  160000 $$SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
6055 167400 $$SWR=167400
6056 000300 $$SWRMK=300
6057 000001 $TN=1
6058 .SBTTL OPERATIONAL SWITCH SETTINGS
(1)  ;*
(1)  ;* SWITCH USE
(1)  ;* -----
(1)  ;* 15 HALT ON ERROR
(1)  ;* 14 LOOP ON TEST
(1)  ;* 13 INHIBIT ERROR TYPEOUTS
(1)  ;* 11 INHIBIT ITERATIONS
(1)  ;* 10 BELL ON ERROR
(1)  ;* 9 LOOP ON ERROR
(1)  ;* 8 LOOP ON TEST IN SWR<5:0>
6059 .SBTTL BASIC DEFINITIONS
(1)  ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)  001100 STACK= 1100
(1)  .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
(1)  .EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL
(1)  ;*MISCELLANEOUS DEFINITIONS
(1)  000011 HT= 11 ;:CODE FOR HORIZONTAL TAB
(1)  000012 LF= 12 ;:CODE FOR LINE FEED
(1)  000015 CR= 15 ;:CODE FOR CARRIAGE RETURN
(1)  000200 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
(1)  177776 PS= 177776 ;:PROCESSOR STATUS WORD
(1)  .EQUIV PS,PSW
(1)  177774 STKLMT= 177774 ;:STACK LIMIT REGISTER
(1)  177772 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
(1)  177570 DSWR= 177570 ;:HARDWARE SWITCH REGISTER
(1)  177570 DDISP= 177570 ;:HARDWARE DISPLAY REGISTER
(1)  ;***** THE FOLLOWING ODT START ADDRESS FOR SBC 11/21 IS ADDED
(1)  170000 ODTST= 170000
(1)  ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)  000000 R0= %0 ;:GENERAL REGISTER
(1)  000001 R1= %1 ;:GENERAL REGISTER
(1)  000002 R2= %2 ;:GENERAL REGISTER
(1)  000003 R3= %3 ;:GENERAL REGISTER
(1)  000004 R4= %4 ;:GENERAL REGISTER
(1)  000005 R5= %5 ;:GENERAL REGISTER
(1)  000006 R6= %6 ;:GENERAL REGISTER
(1)  000007 R7= %7 ;:GENERAL REGISTER
(1)  000006 SP= %6 ;:STACK POINTER
  
```

```

(1)      000007      PC=      %7      ;;PROGRAM COUNTER
(1)
(1)      ;*PRIORITY LEVEL DEFINITIONS
(1)      000000      PR0=      0      ;;PRIORITY LEVEL 0
(1)      000040      PR1=      40     ;;PRIORITY LEVEL 1
(1)      000100      PR2=      100    ;;PRIORITY LEVEL 2
(1)      000140      PR3=      140    ;;PRIORITY LEVEL 3
(1)      000200      PR4=      200    ;;PRIORITY LEVEL 4
(1)      000240      PR5=      240    ;;PRIORITY LEVEL 5
(1)      000300      PR6=      300    ;;PRIORITY LEVEL 6
(1)      000340      PR7=      340    ;;PRIORITY LEVEL 7
(1)
(1)      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
(1)      100000      SW15=     100000
(1)      040000      SW14=     40000
(1)      020000      SW13=     20000
(1)      010000      SW12=     10000
(1)      004000      SW11=     4000
(1)      002000      SW10=     2000
(1)      001000      SW09=     1000
(1)      000400      SW08=     400
(1)      000200      SW07=     200
(1)      000100      SW06=     100
(1)      000040      SW05=     40
(1)      000020      SW04=     20
(1)      000010      SW03=     10
(1)      000004      SW02=     4
(1)      000002      SW01=     2
(1)      000001      SW00=     1
(1)      .EQUIV      SW09,SW9
(1)      .EQUIV      SW08,SW8
(1)      .EQUIV      SW07,SW7
(1)      .EQUIV      SW06,SW6
(1)      .EQUIV      SW05,SW5
(1)      .EQUIV      SW04,SW4
(1)      .EQUIV      SW03,SW3
(1)      .EQUIV      SW02,SW2
(1)      .EQUIV      SW01,SW1
(1)      .EQUIV      SW00,SW0
(1)
(1)      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)      100000      BIT15=    100000
(1)      040000      BIT14=    40000
(1)      020000      BIT13=    20000
(1)      010000      BIT12=    10000
(1)      004000      BIT11=    4000
(1)      002000      BIT10=    2000
(1)      001000      BIT09=    1000
(1)      000400      BIT08=    400
(1)      000200      BIT07=    200
(1)      000100      BIT06=    100
(1)      000040      BIT05=    40
(1)      000020      BIT04=    20
(1)      000010      BIT03=    10
(1)      000004      BIT02=    4
(1)      000002      BIT01=    2
  
```


BASIC DEFINITIONS

```

(1)          000001          BIT00= 1
(1)          .EQUIV BIT09,BIT9
(1)          .EQUIV BIT08,BIT8
(1)          .EQUIV BIT07,BIT7
(1)          .EQUIV BIT06,BIT6
(1)          .EQUIV BIT05,BIT5
(1)          .EQUIV BIT04,BIT4
(1)          .EQUIV BIT03,BIT3
(1)          .EQUIV BIT02,BIT2
(1)          .EQUIV BIT01,BIT1
(1)          .EQUIV BIT00,BIT0

(1)          000004          ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)          000010          ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
(1)          000014          RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
(1)          000014          TBITVEC=14        ;; "T" BIT
(1)          000014          TRTVEC= 14         ;; TRACE TRAP
(1)          000014          BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
(1)          000020          IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)          000024          PWRVEC= 24         ;; POWER FAIL
(1)          000030          EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
(1)          000034          TRAPVEC=34        ;; "TRAP" TRAP
(1)          000060          TKVEC= 60          ;; TTY KEYBOARD VECTOR
(1)          000064          TPVEC= 64          ;; TTY PRINTER VECTOR
(1)          ;***** THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED
(1)          000100          LKVEC= 100         ;; LINE CLOCK VECTOR
(1)          000140          BRKVEC= 140        ;; BREAK VECTOR
(1)          000240          PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR
6060          174600          ABASE= 174600    ;; BASE DRV11B BUS ADRS EQUATE
6061          000210          AVECT1= 000210   ;; BASE DRV11B VECTOR ADRS EQUATE -
6062          000001          ADEVM= 1         ;; DEFAULT TO ONE DRV11B
6063          106427          MTPS=106427     ;; INSTR EQUATE THAT MOVES BYTE TO PSW
6064          000000          TMAIN: 177522    ;; MAINTENANCE REGISTER(FOR USE WITH KDF11-B)
6065          .SBTTL TRAP CATCHER

(1)          000000          .=0
(1)          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)          000174          .=174
(1)          000174          000000          DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
(1)          000176          000000          SWREG: .WORD 0          ;; SOFTWARE SWITCH REGISTER
(1)          .SBTTL STARTING ADDRESS(ES)
(1)          000200          000137          001550          JMP @#START ;; JUMP TO STARTING ADDRESS OF PROGRAM
6066          000100          000104          000200          000002          .=100
6067          .WORD 104,200,2          ;IF 'B EVENT' ON Q BUS IS CONNECTED
6068          ;IGNORE IT'S INTERRUPT - JUST DO A RTI

```

```

6070      .SBTTL  ACT11 HOOKS
(1)
(2)      ::*****
(1)      :HOOKS REQUIRED BY ACT11
(1)      000106      $SVPC-.          ;SAVE PC
(1)      000046      .=46
(1) 000046 010056      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)      000052      .=52
(1) 000052 000000      .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)      000106      .=$SVPC          ;; RESTORE PC
(1)      001000      .=1000
6071
6072      .SBTTL  APT PARAMETER BLOCK
(1)
(2)      ::*****
(1)      :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      :*****
(1)      001000      .$X=.          ;;SAVE CURRENT LOCATION
(1)      000024      .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200      200          ;;FOR APT START UP
(1)      000044      .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000      $APTHDR       ;;POINT TO APT HEADER BLOCK
(1)      001000      .=.$X          ;;RESET LOCATION COUNTER
(2)      :*****
(1)      :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)      :INTERFACE SPEC.
(1)
(1)
(1) 001000      $APTHD:
(1) 001000 000000      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001174      $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000031      $TSTM: .WORD 25.        ;;RUN TIM OF LONGEST TEST
(1) 001006 000006      $PASTM: .WORD 6.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000144      $UNITM: .WORD 100.       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000052      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
    
```

6073

.SBTTL COMMON TAGS

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

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

SCMTAG: =1100
\$STNM: .WORD 0
\$ERFLG: .BYTE 0
\$ICNT: .WORD 0
\$LPADR: .WORD 0
\$LPERR: .WORD 0
\$ERTTL: .WORD 0
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 0
\$GDADR: .WORD 0
\$BDADR: .WORD 0
\$GDDAT: .WORD 0
\$BDDAT: .WORD 0
\$AUTOB: .BYTE 0
\$INTAG: .BYTE 0
\$SWR: .WORD DSWR
\$DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$STPFLG: .BYTE 0
\$TIMES: 0
\$ESCAPE: 0
\$BELL: .ASCIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

:::START OF COMMON TAGS
:::CONTAINS THE TEST NUMBER
:::CONTAINS ERROR FLAG
:::CONTAINS SUBTEST ITERATION COUNT
:::CONTAINS SCOPE LOOP ADDRESS
:::CONTAINS SCOPE RETURN FOR ERRORS
:::CONTAINS TOTAL ERRORS DETECTED
:::CONTAINS ITEM CONTROL BYTE
:::CONTAINS MAX. ERRORS PER TEST
:::CONTAINS PC OF LAST ERROR INSTRUCTION
:::CONTAINS ADDRESS OF 'GOOD' DATA
:::CONTAINS ADDRESS OF 'BAD' DATA
:::CONTAINS 'GOOD' DATA
:::CONTAINS 'BAD' DATA
:::RESERVED--NOT TO BE USED
:::AUTOMATIC MODE INDICATOR
:::INTERRUPT MODE INDICATOR
:::ADDRESS OF SWITCH REGISTER
:::ADDRESS OF DISPLAY REGISTER
:::TTY KBD STATUS
:::TTY KBD BUFFER
:::TTY PRINTER STATUS REG. ADDRESS
:::TTY PRINTER BUFFER REG. ADDRESS
:::CONTAINS NULL CHARACTER FOR FILLS
:::CONTAINS # OF FILLER CHARACTERS REQUIRED
:::INSERT FILL CHARS. AFTER A 'LINE FEED'
:::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
:::MAX. NUMBER OF ITERATIONS
:::ESCAPE ON ERROR ADDRESS
:::CODE FOR BELL
:::QUESTION MARK
:::CARRIAGE RETURN
:::LINE FEED

.SBTTL APT MAILBOX-ETABLE

(2) 001174
(2) 001174 000000
(2) 001176 000000
(2) 001200 000000
(2) 001202 000000
(2) 001204 000000
(2) 001206 000000

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

Address	Value	Field Name	Format	Description
(2) 001210	000000	\$MSGAD	.WORD	AMSGAD ::MESSAGE ADDRESS
(2) 001212	000000	\$MSGLG	.WORD	AMSLG ::MESSAGE LENGTH
(2) 001214		\$ETABLE		ENV ::APT ENVIRONMENT TABLE
(2) 001214	000	\$ENV	.BYTE	AENV ::ENVIRONMENT BYTE
(2) 001215	000	\$ENVM	.BYTE	AENVM
(2) 001216	000000	\$SWREG	.WORD	ASWREG ::APT SWITCH REGISTER
(2) 001220	000000	\$USWR	.WORD	AJSWR ::USER SWITCHES
(2) 001222	000000	\$CPUOP	.WORD	ACPUOP ::CPU TYPE,OPTIONS
(2)				BITS 15-11=CPU TYPE
(2)				11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)				11/70=06,PDQ=07,Q=10
(2)				BIT 10=REAL TIME CLOCK
(2)				BIT 9=FLOATING POINT PROCESSOR
(2)				BIT 8=MEMORY MANAGEMENT
(2) 001224	000	\$MAMS1	.BYTE	AMAMS1 ::HIGH ADDRESS,M.S. BYTE
(2) 001225	000	\$MTYP1	.BYTE	AMTYP1 ::MEM. TYPE,BLK#1
(2)				MEM.TYPE BYTE -- (HIGH BYTE)
(2)				900 NSEC CORE=001
(2)				300 NSEC BIPOLAR=002
(2)				500 NSEC MOS=003
(2) 001226	000000	\$MADR1	.WORD	AMADR1 ::HIGH ADDRESS,BLK#1
(2)				MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2) 001230	000	\$MAMS2	.BYTE	AMAMS2 ::HIGH ADDRESS,M.S. BYTE
(2) 001231	000	\$MTYP2	.BYTE	AMTYP2 ::MEM. TYPE,BLK#2
(2) 001232	000000	\$MADR2	.WORD	AMADR2 ::MEM.LAST ADDRESS,BLK#2
(2) 001234	000	\$MAMS3	.BYTE	AMAMS3 ::HIGH ADDRESS,M.S.BYTE
(2) 001235	000	\$MTYP3	.BYTE	AMTYP3 ::MEM. TYPE,BLK#3
(2) 001236	000000	\$MADR3	.WORD	AMADR3 ::MEM.LAST ADDRESS,BLK#3
(2) 001240	000	\$MAMS4	.BYTE	AMAMS4 ::HIGH ADDRESS,M.S.BYTE
(2) 001241	000	\$MTYP4	.BYTE	AMTYP4 ::MEM. TYPE,BLK#4
(2) 001242	000000	\$MADR4	.WORD	AMADR4 ::MEM.LAST ADDRESS,BLK#4
(2) 001244	000210	\$VECT1	.WORD	AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001246	000000	\$VECT2	.WORD	AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001250	174600	\$BASE	.WORD	ABASE ::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001252	000001	\$DEVN	.WORD	ADEVN ::DEVICE MAP
(2) 001254	000000	\$CDW1	.WORD	ACDW1 ::CONTROLLER DESCRIPTION WORD#1
(2) 001256	000000	\$CDW2	.WORD	ACDW2 ::CONTROLLER DESCRIPTION WORD#2
(2) 001260	000000	\$DDW0	.WORD	ADDW0 ::DEVICE DESCRIPTOR WORD#0
(2) 001262	000000	\$DDW1	.WORD	ADDW1 ::DEVICE DESCRIPTOR WORD#1
(2) 001264	000000	\$DDW2	.WORD	ADDW2 ::DEVICE DESCRIPTOR WORD#2
(2) 001266	000000	\$DDW3	.WORD	ADDW3 ::DEVICE DESCRIPTOR WORD#3
(2) 001270	000000	\$DDW4	.WORD	ADDW4 ::DEVICE DESCRIPTOR WORD#4
(2) 001272	000000	\$DDW5	.WORD	ADDW5 ::DEVICE DESCRIPTOR WORD#5
(2) 001274	000000	\$DDW6	.WORD	ADDW6 ::DEVICE DESCRIPTOR WORD#6
(2) 001276	000000	\$DDW7	.WORD	ADDW7 ::DEVICE DESCRIPTOR WORD#7
(2) 001300	000000	\$DDW8	.WORD	ADDW8 ::DEVICE DESCRIPTOR WORD#8
(2) 001302	000000	\$DDW9	.WORD	ADDW9 ::DEVICE DESCRIPTOR WORD#9
(2) 001304	000000	\$DDW10	.WORD	ADDW10 ::DEVICE DESCRIPTOR WORD#10
(2) 001306	000000	\$DDW11	.WORD	ADDW11 ::DEVICE DESCRIPTOR WORD#11
(2) 001310	000000	\$DDW12	.WORD	ADDW12 ::DEVICE DESCRIPTOR WORD#12
(2) 001312	000000	\$DDW13	.WORD	ADDW13 ::DEVICE DESCRIPTOR WORD#13
(2) 001314	000000	\$DDW14	.WORD	ADDW14 ::DEVICE DESCRIPTOR WORD#14
(2) 001316	000000	\$DDW15	.WORD	ADDW15 ::DEVICE DESCRIPTOR WORD#15

(2)
(2) 001320
(2)

SETEND:

6167			:ERROR	20					
6168	001510	014722		EM20			:DATA ER FROM I/O PAGE (XCSR)		
6169	001512	015024		DH2			:ERRPC TSTNUM BUSADR ADRS EXPCT RCVD		
6170	001514	015144		DT2			:SERRPC TSTNUM \$BDADR \$GDADR \$GDDAT \$BDDAT		
6171	001516	000000		0					
6172									
6173									
6174			:DRV11B BUS REGISTER ADDRESS POINTERS						
6175									
6176	001520	174600	DRVWCR:	ABASE			:WORD COUNT		
6177	001522	174602	DRVBAR:	ABASE+2			:BUFFER ADDRESS		
6178	001524	174604	DRVCSR:	ABASE+4			:COMMAND/STATUS		
6179	001526	174606	DRVDBR:	ABASE+6			:DATA BUFFER		
6180									
6181			:DRV11B VECTOR ADDRESS POINTERS						
6182									
6183	001530	000210	DRVCT0:	AVECT1			:READY, NEX & INCOMPLETE DATIO VECTOR		
6184	001532	000212	DRVCT2:	AVECT1+2			:NEW PSW ON INTR		
6185									
6186			:COMMON PROGRAM LOCATION(S)						
6187									
6188	001534	000000	TSTNUM:	0			:CONTAINS TEST NUMBER ON ERROR		
6189	001536	000001	DMAP:	1			:DEVICE MAP - EA BIT SAYS TEST THAT DRV11B		
6190	001540	000000	CORSZ:	0			:CONTAINS 1ST NON-EXISTANT MEM ADRS		
6191	001542	015316	DBUFP:	DBUF			:CONTAINS CURRENT 4K NPR BUFFER ADRS		
6192	001544	000000	IOPAGE:	0			:CONTAINS ADDRESS FOR I/O TRANSFERS		
6193	001546	000000	KDF:	0			:IF KDF11-B		


```

6196          .SBTTL PROGRAM START
6197 001550 START:
(1)          .SBTTL INITIALIZE THE COMMON TAGS
(1)          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001550 012706 001100 MOV    # $CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 001554 005026 CLR    (R6)+           ;;CLEAR MEMORY LOCATION
(1) 001556 022706 001140 CMP    #SWR,R6      ;;DONE?
(1) 001562 001374 BNE    -6             ;;LOOP BACK IF NO
(1) 001564 012706 001100 MOV    #STACK,SP     ;;SETUP THE STACK POINTER
(1)          ;;INITIALIZE A FEW VECTORS
(1) 001570 012737 012374 000020 MOV    # $SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 001576 012737 000300 000022 MOV    #PR6,@#IOTVEC+2 ;;LEVEL 6
(1) 001604 012737 012032 000030 MOV    # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 001612 012737 000300 000032 MOV    #PR6,@#EMTVEC+2 ;;LEVEL 6
(1)          ;;BIT02
(1) 001620 012737 014004 000034 MOV    #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 001626 012737 000300 000036 MOV    #PR6,@#TRAPVEC+2;LEVEL 6
(1) 001634 012737 013600 000024 MOV    # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 001642 012737 000300 000026 MOV    #PR6,@#PWRVEC+2 ;;LEVEL 6
(1) 001650 005037 001160 CLR    TIMES         ;;INITIALIZE NUMBER OF ITERATIONS
(1) 001654 005037 001162 CLR    $ESCAPE       ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 001660 112737 000001 001115 MOV    #1,$ERMAX    ;;ALLOW ONE ERROR PER TEST
(1) 001666 012737 001666 001106 MOV    #,$SLPADR    ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 001674 012737 001674 001110 MOV    #,$SLPERR    ;;SETUP THE ERROR LOOP ADDRESS
(2)          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001702 013746 000004 MOV    @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 001706 012737 001742 000004 MOV    #64$,@#ERRVEC ;;SET UP ERROR VECTOR
(2) 001714 012737 177570 001140 MOV    #DSWR,SWR    ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001722 012737 177570 001142 MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001730 022777 177777 177202 CMP    #-1,@SWR    ;;TRY TO REFERENCE HARDWARE SWR
(2) 001736 001012 BNE    66$         ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)          ;;AND THE HARDWARE SWR IS NOT = -1
(2) 001740 000403 BR    65$         ;;BRANCH IF NO TIMEOUT
(2) 001742 012716 001750 64$: MOV    #65$,(SP) ;;SET UP FOR TRAP RETURN
(2) 001746 000002 RTI
(2) 001750 012737 000176 001140 65$: MOV    #SWREG,SWP    ;;POINT TO SOFTWARE SWR
(2) 001756 012737 000174 001142 MOV    #DISPREG,DISPLAY
(2) 001764 012637 000004 66$: MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)          CLR    $PASS    ;;CLEAR PASS COUNT
(2) 001770 005037 001202 BITB   #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 001774 132737 000200 001215 BEQ    67$         ;;YES,USE NON-APT SWITCH
(2) 002002 001403 MOV    #SWREG,SWR    ;;NO,USE APT SWITCH REGISTER
(2) 002004 012737 001216 001140 67$:
(2) 002012 START1: MOV    #DRVWCR,R0 ;;SET UP REG ADRS PCINTERS
6198 002012 012700 001520 MOV    $BASE,R1    ;;GET BASE ADRS
6199 002016 013701 001250 SETUP2: MOV    R1,(R0)+ ;;LOAD EM
6200 002022 010120 ADD    #2,R1
6201 002024 062701 000002 CMP    #DRVDBR+2,PC ;;ALL DONE?
6202 002030 022700 001530 BNE    SETUP2     ;;BR IF NOT
6203 002034 001372 MOV    #DRVCT0,RC   ;;SET UP DRV11B VECTOR ADRS POINTER
6204 002036 012700 001530 MOV    $VECT1,R1   ;;GET BASE VECTOR ADRS
6205 002042 013701 001244 BIC    #170000,R1 ;;CLR OUT PRIORITY BITS
6206 002046 042701 170000 SETUP3: MOV    R1,(R0)+
6207 002052 010120 ADD    #2,R1
6208 002054 062701 000002
    
```

```

6209 002060 022700 001534      CMP      #DRVCT2+2,RO      ;ALL DONE?
6210 002064 001372      BNE      SETUP3          ;BR IF NOT
6211      .SBTTL TYPE PROGRAM NAME
(1)      ::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002066 005227 177777      INC      #-1             ;:FIRST TIME?
(1) 002072 001052      BNE      64$             ;:BRANCH IF NO
(1) 002074 104401 002142      TYPE     ,65$           ;:TYPE ASCIZ STRING
(2)      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002100 005737 000042      TST     @#42            ;:ARE WE RUNNING UNDER XXDP/ACT?
(2) 002104 001012      BNE      66$             ;:BRANCH IF YES
(2) 002106 123727 001214 000001  CMPB     $ENV,#1         ;:ARE WE RUNNING UNDER APT?
(2) 002114 001406      BEQ     66$             ;:BRANCH IF YES
(2) 002116 023727 001140 000176  CMP     SWR,#SWREG       ;:SOFTWARE SWITCH REG SELECTED?
(2) 002124 001005      BNE      67$             ;:BRANCH IF NO
(2) 002126 104406      GTSWR                    ;:GET SOFT-SWR SETTINGS
(2) 002130 000403      BR      67$
(2) 002132 112737 000001 001134 66$:   MOVB     #1,$AUTOB       ;:SET AUTO-MODE INDICATOR
(2) 002140      67$:
(1) 002140 000427      BR      64$             ;:GET OVER THE ASCIZ
(1)      ::65$: .ASCIZ <CRLF>#MD-11-CNDRA-A DRV11B DMA INTERFAE DIAG #<CRLF>
(1)      64$:
6212 002220 005227 177777      INC      #-1             ;:FIRST PASS?
6213 002224 001022      BNE      100$            ;:BRANCH IF NO
6214 002226 122737 000001 001134  CMPB     #1,$AUTOB       ;:MANUAL INTERVENTION PERMITTED?
6215 002234 001416      BEQ     100$            ;:BR IF NO
6216 002236 104401 015172      TYPE     ,KDF11B        ;:ASK ABOUT KDF11-B
6217 002242 104410      RDCHR                    ;:READ ANSWER
6218 002244 012637 001546      MOV     (SP)+,KDF        ;:STORE ANSWER
6219 002250 104401 001546      TYPE     ,KDF           ;:ECHO ANSWER
6220 002254 123727 001546 000131  CMPB     KDF,#131        ;:IS IT KDF11-B?
6221 002262 001003      BNE      100$            ;:BRANCH IF NOT
6222 002264 152777 000200 176646  BISB     #BIT07,@SWR     ;:IF YES,SET SWREG
6223 002272 005737 001540 100$:   TST     CORSZ           ;:TEST IF FIRST PASS
6224 002276 001002      BNE      CORSZR         ;:BR IF NOT
6225 002300 104401 014521      TYPE     ,WARN          ;:TELL THE OPERATOR TO TURN OFF DMA REFRESH
6226      ::*****
6227      :LET'S SEE HOW MUCH MEM WE HAVE
6228      :*****
6229 002304 012700 020000 000004  CORSZR:  MOV     #20000,RO     ;:USE RO TO LOOK
6230 002310 012737 002322      MOV     #2$,@#ERRVEC     ;:SET UP TIME OUT RETURN ADRS
6231 002316 005720 1$:      TST     (RO)+           ;:TAKE A LOOK
6232 002320 000776      BR      1$              ;:UNTIL TIMEOUT
6233 002322 042700 017777 2$:      BIC     #17777,RO        ;:POINT TO 1ST NON-EXSISTANT 4K BLK
6234 002326 010037 001540      MOV     RO,CORSZ         ;:SAVE FOR LATER
6235 002332 012737 000006 000004  MOV     #ERRVEC+2,@#ERRVEC ;:RESTORE VECTOR
6236 002340 012737 015316 001542  MOV     #DBUF,DBUFP       ;:INITIALIZE TO LOWEST 4K
6237 002346 012737 000000 001206  MOV     #0,$UNIT          ;:SET UP UNIT COUNT
6238 002354 013737 001252 001536  MOV     $DEV#,$DMAP       ;:GET THE # & POSITION OF DRV11B'S
6239 002362 042737 177400 001536  BIC     #177400,$DMAP     ;:UP TO 8 ONLY
6240 002370 001406      BEQ     RESTRT           ;:GO CONTINUE AS IF SOMETHING WAS SELECTED
6241 002372 032737 000001 001536  BIT     #1,$DMAP         ;:IS 1ST DRV11B SELECTED?
6242 002400 001002      BNE      RESTRT         ;:BR IF SO
6243 002402 000137 007674      JMP     NXDEV1           ;:NO - GO ADVANCE BASE DRV11B ADDRESSES
6244 002406 106427 000200  RESTRT:  MTPS     #200           ;:SET PRIORITY TO HIGHEST LEVEL
6245 002412 012706 001100      MOV     #STACK,SP        ;:ALWAYS RESET STACK PTR
6246 002416 013737 001206 001204  MOV     $UNIT,$DEVCT     ;:LOAD APT COUNTER
  
```

```

6247 002424 013700 001206          MOV    $UNIT,RO          ;MAKE AN INDEX
6248 002430 006300                ASL    RO                ; VALUE
6249 002432 013760 001520 001260    MOV    DRVWCR,$DDWO(RO)  ;SAVE THE BUS ADDRESS
6250 002440 000005                RESET                   ;INITIALIZE DRV11B BEFORE TESTING
6251                                     ;*****
(3)                                     ;*TEST 1          TEST THAT ALL DRV11B REGS ARE ACCESSIBLE
(3)                                     ;*****
(2) 002442 000240                TST1:  <NOP>
(2)
(1) 002444 012737 002460 001106    MOV    #10$, $LPADR     ;;SET SCOPE LOOP ADDRESS
(2) 002452 012737 000001 001200    MOV    #1,$STSTN       ;;SET TEST NUMBER IN APT MAIL BOX
6252 002460 112737 000001 001102    10$:  MOV    #1,$STNM     ;SET TO TEST #1
6253 002466 012737 002522 001110    MOV    #1$, $LPERR     ;SET UP SCOPE LOOP ADRS
6254 002474 005037 001124          CLR    $GDDAT          ;NO DATA COMPARE
6255 002500 005037 001126          CLR    $BDDAT          ;NO DATA COMPARE
6256 002504 012737 002540 000004    MOV    #2$, @#ERRVEC   ;SET UP TIMEOUT RETURN ADRS
6257 002512 013700 001520          MOV    DRVWCR,RO       ;SET UP 1ST DRV11 BUS ADRS
6258 002516 012701 000004          MOV    #4,R1          ;SET UP REG COUNT
6259 002522 010037 001122    1$:  MOV    RO,$BDADR    ;SET UP CURRENT DRV BUS ADRS
6260 002526 005710                TST    (RO)            ;SEE IF THERE
6261 002530 005720                TST    (RO)+          ;BUMP TO NEXT
6262 002532 005301                DEC    R1              ;COUNT 4 OF THEM
6263 002534 001403                BEQ    3$              ;BR IF ALL DONE
6264 002536 000771                BR     1$              ;TRY NEXT
6265 002540 022626    2$:  CMP    (SP)+,(SP)+   ;FIX STACK SINCE NO RTI
6266 002542 104001                ERROR  1              ;BUS ADRS INDICATED DID NOT RESPOND
6267 002544 012737 000006 000004    3$:  MOV    #ERRVEC+2,@#ERRVEC ;RESTORE LOC 4
6268
6269                                     ;*****
(3)                                     ;*TEST 2          TEST THAT THE WORD COUNT REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
(3)                                     ;*****
(2) 002552 000004                TST2:  SCOPE
(2)
6270 002554 012737 002600 001110    MOV    #1$, $LPERR     ;SET UP SCOPE LOOP ADRS
6271 002562 013737 001520 001122    MOV    DRVWCR,$BDADR   ;SET UP WC REG ADRS
6272 002570 005000                CLR    RO              ;RO SAYS SHIFT PTRN WHEN 0
6273 002572 012737 177776 001124    MOV    #-2,$GDDAT     ;FLOAT 0 RIGHT TO LEFT
6274 002600 013777 001124 176712    1$:  MOV    $GDDAT,@DRVWCR ;LD WC
6275 002606 017737 176706 001126    MOV    @DRVWCR,$BDDAT  ;READ IT BACK
6276 002614 023737 001124 001126    CMP    $GDDAT,$BDDAT  ;CORRECT?
6277 002622 001401                BEQ    2$              ;BR IF SO
6278 002624 104002                ERROR  2              ;WORD COUNT WRITE/READ FAILURE
6279 002626 005137 001124    2$:  COM    $GDDAT        ;COMPELEMENT ZERO
6280 002632 005100                COM    RO              ;RO SAYS SHIFT LEFT WHEN = 0
6281 002634 001361                BNE    1$              ;TRY THE COMPLEMENT IF RO NOT 0
6282 002636 006337 001124    ASL    $GDDAT          ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
6283 002642 005237 001124    INC    $GDDAT          ;KEEP LSB SET
6284 002646 103754                BCS    1$              ;AGAIN TILL ALL PATRNS DONE
6285
6286                                     ;*****
(3)                                     ;*TEST 3          TEST THAT THE BUFFER ADDRESS REG IS WRITE/READABLE (FLOAT 0 COM PTRN)
(3)                                     ;*****
(2) 002650 000004                TST3:  SCOPE
(2)
6287 002652 012737 002676 001110    MOV    #1$, $LPERR     ;SET UP SCOPE LOOP ADRS
6288 002660 013737 001522 001122    MOV    DRVBAR,$BDADR   ;SET UP BA REG ADRS
```

```

6289 002666 005000          CLR      R0          ;RO SAYS SHIFT PTRN WHEN 0
6290 002670 012737 177774 001124  MOV      #-4,$GDDAT ;FLOAT 0 RIGHT TO LEFT
6291 002676 013777 001124 176616 1$:  MOV      $GDDAT,@DRVBAR ;LD BA
6292 002704 017737 176612 001126  MOV      @DRVBAR,$BDDAT ;READ IT BACK
6293 002712 042737 000001 001126  BIC      #BIT00,$BDDAT ;DON'T WANT BIT00
6294 002720 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
6295 002726 001401          BEQ      2$          ;BR IF SO
6296 002730 104002          ERROR    2          ;BUS ADRS WRITE/READ FAILURE
6297 002732 005137 001124          2$:  COM      $GDDAT      ;COMPLEMENT ZERO
6298 002736 042737 000001 001124  BIC      #BIT00,$GDDAT ;BIT 00 NOT INVOLVED
6299 002744 005100          COM      R0          ;RO SAYS SHIFT LEFT WHEN = 0
6300 002746 001353          BNE      1$          ;TRY THE COMPLEMENT IF RO NOT 0
6301 002750 006337 001124          ASL      $GDDAT      ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
6302 002754 103004          BCC      TST4        ;:NEXT TEST IF BIT 15 DONE
6303 002756 062737 000002 001124  ADD      #2,$GDDAT   ;KEEP ADDR LSB SET
6304 002764 000744          BR       1$          ;AGAIN TILL ALL PATTERNS DONE
6305
6306
    
```

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

```

(3)
(3)
(2) 002766 000004  TST4:  SCOPE
(2)
6307 002770 012737 003014 001110  MOV      #1$,$LPERR ;SET UP SCOPE LOOP ADRS
6308 002776 013737 001526 001122  MOV      DRVDBR,$BDADR ;SET UP DB REG ADRS
6309 003004 005000          CLR      R0          ;RO SAYS SHIFT PTRN WHEN 0
6310 003006 012737 177776 001124  MOV      #-2,$GDDAT ;FLOAT 0 RIGHT TO LEFT
6311 003014 013777 001124 176504 1$:  MOV      $GDDAT,@DRVDBR ;LD DB
6312 003022 017737 176500 001126  MOV      @DRVDBR,$BDDAT ;READ IT BACK
6313 003030 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
6314 003036 001401          BEQ      2$          ;BR IF SO
6315 003040 104002          ERROR    2          ;DATA BUFFER WRITE/READ FAILURE (LOOP BACK)
6316 003042 005137 001124          2$:  COM      $GDDAT      ;COMPLEMENT ZERO
6317 003046 005100          COM      R0          ;RO SAYS SHIFT LEFT WHEN = 0
6318 003050 001361          BNE      1$          ;TRY THE COMPLEMENT IF RO NOT 0
6319 003052 006337 001124          ASL      $GDDAT      ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
6320 003056 005237 001124          INC      $GDDAT      ;KEEP LSB SET
6321 003062 103754          BCS      1$          ;AGAIN TILL ALL PATRNS DONE
6322
6323
    
```

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

```

(3)
(3)
(2) 003064 000004  TST5:  SCOPE
(2)
6324 003066 013700 001526  MOV      DRVDBR,R0   ;GET DB REG ADRS
6325 003072 010037 001122  MOV      R0,$BDADR  ;SET UP DB REG ADRS
6326 003076 005010          CLR      (R0)       ;ZERO DATA BUFFER REG
6327 003100 012737 177177 001124  MOV      #177177,$GDDAT ;LD EXPECTED
6328 003106 012737 077776 001126  MOV      #77776,$BDDAT ;SEND DATA FROM 'BDDAT'
6329 003114 153760 001126 000001  BISB    $BDDAT,1(R0) ;LOAD HI BYTE DB
6330 003122 153710 001127          BISB    $BDDAT+1,(R0) ;LOAD LO BYTE DB
6331 003126 011037 001126          MOV      (R0),$BDDAT ;READ IT BACK
6332 003132 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
6333 003140 001401          BEQ      TST6        ;:NEXT TEST IF SO
6334 003142 104002          ERROR    2          ;DATA ERROR ON BYTE ADDRESSING THE DATA BUFFER REG
6335
6336
    
```

```
(3) ;*TEST 6 TEST THAT RESET CLEARS WORD COUNT, BUS ADDRESS & DATA REGS
(3) ;*****
(2) 003144 000004 TST6: SCOPE
(2)
(1) 003146 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
6337 003154 005037 001124 CLR $GDDAT ;LD EXPECTED
6338 003160 012777 177777 176332 MOV #-1,@DRVWCR ;SET ALL BITS - WC REG
6339 003166 012777 177776 176326 MOV #-2,@DRVBAR ;SET ALL BITS - BUS ADRS REG
6340 003174 012777 177777 176324 MOV #-1,@DRVDBR ;SET ALL BITS - DB OUT REG
6341 003202 000005 RESET ;DO A BUS RESET
6342 003204 017737 176310 001126 MOV @DRVWCR,$BDDAT ;READ WC REG
6343 003212 001404 BEQ 1$ ;BR IF CLRED
6344 003214 013737 001520 001122 MOV DRVWCR,$BDADR ;SET UP WC REG ADRS
6345 003222 104003 ERROR 3 ;RESET FAILED TO CLR WC REG
6346 003224 017737 176272 001126 1$: MOV @DRVBAR,$BDDAT ;READ BUS ADRS REG
6347 003232 042737 000001 001126 BIC #BIT00,$BDDAT ;DON'T WANT BITT00
6348 003240 001404 BEQ 2$ ;BR IF CLRED
6349 003242 013737 001522 001122 MOV DRVBAR,$BDADR ;SET UP BA REG ADRS
6350 003250 104003 ERROR 3 ;RESET FAILED TO CLR BUS ADRS REG
6351 003252 017737 176250 001126 2$: MOV @DRVDBR,$BDDAT ;READ DATA BUFFER REG
6352 003260 001404 BEQ TST7 ;NEXT TEST IF CLRED
6353 003262 013737 001526 001122 MOV DRVDBR,$BDADR ;SET UP DB REG ADRS
6354 003270 104003 ERROR 3 ;RESET FAILED TO CLR DATA BUFFER OUT REG
6355
6356 ;*****
```

```
(3) ;*TEST 7 TEST THAT THE CONTROL/STATUS REG IS WRITE/READABLE (COUNT PTRN)
(3) ;*****
(2) 003272 000004 TST7: SCOPE
(2)
(1) 003274 012737 000010 001160 MOV #10,$TIMES ;;DO 10 ITERATIONS
6357 003302 106427 000200 MTPS #200 ;DON'T WANT ANY INTRs
6358 003306 004537 010112 JSR R5,SETVEC ;SET UP INTR RETURN ADRS IN CASE
6359 003312 003410 3$ ;RETURN TO 3$ ON ILLEGAL INTR
6360 003314 013737 003334 001110 MOV 1$,SLPERR ;SET UP SCOPE LOOP ADRS
6361 003322 013737 001524 001122 MOV DRVCSR,$BDADR ;SET UP CSR ADRS
6362 003330 012700 160000 MOV #160000,R0 ;START AT 0 - HI BITS FOR NOISE
6363 003334 010037 001124 1$: MOV R0,$GDDAT ;LD EXPECTED
6364 003340 042737 167201 001124 BIC #167201,$GDDAT ;MASK TO WRITEABLE BITS
6365 003346 010077 176152 MOV R0,@DRVCSR ;LD CSR
6366 003352 017737 176146 001126 MOV @DRVCSR,$BDDAT ;READ IT BACK
6367 003360 042737 007200 001126 BIC #7200,$BDDAT ;DON'T LOOK AT STAT & RDY BITS
6368 003366 023737 001124 001126 CMP $GDDAT,$BDDAT ;CORRECT?
6369 003374 001401 BEQ 2$ ;BR IF SO
6370 003376 104002 ERROR 2 ;CONTROL/STATUS REG WRITE/READ FAILURE
6371 003400 062700 000002 2$: ADD #2,R0 ;ADVANCE COUNT PATTERN
6372 003404 001353 BNE 1$ ;WRITE NEXT PATTERN IF NOT ALL TESTED
6373 003406 000413 BR 4$ ;GO RESTORE VECTOR
6374 003410 022626 3$: CMP (SP)+,(SP)+ ;FIX STACK - SHOULD NOT HAVE INTR'ED
6375 003412 052737 000200 001124 BIS #200,$GDDAT ;CORRECT EXPECTED
6376 003420 017737 176100 001126 MOV @DRVCSR,$BDDAT ;READ CSR
6377 003426 042737 007000 001126 BIC #7000,$BDDAT ;DON'T WANT 'STAT' BITS
6378 003434 104005 ERROR 5 ;CPU FAILED TO LOCK OUT DRV11B INTR REQ
6379 003436 004737 010132 4$: JSR PC,RSTVEC ;GO RESTORE VECTOR
6380
6381 ;*****
```

```
(3) ;*TEST 10 TEST THAT RESET CLEARS ALL WRITEABLE BITS & SET READY IN CSR
```

(3)
(2) 003442 000004
(2)
(1) 003444 012737 000010 001160
6382 003452 013737 001524 001122
6383 003460 012737 000200 001124
6384 003466 012777 177776 176030
6385 003474 000005
6386 003476 017737 176022 001126
6387 003504 023737 001124 001126
6388 003512 001401
6389 003514 104003
6390
6391

TST10: SCOPE
MOV #10,\$TIMES ;;DO 10 ITERATIONS
MOV DRVCSR,\$BDADR ;;SET UP CSR ADRS
MOV #200,\$GDDAT ;;LD EXPECTED
MOV #-2,@DRVCSR ;;LD ALL CSR BITS
RESET ;;DO A BUS RESET
MOV @DRVCSR,\$BDDAT ;;READ CSR
CMP \$GDDAT,\$BDDAT ;;CORRECT?
BEQ TST11 ;;NEXT TEST IF CSR CLRED & READY SET
ERROR 3 ;;RESET FAILED TO SET UP THE CSR

(3)
(3)
(2) 003516 000004
(2)
6392 003520 013700 001524
6393 003524 010037 001122
6394 003530 005010
6395 003532 012737 010300 001124
6396 003540 012737 040020 001126
6397 003546 153760 001126 000001
6398 003554 153710 001127
6399 003560 011037 001126
6400 003564 023737 001124 001126
6401 003572 001401
6402 003574 104002
6403 003576 005010
6404
6405

*TEST 11 TEST THAT THE CSR IS BYTE ADDRESSABLE

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

(3)
(3)
(2) 003600 000004
(2)
6406 003602 012737 003630 001110
6407 003610 013737 001524 001122
6408 003616 012737 007000 001124
6409 003624 012700 000016
6410 003630 010077 175670
6411 003634 017737 175664 001126
6412 003642 042737 170777 001126
6413 003650 023737 001124 001126
6414 003656 001401
6415 003660 104004
6416 003662 162737 001000 001124
6417 003670 162700 000002
6418 003674 100355
6419
6420

*TEST 12 TEST THAT THE 3 'FNCT' BITS CONTROL THE 3 'STAT' BITS (COUNT PTRN)

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

(3)
(3)
(2) 003676 000004
(2)
(1) 003700 012737 000010 001160

*TEST 13 TEST THAT READY SET WILL CAUSE AN INTERRUPT AT LEVEL 0

TST13: SCOPE
MOV #10,\$TIMES ;;DO 10 ITERATIONS

```

6421 003706 106427 000200      MTPS      #200      :DONT WANT INTR YET
6422 003712 000005      RESET     :SET THE READY FLAG BY INIT
6423 003714 012777 003774 175606      MOV       #1$,@DRVCTO :SET UP PREMATURE INTR RETURN ADRS
6424 003722 013737 001524 001122      MOV       DRVCSR,$BDADR :SET UP CSR ADRS
6425 003730 012737 000300 001124      MOV       #300,$GDDAT  :LD EXPECTED (READY + IE)
5426 003736 106427 000000      MTPS      #0        :ALLOW AN INTR
6427 003742 021616      CMP       (SP),(SP)    :STALL
6428 003744 012777 004010 175556      MOV       #2$,@DRVCTO :SET UP EXPECTED INTR RETURN ADRS
6429 003752 052777 000100 175544      BIS       #BIT6,@DRVCSR :ENABLE THE EXPECTED INTERRUPT
6430 003760 021616      CMP       (SP),(SP)    :STALL
6431 003762 017737 175536 001126      MOV       @DRVCSR,$BDDAT :GET THE CSR
6432 003770 104005      ERROR    5          :READY FAILED TO CAUSE AN INTERRUPT
6433 003772 000417      BR       3$         :GO RESTORE VECTOR
6434 003774 022626      1$: CMP       (SP)+,(SP)+ :SHOULD NEVER GET HERE - IE NOT WORKING?
6435 003776 017737 175522 001126      MOV       @DRVCSR,$BDDAT :GET THE CSR
6436 004004 104005      ERROR    5          :READY INTERRUPTED WITHOUT THE IE BIT
6437 004006 000411      BR       3$         :GO RESTORE VECTOR
6438 004010 022626      2$: CMP       (SP)+,(SP)+ :FIX STACK SINCE NO RETURN
6439 004012 017737 175506 001126      MOV       @DRVCSR,$BDDAT :READ STATUS
6440 004020 023737 001124 001126      CMP       $GDDAT,$BDDAT :CORRECT?
6441 004026 001401      BEQ      3$         :BR IF SO
6442 004030 104005      ERROR    5          :INCORRECT STATUS ON READY INTR
6443 004032 004737 010132      3$: JSR       PC,RSTVEC  :GO RESTORE VECTOR
6444
6445
  
```

```

:*****
:*TEST 14      TEST THAT GO CLRS READY & FNCT 2 WILL SET IT
:*****
TST14: SCOPE
  
```

```

(3)
(3)
(2) 004036 000004
(2)
6446 004040 106427 000200      MTPS      #200      :DONT WANT ANY INTRS
6447 004044 013737 001524 001122      MOV       DRVCSR,$BDADR :SET UP CSR ADRS
6448 004052 005037 001124      CLR       $GDDAT       :EXPECT 0
6449 004056 012777 000001 175440      MOV       #1,@DRVCSR   :SET GO WHICH SHOU'LD CLR READY
6450 004064 017737 175434 001126      MOV       @DRVCSR,$BDDAT :READ THE CSR
6451 004072 023737 001124 001126      CMP       $GDDAT,$BDDAT :CORRECT?
6452 004100 001401      BEQ      1$         :BR IF SO
6453 004102 104006      ERROR    6          :THE GO BIT FAILED TO CLR READY
6454 004104 052777 000004 175412      1$: BIS       #4,@DRVCSR  :FNCT 2 SHOULD SET READY
6455 004112 012737 002204 001124      MOV       #2204,$GDDAT  :LD EXPECTED
6456 004120 017737 175400 001126      MOV       @DRVCSR,$BDDAT :GET CSR
6457 004126 023737 001124 001126      CMP       $GDDAT,$BDDAT :CORRECT?
6458 004134 001401      BEQ      TST15      :NEXT TEST IF SET
6459 004136 104006      ERROR    6          :FNCT 2 (VIA ATTN) FAILED TO SET READY
6460
6461
  
```

```

:*****
:*TEST 15      TEST THAT READY CONTROLS 'BAR' BITCO
:*****
TST15: SCOPE
  
```

```

(3)
(3)
(2) 004140 000004
(2)
6462 004142 012777 000004 175354      MOV       #4,@DRVCSR   :SET READY
6463 004150 013737 001522 001122      MOV       DRVBAR,$BDADR :SET UP BAR ADRS
6464 004156 012737 000001 001124      MOV       #1,$GDDAT    :EXPECT LSB OF BAR
6465 004164 005077 175332      CLR       @DRVBAR      :CLR BAR
6466 004170 017737 175326 001126      MOV       @DRVBAR,$BDDAT :READ BAR
6467 004176 023737 001124 001126      CMP       $GDDAT,$BDDAT :CORRECT?
6468 004204 001401      BEQ      1$         :BR IF SO
  
```

```

6469 004206 104002          ERROR 2          ;A00 FAILED TO READ A ONE(SB TIED TO RDY)
6470 004210 012777 000001 175306 1$: MOV #1,@DRVCSR ;SET GO(CLRS BAR BIT00)
6471 004216 017737 175300 001126 MOV @DRVBAR,$BDDAT ;READ BAR
6472 004224 001403          BEQ 2$          ;BR IF ZERO
6473 004226 005037 001124          CLR $GDDAT      ;EXPECTED ZERO
6474 004232 104002          ERROR 2          ;WHEN RDY CLRED-A00 FAILED TO READ A ZERO
6475 004234 012777 000004 175262 2$: MOV #4,@DRVCSR ;INSURE RDY SET BEFORE ADVANCING
6476
6477          ;*****
(3)          ;*TEST 16 TEST THAT 'CYCLE' WILL CLOCK THE DBR (IN)
(3)          ;*****
(2) 004242 000004          TST16: SCOPE
(2)
6478 004244 013737 001526 001122          MOV DRVDBR,$BDADR ;SET UP DBR ADRS
6479 004252 012737 125252 001124          MOV #125252,$GDDAT ;LD EXPECTED
6480 004260 013777 001124 175240          MOV $GDDAT,@DRVDBR ;LD DBR WITH #125252
6481 004266 012777 000400 175230          MOV #400,@DRVCSR ;SET CYCLE - SHOULD CLK DBR (IN)
6482 004274 012777 052525 175224          MOV #52525,@DRVDBR ;CHANGE DBR (OUT) DATA - SHOULD NOT AFFECT (IN)
6483 004302 017737 175220 001126          MOV @DRVDBR,$BDUAT ;READ DBR
6484 004310 023737 001124 001126          CMP $GDDAT,$BDDAT ;CORRECT?
6485 004316 001401          BEQ 1$          ;BR IF SO
6486 004320 104017          ERROR 17         ;CYCLE DID NOT LATCH DBR (IN) DATA
6487 004322 042777 000400 175174 1$: BIC #400,@DRVCSR ;REMOVE CYCLE
6488 004330 012737 052525 001124          MOV #52525,$GDDAT ;NOW EXPECT #52525
6489 004336 017737 175164 001126          MOV @DRVDBR,$BDDAT ;READ DBR
6490 004344 023737 001124 001126          CMP $GDDAT,$BDDAT ;CORRECT?
6491 004352 001401          BEQ TST17       ;NEXT TEST IF SO
6492 004354 104002          ERROR 2          ;DBR FAILED TO READ WHEN CYCLE CLRED (NORMAL)
6493
6494          ;*****
(3)          ;*TEST 17 TEST SINGLE 'DATI' NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)
(3)          ;*****
(2) 004356 000004          TST17: SCOPE
(2)
6495 004360 012737 004402 001110          MOV #1$,$LPERR ;SET UP SCOPE LOOP ADRS
6496 004366 004537 010112          JSR R5,SETVEC ;GO SET UP INTERRUPT RETURN
6497 004372 004476          2$          ;RETURN TO 2$ ON INTR
6498 004374 012700 177776          MOV #-2,R0 ;FLOAT ZERO RIGHT TO LEFT
6499 004400 005001          CLR R1 ;R1 CONTROLS DATA SHIFTING
6500 004402 012777 177777 175110 1$: MOV #-1,@DRVWCR ;DO ONE XFER
6501 004410 012777 015316 175104          MOV #DBUF,@DRVBAR ;GET DATA WORD FROM 'DBUF'
6502 004416 010037 015316          MOV R0,DBUF ;SET UP MEM DATA
6503 004422 012777 000101 175074          MOV #101,@DRVCSR ;SET IE & GO
6504 004430 052777 000400 175066          BIS #400,@DRVCSR ;SET CYCLE
6505 004436 106427 000000          MTPS #0 ;ENABLE THE INTR
6506 004442 013737 001524 001122          MOV DRVCSR,$BDADR ;SET UP CSR ADRS
6507 004450 012737 000700 001124          MOV #700,$GDDAT ;LD EXPECTED
6508 004456 017737 175042 001126          MOV @DRVCSR,$BDDAT ;READ THE CSR
6509 004464 042777 000100 175032          BIC #100,@DRVCSR ;CLR IE
6510 004472 104005          ERROR 5          ;WCO FAILED TO INTERRUPT (CHECK FOR WCO)
6511 004474 000442          BR 4$          ;GO RESTORE VECTOR
6512 004476 022626          2$: CMP (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
6513 004500 004537 010156          JSR R5,CKSTAT ;GO CHECK STATUS
6514 004504 000700          700          ;CSR STATUS EXPECTED
6515 004506 000001          1           ;# OF XFERS
6516 004510 104007          ERROR 7          ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
    
```

6517	004512	000433			BR	4\$:GO RESTORE VECTOR
6518	004514	104010			ERROR	10		:RETURN HERE IF WC ER - EXPECTED 0
6519	004516	000431			BR	4\$:GO RESTORE VECTOR
6520	004520	104011			ERROR	11		:RETURN HERE IF BAR ER - SHOULD = DBUF+2
6521	004522	000427			BR	4\$:GO RESTORE VECTOR
6522	004524	012737	015316	001120	MOV	#DBUF,\$GDADR		:RETURN HERE IF OK - SET UP XFER ADRS
6523	004532	010037	001124		RO,\$GDDAT			:LD EXPECTED
6524	004536	017737	174764	001126	MOV	@DRVDBR,\$BDDAT		:READ DATA XFERED
6525	004544	023737	001124	001126	CMP	\$GDDAT,\$BDDAT		:CORRECT?
6526	004552	001405			BEQ	3\$:BR IF SO
6527	004554	013737	001526	001122	MOV	DRVDBR,\$BDADR		:SET UP DBR ADRS
6528	004562	104012			ERROR	12		:DATA ER - DBR CONTAINS WRONG DATA
6529	004564	000406			BR	4\$:GO RESTORE VECTOR
6530	004566	005100			3\$: COM	RO		:RETURN HERE ON GOOD DATA - NOW COM PATRN
6531	004570	005101			COM	R1		:KEEP TRACK OF COMPLEMENT
6532	004572	001303			BNE	1\$:DO COMPLEMENT OF THIS FLOATING ZERO IF 0
6533	004574	006300			ASL	RO		:WAS DONE - NOW SHIFT ZERO LEFT
6534	004576	005200			INC	RO		:KEEP LSB SET
6535	004600	103700			BCS	1\$:AGAIN TILL ZERO BIT IN CARRY
6536	004602	004737	010132		4\$: JSR	PC,RSTVEC		:GO RESTORE VECTOR
6537								
6538								
(3)								
(3)								
(2)	004606	000004						
(2)								
6539	004610	012737	004632	001110	MOV	#1\$,\$LPERR		:SET UP SCOPE LOOP ADRS
6540	004616	004537	010112		JSR	R5,SETVEC		:GO SET UP INTERRUPT RETURN
6541	004622	004720			2\$:RETURN TO 2\$ ON INTR
6542	004624	012700	177776		MOV	#-2,RO		:FLOAT ZERO RIGHT TO LEFT
6543	004630	005001			CLR	R1		:R1 CONTROLS DATA SHIFTING
6544	004632	012777	177777	174660	1\$: MOV	#-1,@DRVWCR		:DO ONE XFER
6545	004640	012777	015316	174654	MOV	#DBUF,@DRVBAR		:WRITE DATA WORD TO 'DBUF'
6546	004646	010077	174654		MOV	RO,@DRVDBR		:SET UP DATA IN DBR
6547	004652	012777	000103	174644	MOV	#103,@DRVCSR		:SET IE, GO & FNCT1 (C1 CONTROL)
6548	004660	052777	000400	174636	BIS	#400,@DRVCSR		:SET CYCLE
6549	004666	106427	000000		MTPS	#0		:ENABLE THE INTR
6550	004672	013737	001524	001122	MOV	DRVCSR,\$BDADR		:SET UP CSR ADRS
6551	004700	012737	001702	001124	MOV	#1702,\$GDDAT		:LD EXPECTED
6552	004706	017737	174612	001126	MOV	@DRVCSR,\$BDDAT		:READ THE CSR
6553	004714	104005			ERROR	5		:WCO FAILED TO INTERRUPT (CHECK FOR WCO)
6554	004716	000442			BR	4\$:GO RESTORE VECTOR
6555	004720	022626			2\$: CMP	(SP)+,(SP)+		:INTR RETURNS HERE - FIX STACK SINCE NO RTI
6556	004722	004537	010156		JSR	R5,CKSTAT		:GO CHECK STATUS
6557	004726	001702			1702			:CSR STATUS EXPECTED
6558	004730	000001			1			:# OF XFERS
6559	004732	104007			ERROR	7		:RETURN HERE IF STATUS ER - EXPECTED STAT C.
6560								:CYCLE, READY, IE & FNCT 1
6561	004734	000433			BR	4\$:GO RESTORE VECTOR
6562	004736	104010			ERROR	10		:RETURN HERE IF WC ER - EXPECTED 0
6563	004740	000431			BR	4\$:GO RESTORE VECTOR
6564	004742	104011			ERROR	11		:RETURN HERE IF BAR ER - SHOULD = DBUF+2
6565	004744	000427			BR	4\$:GO RESTORE VECTOR
6566	004746	012737	015316	001120	MOV	#DBUF,\$GDADR		:RETURN HERE IF OK - SET UP XFER ADRS
6567	004754	010037	001124		MOV	RO,\$GDDAT		:LD EXPECTED
6568	004760	013737	015316	001126	MOV	DBUF,\$BDDAT		:GET DATA XFERED

 :*TEST 20 TEST SINGLE 'DATO' NPR TRANSFERS (FLOATING 0 COMPLEMENT PATRN)

 TST20: SCOPE

```

6569 004766 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CORRECT?
6570 004774 001405                    BEQ      3$                 ;BR IF SO
6571 004776 013737 001526 001122      MOV      DRVDBR,$BDADR     ;SET UP DBR ADRS
6572 005004 104013                    ERROR    13                 ;DATA ER - MEM CONTAINS WRONG DATA
6573 005006 000406                    BR       4$                 ;GO RESTORE VECTOR
6574 005010 005100                    3$:    COM      R0           ;RETURN HERE ON GOOD DATA - NOW COM PATRN
6575 005012 005101                    COM      R1           ;KEEP TRACK OF COMPLEMENT
6576 005014 001306                    BNE     1$                 ;DO COMPLEMENT OF THIS FLOATING ZERO IF 0
6577 005016 006300                    ASL     R0           ;COMPLEMENT WAS DONE - NOW SHIFT ZERO LEFT
6578 005020 005200                    INC     R0           ;KEEP LSB SET
6579 005022 103703                    BCS     1$                 ;AGAIN TILL ZERO BIT IN CARRY
6580 005024 004737 010132                    4$:    JSR      PC,RSTVEC     ;GO RESTORE VECTOR
6581
6582
(3)
(3)
(2) 005030 000004
(2)
(1) 005032 012737 000010 001160      MOV      #10,$TIMES       ;;DO 10 ITERATIONS
6583 005040 004537 010112                    JSR      R5,SETVEC        ;GO SET UP INTERRUPT RETURN
6584 005044 005142                    1$     ;RETURN TO 1$ ON INTR
6585 005046 004737 010336                    JSR      PC,LDBUF         ;GO LOAD PUFFER WITH COMPLEMENTING PATRN
6586 005052 012777 177470 174440      MOV      #-200,@DRVWCR    ;LOAD WC REG - WILL DO 200 XFERS
6587 005060 012777 015316 174434      MOV      #DBUF,@DRVBAR   ;SET UP CURRENT ADRS
6588 005066 106427 000000                    MTPS    #0               ;ENABLE THE INTR
6589 005072 012777 000101 174424      MOV      #101,@DRVCSR    ;SET IE & GO
6590 005100 052777 000400 174416      BIS      #400,@DRVCSR    ;SET CYCLE
6591 005106 013737 001524 001122      MOV      DRVCSR,$BDADR   ;SET UP CSR ADRS
6592 005114 012737 000700 001124      MOV      #700,$GDDAT    ;LD EXPECTED
6593 005122 017737 174376 001126      MOV      @DRVCSR,$BDDAT ;READ THE CSR
6594 005130 042777 000100 174366      BIC      #100,@DRVCSR    ;CLR INTR ENABLE
6595 005136 104005                    ERROR    5                 ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
6596 005140 000434                    BR       2$                 ;GO RESTORE VECTOR
6597 005142 022626                    1$:    CMP      (SP)+,(SP)+ ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
6598 005144 004537 010156                    JSR      R5,CKSTAT        ;GO CHECK STATUS
6599 005150 000700                    700     ;CSR STATUS EXPECTED
6600 005152 000310                    200     ;# OF XFERS
6601 005154 104007                    ERROR    7                 ;RETURN HERE IF STATUS ER - EXPECTED CYCLE, READY & IE
6602 005156 000425                    BR       2$                 ;GO RESTORE VECTOR
6603 005160 104010                    ERROR    10                ;RETURN HERE IF WC ER - EXPECTED 0
6604 005162 000423                    BR       2$                 ;GO RESTORE VECTOR
6605 005164 104011                    ERROR    11                ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
6606 005166 000421                    BR       2$                 ;GO RESTORE VECTOR
6607 005170 012737 016134 001120      MOV      #DBUF+616,$GDADR ;OK - SET UP LAST XFER ADRS WHERE #70707 SHOULD BE
6608 005176 012737 070707 001124      MOV      #70707,$GDDAT   ;LD EXPECTED
6609 005204 017737 174316 001126      MOV      @DRVDBR,$BDDAT ;DBR SHOULD HAVE LAST DATUM
6610 005212 023737 001124 001126      CMP      $GDDAT,$BDDAT   ;CORRECT?
6611 005220 001404                    BEQ     2$                 ;BR IF SO
6612 005222 013737 001526 001122      MOV      DRVDBR,$BDADR   ;SET UP DBR ADRS
6613 005230 104012                    ERROR    12                ;DATA ER - DBR DID NOT CONTAIN EXPECTED LAST XFER
6614 005232 004737 010132                    2$:    JSR      PC,RSTVEC     ;GO RESTORE VECTOR
6615
6616
(3)
(3)
(2) 005236 000004
TST22: SCOPE

```

```

(2)
(1) 005240 012737 000010 001160      MOV    #10,$TIMES      ;;DO 10 ITERATIONS
6617 005246 004537 010112      JSR    R5,SETVEC      ;GO SET UP INTERRUPT RETURN
6618 005252 005352                1$    ;RETURN TO 1$ ON INTR
6619 005254 012777 177470 174236      MOV    #-200.,@DRVWCR  ;WORD WC REG - WILL DO 200 XFER'S
6620 005262 012777 015316 174232      MOV    #DBUF,@DRVBAR  ;SET UP CURRENT ADRS
6621 005270 012777 177377 174230      MOV    #177377,@DRVDBR ;THIS WILL BE WRITTEN TO MEM
6622 005276 106427 000000          MTPS   #0              ;ENABLE THE INTR
6623 005302 012777 000103 174214      MOV    #103,@DRVCSR   ;SET IE, FNCT 1 & GO
6624 005310 052777 000400 174206      BIS    #400,@DRVCSR   ;SET CYCLE
6625 005316 013737 001524 001122      MOV    DRVCSR,$BDDADR ;SET UP CSR ADRS
6626 005324 012737 001702 001124      MOV    #1702,$GDDAT   ;LD EXPECTED
6627 005332 017737 174166 001126      MOV    @DRVCSR,$BDDAT ;READ THE CSR
6628 005340 042777 000100 174156      BIC    #100,@DRVCSR   ;CLR INTR ENABLE
6629 005346 104005                ERROR  5              ;WCO FAILED TO INTERRUPT (SNGL CYCL ON COULD CAUSE THIS)
6630 005350 000416                BR     2$             ;GO RESTORE VECTOR
6631 005352 022626                1$:  CMP    (SP)+,(SP)+  ;INTR RETURNS HERE - FIX STACK SINCE NO RTI
6632 005354 004537 010156      JSR    R5,CKSTAT      ;GO CHECK STATUS
6633 005360 001702                1702  ;CSR STATUS EXPECTED
6634 005362 000310                200.  ;# OF XFERS
6635 005364 104007                ERROR  7              ;RETURN HERE IF STATUS ER - EXPECTED STAT C,
6636                                ;CYCLE, READY, IE & FNCT 1
6637 005366 000407                BR     2$             ;GO RESTORE VECTOR
6638 005370 104010                ERROR  10            ;RETURN HERE IF WC ER - EXPECTED 0
6639 005372 000405                BR     2$             ;GO RESTORE VECTOR
6640 005374 104011                ERROR  11            ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
6641 005376 000403                BR     2$             ;GO RESTORE VECTOR
6642 005400 004737 010460      JSR    PC,CKDAT       ;RETURN HERE IF OK - NOW GO CHECK DATA
6643 005404 104013                ERROR  13            ;RETURN HERE IF DATA ER - DBR CONTAINS WRONG DATA
6644 005406 004737 010132      2$:  JSR    PC,RSTVEC    ;RETURN HERE IF DATA CHECK OK - GO RESTORE VECTOR
6645
6646                                ;*****
(3)                                ;*TEST 23 TEST THAT THE CPU IS LOCKED OUT WITH SINGLE CYCLE OFF
(3)                                ;*****
(2) 005412 000004      TST23: SCOPE
(2)
(1) 005414 012737 000010 001160      MOV    #10,$TIMES      ;;DO 10 ITERATIONS
6647 005422 013737 001524 001122      MOV    DRVCSR,$BDDADR ;SET UP CSR ADRS
6648 005430 004537 010112      JSR    R5,SETVEC      ;GO SET UP INTR RETURN
6649 005434 005544                3$    ;RETURN TO 3$ ON INTR
6650 005436 012700 000010          MOV    #10,R0         ;DO EIGHT 200 WORD XFER'S
6651 005442 005037 001126          CLR    $BDDAT         ;USR $BDDAT AS A COUNTER
6652 005446 012777 177470 174044      1$:  MOV    #-200.,@DRVWCR ;DO 200 XFERS (DATI'S)
6653 005454 012777 015316 174040      MOV    #DBUF,@DRVBAR ;FROM DBUF
6654 005462 106427 000000          MTPS   #0              ;ALLOW AN INTR
6655 005466 012777 000101 174030      MOV    #101,@DRVCSR   ;SET IE & GO
6656 005474 052777 000400 174022      BIS    #400,@DRVCSR   ;SET CYCLE
6657 005502 000240                NOP                    ;FREEBEE
6658 005504 000240                NOP
6659 005506 000240                NOP
6660 005510 005237 001126                2$:  INC    $BDDAT         ;START COUNTING - SHOULD NEVER GET HERE
6661 005514 001375                BNE    2$             ;UNTIL 64K
6662 005516 012737 000700 001124      MOV    #700,$GDDAT   ;LD EXPECTED
6663 005524 017737 173774 001126      MOV    @DRVCSR,$BDDAT ;READ STATUS
6664 005532 042777 000100 173764      BIC    #100,@DRVCSR   ;CLR IE
6665 005540 104005                ERROR  5              ;NO INTERRUPT ON 200 DATI'S

```



```

6666 005542 000407          BR      4$          ;GO RESTORE VECTOR
6667 005544 022626          3$:  CMP      (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
6668 005546 005300          DEC      R0          ;DONE 8 TIMES?
6669 005550 001336          BNE     1$          ;BR IF NOT
6670 005552 005737 001126    TST     $BDDAT      ;SHOULD STILL BE ZERO
6671 005556 001401          BEQ     4$          ;BR IF SO
6672 005560 104014          ERROR   14          ;BURST MD (SINGLE CYCLE=0) FAILS TO LOCK OUT CPU
6673 005562 004737 010132    4$:  JSR     PC,RSTVEC ;GO RESTORE VECTOR
6674
6675

```

```

:*****
:*TEST 24 TEST THAT THE CPU IS NOT LOCKED OUT WITH SINGLE CYCLE ON
:*****

```

```

(3)
(3)
(2) 005566 000004          TST24: SCOPE
(2)
(1) 005570 012737 000010 001160    MOV     #10,$TIMES ;:DO 10 ITERATIONS
6676 005576 013737 001524 001122    MOV     DRVCSR,$BDADR ;SET UP CSR ADRS
6677 005604 004537 010112    JSR     R5,SETVEC  ;GO SET UP INTR RETURN
6678 005610 005716          3$:  MOV     #10,R0    ;RETURN TO 3$ ON INTR
6679 005612 012700 000010    MOV     #0,$BDDAT  ;DO EIGHT 200 WORD XFER'S
6680 005616 012737 000000 001126    MOV     #-200,@DRVWCR ;USE $BDDAT AS A COUNTER
6681 005624 012777 177470 173666    1$:  MOV     #DBUF,@DRVBAR ;DO 200 XFERS (DATI'S)
6682 005632 012777 015316 173662    MOV     MTPS #0    ;FROM DBUF
6683 005640 106427 000000    MTPS   #0          ;ALLOW AN INTR
6684 005644 012777 000111 173652    MOV     #111,@DRVCSR ;SET IE, FNCT3 & GO
6685 005652 052777 000400 173644    BIS     #400,@DRVCSR ;SET CYCLE
6686 005660 000240          NOP                    ;FREEBEE
6687 005662 005237 001126          2$:  INC     $BDDAT    ;START COUNTING
6688 005666 001375          BNE     2$          ;UNTIL 64K - SHOULD INTR BEFORE OVERFLOW
6689 005670 012737 004710 001124    MOV     #4710,$GDDAT ;LD EXPECTED
6690 005676 017737 173622 001126    MOV     @DRVCSR,$BDDAT ;READ STATUS
6691 005704 042777 000100 173612    BIC     #100,@DRVCSR ;CLR IE
6692 005712 104005          ERROR   5          ;NO INTERRUPT ON 200 DATI'S (WITH SINGLE CYCLE)
6693 005714 000423          BR      5$          ;GO RESTORE VECTOR
6694 005716 022626          3$:  CMP     (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
6695 005720 005300          DEC     R0          ;DONE 8 TIMES?
6696 005722 001340          BNE     1$          ;BR IF NOT
6697 005724 022737 000000 001126    CMP     #0,$BDDAT  ;$BDDAT SHOULD HAVE BEEN COUNTED
6698 005732 103401          BCS     4$          ;BR IF SO
6699 005734 104015          ERROR   15         ;CPU APPEARED LOCKED OUT WITH SINGLE CYCLE SET
6700 005736 017737 173562 001126    4$:  MOV     @DRVCSR,$BDDAT ;READ STATUS
6701 005744 012737 004710 001124    MOV     #4710,$GDDAT ;LD EXPECTED
6702 005752 023737 001124 001126    CMP     $GDDAT,$BDDAT ;CORRECT?
6703 005760 001401          BEQ     5$          ;BR IF SO
6704 005762 104007          ERROR   7          ;STATUS INCORRECT ON XFER WITH SINGLE CYCLE SET
6705 005764 004737 010132    5$:  JSR     PC,RSTVEC  ;GO RESTORE VECTOR
6706
6707

```

```

:*****
:*TEST 25 TEST THAT MAINT MODE CONTROLS FNCT BITS, XFER DIR & SINGLE CYCLE
:*****

```

```

(3)
(3)
(2) 005770 000004          TST25: SCOPE
(2)
(1) 005772 012737 000200 001160    MOV     #200,$TIMES ;:DO 200 ITERATIONS
6708 006000 004537 010112    JSR     R5,SETVEC  ;GO SET UP INTR RETURN
6709 006004 006122          2$:  MOV     PC,LDBUF1 ;RETURN TO 2$ ON INTR
6710 006006 004737 010410          JSR     PC,LDBUF1  ;GO SET UP DBUF (SPECIAL COM PATTERN)
6711 006012 012737 011702 006130    MOV     #11702,3$ ;3$ CONTAINS EXPECTED STATUS

```

```

6712 006020 012737 000001 006132      MOV      #1,4$      ;4$ CONTAINS THE CURRENT XFER NO # (MAX 8)
6713 006026 106427 000000      MTPS     #0        ;ALLOW INTR
6714 006032 012777 015316 173462      MOV      #DBUF,@DRVBAR ;SET UP CURRENT ADRS
6715 006040 013777 006132 173452      MOV      4$,@DRVWCR   ;GET XFER #
6716 006046 005477 173446      NEG      @DRVWCR      ;NEGATE FOR WC
6717 006052 012777 010101 173444      MOV      #10101,@DRVCSR ;SET UP MAINT, IE & GO
6718 006060 052777 000400 173436      BIS      #400,@DRVCSR  ;SET CYCLE
6719 006066 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRS
6720 006074 013737 006130 001124      MOV      3$, $GDDAT   ;LD EXPECTED
6721 006102 017737 173416 001126      MOV      @DRVCSR,$BDDAT ;READ STATUS
6722 006110 042777 000100 173406      BIC      #100,@DRVCSR  ;DISABLE IE
6723 006116 104005      ERROR    5          ;NO INTR ON XFER (IN MAINT MD)
6724 006120 000440      BR       6$        ;GO RESTORE VECTOR
6725 006122 022626      CMP      (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
6726 006124 004537 010156      JSR      R5,CKSTAT   ;GO CHECK STATUS
6727 006130 011702      3$:     11702      ;THIS LOCATION WILL CONTAIN EXPECTED STATUS
6728 006132 000001      4$:     1          ;THIS LOCATION WILL CONTAIN CURRENT XFER # (MAX 8)
6729 006134 104007      ERROR    7          ;RETURN HERE IF STATUS ER - WILL EXPECT
6730                                     ;MAINT, COUNT INCREASE OF FNCT & STAT BITS,
6731                                     ;CYCLE, READY & IE
6732 006136 000431      BR       6$        ;GO RESTORE VECTOR
6733 006140 104010      ERROR    10       ;RETURN HERE IF WC ER - SHOULD BE 0
6734 006142 000427      BR       6$        ;GO RESTORE VECTOR
6735 006144 104011      ERROR    11       ;RETURN HERE IF BAR ER-
6736 006146 000425      BR       6$        ;GO RESTORE VECTOR
6737 006150 062737 001002 006130      ADD      #1002,3$    ;RETURN HERE IF OK - ADVANCE EXPECTED STATUS LOC
6738 006156 032737 020000 006130      BIT      #BIT13,3$  ;LOOK FOR OVERFLOW
6739 006164 001403      BEQ     5$        ;BR IF NOT
6740 006166 012737 010700 006130      MOV      #10700,3$  ;FNCT & STAT BITS SHOULD BE ZERO THIS TIME
6741 006174 005237 006132 5$:      INC      4$        ;ADVANCE CURRENT XFER #
6742 006200 022737 000011 006132      CMP      #11,4$    ;HAVE 10 XFERS BEEN DONE
6743 006206 001307      BNE     1$        ;BR IF NOT
6744 006210 004537 010526      JSR      R5,CKDAT1  ;NOW GO CHECK DATA
6745 006214 000010      10      ;# OF XFER'S TO CHECK
6746 006216 104013      ERROR    13       ;RETURN HERE IF DATA ER - (WITH MAINT SET)
6747 006220 000240      NOP     ;RESTORE VECTOR NEXT
6748 006222 004737 010132 6$:      JSR      PC,RSTVEC  ;GO RESTORE VECTOR
6749
6750      ;:*****
(3)      ;*TEST 26      TEST THAT A DATI FROM A NON-EXISTANT BUS ADRS SETS 'NEX'
(3)      ;:*****
(2) 006226 000004      TST26: SCOPE
(2)
(1) 006230 012737 000100 001160      MOV      #100,$TIMES ;:DO 100 ITERATIONS
6751 006236 012700 000002      MOV      #2,R0      ;R0 WHEN ZERO SAYS CLR 'NEX' WITH RESET
6752 006242 004537 010112 1$:      JSR      R5,SETVEC  ;GO SET UP INTERRUPT RETURN
6753 006246 006340      2$      ;RETURN TO 2$ ON TIMEOUT INTR
6754 006250 012777 177777 173242      MOV      #-1,@DRVWCR ;SET UP FGR ONE XFER'S
6755 006256 012777 160000 173236      MOV      #160000,@DRVBAR ;SET UP CA TO 160000 (RESERVED)
6756 006264 106427 000000      MTPS     #0        ;ALLOW INTR
6757 006270 012777 000161 173226      MOV      #161,@DRVCSR ;SET IE, XAD 17,16 & GO
6758 006276 052777 000400 173220      BIS      #400,@DRVCSR ;SET CYCLE
6759 006304 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRS - SHOULD NEVER GET HERE
6760 006312 012737 140760 001124      MOV      #140760,$GDDAT ;LD EXPECTED
6761 006320 017737 173200 001126      MOV      @DRVCSR,$BDDAT ;GIVE THEM THE STATUS
6762 006326 042777 000100 173170      BIC      #100,@DRVCSR ;CLR IE

```

```

6763 006334 104016          ERROR 16          :NEX FAILED TO CAUSE AN INTERRUPT
6764 006336 000521          BR 7$           :GO RESTORE VECTOR
6765 006340 022626          2$: CMP (SP)+,(SP)+ :SHOULD INTR RETURN HERE - FIX STACK
6766 006342 017737 173156 001126      MOV @DRVCSR,$BDDAT :READ THE CSR
6767 006350 012737 140760 001124      MOV #140760,$GDDAT :LD EXPECTED
6768 006356 023737 001124 001126      CMP $GDDAT,$BDDAT :CORRECT?
6769 006364 001405          BEQ 3$          :BR IF SO
6770 006366 013737 001524 001122      MOV DRVCSR,$BDADR :SET UP CSR ADRS
6771 006374 104016          ERROR 16          :STATUS ER - EXPECTED
6772                                :ER, NEX, CYCLE, READY, IE, XAD17 & XAD16
6773 006376 000501          BR 7$           :GO RESTORE VECTOR
6774 006400 017737 173114 001126      3$: MOV @DRVWCR,$BDDAT :READ WORD COUNT
6775 006406 012737 177777 001124      MOV #-1,$GDDAT    :SHOULD STILL HAVE -1
6776 006414 023737 001124 001126      CMP $GDDAT,$BDDAT :CORRECT?
6777 006422 001405          BEQ 4$          :BR IF SO
6778 006424 013737 001520 001122      MOV DRVWCR,$BDADR :SET UP WCR ADRS
6779 006432 104010          ERROR 10          :WC INCREMENTED ON A TIMEOUT ER
6780 006434 000462          BR 7$           :GO RESTORE VECTOR
6781 006436 017737 173060 001126      4$: MOV @DRVBAR,$BDDAT :READ BUFFER ADRS
6782 006444 012737 160000 001124      MOV #160000,$GDDAT :SHOULD NOT HAVE INCREMENTED
6783 006452 023737 001124 001126      CMP $GDDAT,$BDDAT :CORRECT?
6784 006460 001405          BEQ 5$          :BR IF SO
6785 006462 013737 001522 001122      MOV DRVBAR,$BDADR :SET UP BAR ADRS
6786 006470 104011          ERROR 11          :BAR INCREMENTED ON A TIMEOUT ER
6787 006472 000443          BR 7$           :GO RESTORE VECTOR
6788 006474 005300          5$: DEC R0         :KEEP TRACK ON HOW TO CLR
6789 006476 001422          BEQ 6$          :BR IF CLR BY RESET
6790 006500 042777 040000 173016      BIC #40000,@DRVCSR :WRITE NEX TO ZERO
6791 006506 017737 173012 001126      MOV @DRVCSR,$BDDAT :READ CSR
6792 006514 012737 000760 001124      MOV #760,$GDDAT   :LD EXPECTED
6793 006522 023737 001124 001126      CMP $GDDAT,$BDDAT :CORRECT?
6794 006530 001644          BEQ 1$          :BR IF SO + REPEAT TEST FOR RESET TEST
6795 006532 013737 001524 001122      MOV DRVCSR,$BDADR :SET UP CSR ADRS
6796 006540 104016          ERROR 16          :'NEX' FAILED TO WRITE TO ZERO
6797 006542 000417          BR 7$           :GO RESTORE VECTOR
6798 006544 000005          6$: RESET        :ISSUE BUS RESET
6799 006546 017737 172752 001126      MOV @DRVCSR,$BDDAT :READ THE CSR
6800 006554 012737 000200 001124      MOV #200,$GDDAT   :EXPECT ONLY READY
6801 006562 023737 001124 001126      CMP $GDDAT,$BDDAT :CORRECT?
6802 006570 001404          BEQ 7$          :BR IF SO
6803 006572 013737 001524 001122      MOV DRVCSR,$BDADR :SET UP CSR ADRS
6804 006600 104016          ERROR 16          :RESET FAILED TO CLR 'NEX'
6805 006602 004737 010132      7$: JSR PC,RSTVEC  :GO RESTORE VECTOR
6806                                :*****
        (3)                                :*TEST 27 TEST 200 NPR TRANSFERS IN MAINT MODE
        (3)                                :*****
        (2) 006606 000004          TST27: SCOPE
        (2)
        (1) 006610 012737 000010 001160      MOV #10,$TIMES    ;;DO 10 ITERATIONS
6807 006616 004537 010112          JSR R5,SETVEC    :GO SET UP INTR RETURN
6808 006622 006734          2$           :RETURN TO 2$ ON INTR
6809 006624 004737 010410          JSR PC,LDBUF1    :GO SET UP DBUF (SPECIAL COM PATTERN)
6810 006630 012777 015316 172664      MOV #DBUF,@DRVBAR :SET UP CURRENT ADRS
6811 006636 012777 177470 172654      MOV #-200.,@DRVWCR :SET UP FOR 200 XFER'S
6812 006644 106427 000000          MTPS #0         :ALLOW INTR
6813 006650 012777 010101 172646      MOV #10101,@DRVCSR :SET MAINT, IE & GO
  
```

```

6814 006656 052777 000400 172640      BIS      #400,@DRVCSR      ;SET CYCLE
6815 006664 012737 000000 001126      MOV      #0,$BDDAT    ;SET UP A COUNTER
6816 006672 005237 001126      1$:     INC      $BDDAT    ;COUNT AWAY
6817 006676 001375          BNE      1$          ;WAIT TILL DONE - SHOULD INTR BEFORE OVFL0
6818 006700 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRS
6819 006706 012737 010700 001124      MOV      #10700,$GDDAT ;LD EXPECTED
6820 006714 017737 172604 001126      MOV      @DRVCSR,$BDDAT ;READ STATUS
6821 006722 042777 000100 172574      BIC      #100,@DRVCSR  ;DISABLE IE
6822 006730 104005          ERROR    5          ;NO INTR AFTER 200 MAINT MODE XFER'S
6823 006732 000420          BR      3$          ;GO RESTORE VECTOR
6824 006734 022626      2$:     CMP      (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
6825 006736 004537 010156      JSR      R5,CKSTAT    ;GO CHECK STATUS
6826 006742 010700          10700    ;EXPECTED STATUS
6827 006744 000310          200.    ;# OF XFERS
6828 006746 104007          ERROR    7          ;RETURN HERE IF STATUS ER - EXPECTED MAINT.
6829          ;CYCLE, READY & IE
6830 006750 000411          BR      3$          ;GO RESTORE VECTOR
6831 006752 104010          ERROR    10         ;RETURN HERE IF WC ER - SHOULD BE 0
6832 006754 000407          BR      3$          ;GO RESTORE VECTOR
6833 006756 104011          ERROR    11         ;RETURN HERE IF BAR ER - SHOULD = DBUF+620
6834 006760 000405          BR      3$          ;GO RESTORE VECTOR
6835 006762 004537 010526      JSR      R5,CKDAT1    ;RETURN HERE IF OK - NOW GO CHECK DATA
6836 006766 000310          200.    ;# OF XFER'S TO CHECK
6837 006770 104013          ERROR    13         ;RETURN HERE IF DATA ER - (WITH MAINT SET)
6838 006772 000240          NOP          ;RESTORE VECTOR NEXT
6839 006774 004737 010132      3$:     JSR      PC,RSTVEC ;GO RESTORE VECTOR
6840          ;*****
        ;*TEST 30      TEST A 200 WORD MAINT MODE XFER TO EACH ADDITIONAL AVAILABLE 4K
        ;*****
        (3)
        (3)
        (2) 007000 000004      TST30:  SCOPE
        (2)
        (1) 007002 012737 000005 001160      MOV      #5,$TIMES    ;;DO 5 ITERATIONS
6841 007010 004537 010112      JSR      R5,SETVEC    ;GO SET UP INTR RETURN
6842 007014 007150          3$          ;RETURN TO 3$ ON INTR
6843 007016 005037 001542          CLR      DBUF         ;GET LOWEST BUFFER ADRS
6844 007022 062737 020000 001542      1$:     ADD      #20000,DBUF ;POINT TO NEXT 4K
6845 007030 023737 001540 001542      CMP      CORSZ,DBUF   ;IS THE 4K THERE?
6846 007036 001465          BEQ      4$          ;BR IF NOT
6847 007040 004737 010410          JSR      PC,LDBUF1    ;GO SET UP SPECIAL COMPEMENT PATTERN
6848 007044 013777 001542 172450      MOV      DBUF,@DRVBAR ;SET UP BUFFER ADRS
6849 007052 012777 177470 172440      MOV      #-200.,@DRVWCR ;SET UP FOR 200 XFER'S
6850 007060 106427 000000          MTPS     #0          ;ALLOW INTR
6851 007064 012777 010101 172432      MOV      #10101,@DRVCSR ;SET MAINT,IE & GO
6852 007072 052777 000400 172424      BIS      #400,@DRVCSR ;SET CYCLE
6853 007100 012737 000000 001126      MOV      #0,$BDDAT    ;SET UP A COUNTER
6854 007106 005237 001126      2$:     INC      $BDDAT    ;COUNT AWAY
6855 007112 001375          BNE      2$          ;SHOULD ALWAYS INTR FROM THIS LOOP
6856 007114 013737 001524 001122      MOV      DRVCSR,$BDADR ;SET UP CSR ADRA
6857 007122 012737 010700 001124      MOV      #10700,$GDDAT ;LD EXPECTED
6858 007130 017737 172370 001126      MOV      @DRVCSR,$BDDAT ;READ CSR
6859 007136 042777 000100 172360      BIC      #100,@DRVCSR  ;DISABLE IE
6860 007144 104005          ERROR    5          ;NO INTR AFTER 200 MAINT MODE XFER'S
6861 007146 000421          BR      4$          ;GO RESTORE VECTOR
6862 007150 022626      3$:     CMP      (SP)+,(SP)+ ;RETURN HERE ON INTR - FIX STACK SINCE NO RTI
6863 007152 004537 010156      JSR      R5,CKSTAT    ;GO CHECK STATUS
6864 007156 010700          10700    ;EXPECTED STATUS
    
```

6865	007160	000310				200.			:# OF XFER'S
6866	007162	104007				ERROR	7		:RETURN HERE IF STATUS ER - EXPECTED MAINT,
6867									:CYCLE, READY & IE
6868	007164	000412				BR	4\$:GO RESTORE VECTOR
6869	007166	104010				ERROR	10		:RETURN HERE IF WC ER - SHOULD = 0
6870	007170	000410				BR	4\$:GO RESTORE VECTOR
6871	007172	104011				ERROR	11		:RETURN HERE IF BAR ER - SHOULD = DBUFP+620
6872	007174	000406				BR	4\$:GO RESTORE VECTOR
6873	007176	004537	010526			JSR	R5,CKDAT1		:RETURN HERE IF OK - NOW GO CK DATA
6874	007202	000310				200.			:# OF XFER'S TO CK
6875	007204	104013				ERROR	13		:RETURN HERE IF DATA ER
6876	007206	000401				BR	4\$:GO RESTORE VECTOR
6877	007210	000704				BR	1\$:TRY NEXT BANK
6878	007212	012737	015316	001542	4\$:	MOV	#DBUF,DBUFP		:RESTORE BUFFER ADRS TO LOWEST 4K
6879	007220	004737	010132			JSR	PC,RSTVEC		:GO RESTORE VECTOR
6880									*****
(3)									:*TEST 31 TEST THE ADDRESS (I/O) ABILITY TO THE TTY PRINTER CSR
(3)									*****
(2)	007224	000004				TST31:	SCOPE		
(2)									
6881	007226	105777	171706			TSTB	@SWR		:KDF11-B?
6882	007232	100507				BMI	ALTERN		:IF YES,GO TO THE NEXT SUBTEST
6883	007234	004537	010112		101\$:	JSR	R5,SETVEC		:GO SET UP INTR RETURN
6884	007240	007344				1\$:RETURN TO 1\$ ON INTR
6885	007242	012777	177777	172250		MOV	#-1,@DRVWCR		:SET UP WC - 1 XFER
6886	007250	013737	001150	001542		MOV	\$TPS,DBUFP		:SET UP BUFFER ADRS TO PRINTER CSR ADRS
6887	007256	013777	001150	172236		MOV	\$TPS,@DRVBAR		:SET UP BUFFER ADRS - FROM PRINTER CSR
6888	007264	106427	000000			MTPS	#0		:ALLOW INTR
6889	007270	005077	172232			CLR	@DRVDBR		:ZERO THE DBR
6890	007274	012777	000161	172222		MOV	#161,@DRVCSR		:SET IE, XAD17 & XAD16, & GO
6891	007302	052777	000400	172214		BIS	#400,@DRVCSR		:SET CYCLE
6892	007310	013737	001524	001122		MOV	DRVCSR,\$BDADR		:SET UP CSR ADRS
6893	007316	012737	000760	001124		MOV	#760,\$GDDAT		:LD EXPECTED STATUS
6894	007324	017737	172174	001126		MOV	@DRVCSR,\$BDDAT		:READ THE CSR
6895	007332	042777	000100	172164		BIC	#100,@DRVCSR		:DISABLE IE
6896	007340	104005				ERROR	5		:NO INTR ON 1 WD XFER FROM XCSR
6897	007342	000434				BR	2\$:GO RESTORE VECTOR
6898	007344	022626			1\$:	CMP	(SP)+,(SP)+		:INTR RETURNS HERE - FIX STK SINCE NO RTI
6899	007346	004537	010156			JSR	R5,CKSTAT		:GO CK STATUS
6900	007352	000760				760			:CSR EXPECTED STATUS
6901	007354	000001				1			:# OF XFER'S
6902	007356	104007				ERROR	7		:RETURN HERE IF STATUS ER - EXPECTED CYCLE,
6903									:XAD17 & XAD16, RDY & IE
6904	007360	000425				BR	2\$:GO RESTORE VECTOR
6905	007362	104010				ERROR	10		:RETURN HERE IF WC ER - EXPECTED 0
6906	007364	000423				BR	2\$:GO RESTORE VECTOR
6907	007366	104011				ERROR	11		:RETURN HERE IF BAR ER - SHOULD = XCSR+2
6908	007370	000421				BR	2\$:GO RESTORE VECTOR
6909	007372	017737	171552	001124		MOV	@\$TPS,\$GDDAT		:RETURN HERE IF OK - GET XCSR CONTENTS
6910	007400	017737	172122	001126		MOV	@DRVDBR,\$BDDAT		:READ DBR
6911	007406	023737	001124	001126		CMP	\$GDDAT,\$BDDAT		:CORRECT?
6912	007414	001407				BEQ	2\$:BR IF SO
6913	007416	013737	001526	001122		MOV	DRVDBR,\$BDADR		:SET UP DBR ADRS
6914	007424	013737	001150	001120		MOV	\$TPS,\$GDADR		:GET ADRS OF DATA (XCSR)
6915	007432	104020				ERROR	20		:DATA ER FROM TTY PRINTER CSR
6916	007434	012737	015316	001542	2\$:	MOV	#DBUF,DBUFP		:RESTORE BUFFER ADRS TO LOWEST 4K

```

6917 007442 004737 010132      JSR    PC,RSTVEC      ;GO RESTORE VECTOR
6918 007446 000137 007672      JMP    NXDEV          ;IGNORE NEXT SUBTEST
6919
6920
6921
6922
6923 007452 005737 001544      ALTERN: TST    IOPAGE      ;I/O TRANSFER DESIRED?
6924 007456 001505                BEQ    NXDEV          ;IF NOT, SKIP THE TEST
6925 007460 004537 010112      JSR    R5,SETVEC     ;GO SET UP INTR RETURN
6926 007464 007570                1$
6927 007466 012777 177777 172024    MOV    #-1,@DRVWCR    ;SET UP WC - 1 XFER
6928 007474 013737 001544 001542    MOV    IOPAGE,DBUFP   ;SET UP BUFFER ADRS TO I/O CSR ADRS
6929 007502 013777 001544 172012    MOV    IOPAGE,@DRVBAR ;SET UP BUFFER ADRS - FROM I/O CSR
6930 007510 106427 000000                MTPS   #0            ;ALLOW INTR
6931 007514 005077 172006                CLR    @DRVDBR       ;ZERO THE DBR
6932 007520 012777 000161 171776    MOV    #161,@DRVCSR   ;SET IE, XAD17 & XAD16, & GO
6933 007526 052777 000400 171770    BIS    #400,@DRVCSR   ;SET CYCLE
6934 007534 013737 001524 001122    MOV    DRVCSR,$BDADR  ;SET UP CSR ADRS
6935 007542 012737 000760 001124    MOV    #760,$GDDAT   ;LD EXPECTED STATUS
6936 007550 017737 171750 001126    MOV    @DRVCSR,$BDDAT ;READ THE CSR
6937 007556 042777 000100 171740    BIC    #100,@DRVCSR   ;DISABLE IE
6938 007564 104005                ERROR  5            ;NO INTR ON 1 WD XFER FROM XCSR
6939 007566 000434                BR     2$           ;GO RESTORE VECTOR
6940 007570 022626                1$:  CMP    (SP)+,(SP)+   ;INTR RETURNS HERE - FIX STK SINCE NO RTI
6941 007572 004537 010156      JSR    R5,CKSTAT     ;GO CK STATUS
6942 007576 000760                760
6943 007600 000001                1
6944 007602 104007                ERROR  7            ;RETURN HERE IF STATUS ER - EXPECTED CYCLE,
6945
6946 007604 000425                BR     2$           ;XAD17 & XAD16, RDY & IE
6947 007606 104010                ERROR  10          ;GO RESTORE VECTOR
6948 007610 000423                BR     2$           ;RETURN HERE IF WC ER - EXPECTED 0
6949 007612 104011                ERROR  11          ;GO RESTORE VECTOR
6950 007614 000421                BR     2$           ;RETURN HERE IF BAR ER - SHOULD = XCSR+2
6951 007616 017737 171722 001124    MOV    @IOPAGE,$GDDAT ;GO RESTORE VECTOR
6952 007624 017737 171676 001126    MOV    @DRVDBR,$BDDAT ;RETURN HERE IF OK - GET XCSR CONTENTS
6953 007632 023737 001124 001126    CMP    $GDDAT,$BDDAT ;READ DBR
6954 007640 001407                BEQ    2$           ;CORRECT?
6955 007642 013737 001526 001122    MOV    DRVDBR,$BDADR  ;BR IF SO
6956 007650 013737 001544 001120    MOV    IOPAGE,$GDADR  ;SET UP DBR ADRS
6957 007656 104020                ERROR  20          ;GET ADRS OF DATA (XCSR)
6958 007660 012737 015316 001542 2$:  MOV    #DBUF,DBUFP   ;DATA ER FROM TTY PRINTER CSR
6959 007666 004737 010132      JSR    PC,RSTVEC     ;RESTORE BUFFER ADRS TO LOWEST 4K
6960
6961
6962
6963
6964 007672 000004                NXDEV: SCOPE
6965 007674 000241                NXDEV1: CLC
6966 007676 006037 001536      ROR    DMAP          ;CLR CARRY
6967 007702 001432                BEQ    $EOP         ;LOOK FOR NEXT
6968 007704 062737 000010 001520    ADD    #10,DRVWCR    ;BR IF ALL TESTED
6969 007712 062737 000010 001522    ADD    #10,DRVBAR    ;OFFSET BASE BUS ADRS TO NEXT DRV11B
6970 007720 062737 000010 001524    ADD    #10,DRVCSR
6971 007726 062737 000010 001526    ADD    #10,DRVDBR
6972 007734 062737 000004 001530    ADD    #4,DRVCTO    ;OFFSET VECTOR ADRS TO NEXT

```

6973 007742 062737 000004 001532
 6974 007750 005237 001206
 6975 007754 032737 000001 001536
 6976 007762 001744
 6977 007764 000137 002406
 6978
 (1)
 (2)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1) 007770
 (2) 007770 000240
 (1) 007772 005037 001102
 (1) 007776 005037 001160
 (1) 010002 005237 001202
 (1) 010006 042737 100000 001202
 (1) 010014 005327
 (1) 010016 000001
 (1) 010020 003022
 (1) 010022 012737
 (1) 010024 000001
 (1) 010026 010016
 (1) 010030 104401 010075
 (2) 010034 013746 001202
 (2) 010040 104405
 (1) 010042 104401 010072
 (1) 010046 013700 000042
 (1) 010052 001405
 (1) 010054 000005
 (1) 010056 004710
 (1) 010060 000240
 (1) 010062 000240
 (1) 010064 000240
 (1) 010066
 (1) 010066 000137
 (1) 010070 002012
 (1)
 (1)
 (1) 010072 377 377 000
 (1) 010075 015 042412 042116
 (1) 010102 050040 051501 020123
 (1) 010110 000043

```

ADD #4,DRVCT2 ;
INC $UNIT ;COUNT DEVICE
BIT #1,DMAP ;IS IT SELECTED?
BEQ NXDEV1 ;BR IF NOT
JMP RESTRT ;TEST NEXT
.SBTTL END OF PASS ROUTINE

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

$EOP:
NOP
CLR $STNM ;:ZERO THE TEST NUMBER
CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
INC $PASS ;:INCREMENT THE PASS NUMBER
BIC #10000,$PASS ;:DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;:LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ;:YES
MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
$ENDCT: .WORD 1
$EOPCT
TYPE $SENDMG ;:TYPE 'END PASS #'
MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $ENULL ;:TYPE A NULL CHARACTER
$GET42: MOV @#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 @ (PC)+ ;:RETURN
$RTNAD: .WORD START1

$ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
$ENDMG: .ASCIIZ <15><12>/END PASS #/
  
```


.SBTTL PROGRAM SUBROUTINES

```

6980
6981
6982
6983
6984
6985
6986
6987
6988 010112 106427 000200
6989 010116 012577 171406
6990 010122 012777 000200 171402
6991 010130 000205
6992
6993
6994
6995
6996
6997 010132 005077 171366
6998 010136 013777 001532 171364
6999 010144 005077 171362
7000 010150 106427 000200
7001 010154 000207
7002
7003
7004
7005
7006
7007
7008
7009 010156 017737 171342 001126
7010 010164 042777 000100 171332
7011 010172 012537 001124
7012 010176 023737 001124 001126
7013 010204 001406
7014 010206 013737 001524 001122
7015 010214 062705 000002
7016 010220 000205
7017 010222 017737 171272 001126 1$:
7018 010230 001410
7019 010232 013737 001520 001122
7020 010240 005037 001124
7021 010244 062705 000006
7022 010250 000205
7023 010252 011537 001124 2$:
7024 010256 006357 001124
7025 010262 063737 001542 001124
7026 010270 017737 171226 001126
7027 010276 042737 000001 001126
7028 010304 023737 001124 001126
7029 010312 001406
7030 010314 013737 001522 001122
7031 010322 062705 000012
7032 010326 000205
7033 010330 062705 000016 3$:
7034 010334 000205
7035
  
```

```

:*****
:THIS ROUTINE SETS THE PRIORITY LEVEL FOR NO INTERRUPT -
:SETS UP THE DRV11B INTERRUPT TO RETURN ON INTERRUPT
:TO THE ADDRESS INDICATED ((R5)) BY THE CALL +2
:*****
  
```

```

SETVEC: MTPS #200 ;SET UP FOR NO INTERRUPT
        MOV (R5)+,@DRVCT0 ;SET UP INTR RETURN ADRS
        MOV #200,@DRVCT2 ;KEEP PRIORITY LEVEL AT TOP ON INTR
        RTS R5 ;EXIT
  
```

```

:*****
:THIS ROUTINE CLEARS THE DRV11B CSR - RESTORES THE DRV11B
:INTERRUPT VECTOR TO A HALT - RAISES PRIORITY LEVEL
:*****
  
```

```

RSTVEC: CLR @DRVCSR ;CLR STATUS & CONTROL
        MOV DRVCT2,@DRVCT0 ;POINT VECTOR TO HALT
        CLR @DRVCT2 ;SET UP HALT
        MTPS #200 ;RAISE PRIORITY LEVEL
        RTS PC ;EXIT
  
```

```

:*****
:THIS ROUTINE CHECKS ON ALL DATA TRANSFERS FOR: CORRECT STATUS,
:CORRECT WORD COUNT & CORRECT BUFFER ADDRESS - THE EXPECTED DATA IS
:SUPPLIED IN THE CALL +2 & +4 - THE RETURN IS TO +22 IF NO ERRORS
:DETECTED - IF AN ERROR IS DETECTED THE RETURN IS TO THE APPROPRIATE ERROR EMT
:*****
  
```

```

CKSTAT: MOV @DRVCSR,$BDDAT ;READ THE STATUS
        BIC #100,@DRVCSR ;DISABLE THE IE BIT
        MOV (R5)+,$GDDAT ;SET UP EXPECTED STATUS
        CMP $GDDAT,$BDDAT ;CORRECT?
        BEQ 1$ ;BR IF SO
        MOV DRVCSR,$BDADR ;SET UP CSR ADRS
        ADD #2,R5 ;POINT TO THE CSR ER
        RTS R5 ;EXIT HERE ON STATUS ERROR
1$: MOV @DRVWCR,$BDDAT ;GET WC
    BEQ 2$ ;BR IF ZERO
    MOV DRVWCR,$BDADR ;SET UP WCR ADRS
    CLR $GDDAT ;EXPECTED 0
    ADD #6,R5 ;POINT TO THE WCR ER
    RTS R5 ;EXIT HERE ON WCR ER
2$: MOV (R5),$GDDAT ;GET XFER #
    ASL $GDDAT ;CONVERT TO WORD
    ADD DBUFP,$GDDAT ;POINT TO LAST XFER +2
    MOV @DRVBAR,$BDDAT ;GET BA
    BIC #BIT00,$BDDAT ;DON'T WANT BIT00
    CMP $GDDAT,$BDDAT ;CORRECT?
    BEQ 3$ ;BR IF SO
    MOV DRVBAR,$BDADR ;SET UP BAR ADRS
    ADD #12,R5 ;POINT TO BAR ER
    RTS R5 ;EXIT HERE ON BAR ER
3$: ADD #16,R5 ;ALL OK - POINT TO GOOD EXIT
    RTS R5 ;EXIT HERE IF NO ERRORS
  
```

7036
7037
7038
7039
7040
7041
7042 010336 012703 015316

```
*****  
: THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN  
: FOR 199 LOCATIONS - THE LAST LOCATION IS LOADED WITH THE  
: #70707 WHICH SHOULD BE THE DATA WORD AVAILABLE IN THE DBR  
: AT THE COMPLETION OF A 200 WORD TRANSFER  
: *****  
LDBUF: MOV #DBUF,R3 ;GET BUFFER ADRS
```

```

7044 010342 012704 177776 1$: MOV #177776,R4 ;SET UP FLOATING ZERO PATRN
7045 010346 010423 2$: MOV R4,(R3)+ ;LOAD IT (FLOATING 0)
7046 010350 005104 COM R4 ;MAKE INTO FLOATING 1
7047 010352 022703 016134 CMP #DBUF+616,R3 ;AT END OF BUFFER?
7048 010356 001003 BNE 4$ ;BR IF NOT
7049 010360 012713 070707 3$: MOV #70707,(R3) ;LOAD LAST DATVAR (SPECIAL)
7050 010364 000207 RTS PC ;GET OUT
7051 010366 010423 4$: MOV R4,(R3)+ ;LOAD IT (FLOATING 1)
7052 010370 022703 016134 CMP #DBUF+616,R3 ;AT END OF BUFFER?
7053 010374 001771 BEQ 3$ ;BR IF SO
7054 010376 005104 COM R4 ;BACK TO FLOATING ZERO
7055 010400 006304 ASL R4 ;SHIFT LEFT
7056 010402 005204 INC R4 ;KEEP LSB SET
7057 010404 103356 BCC 1$ ;GO RESET FLOATING PATRN
7058 010406 000757 BR 2$ ;GO LOAD NEXT PATRN
7059
7060
7061
7062
7063
7064
7065

```

```

:*****
:THIS ROUTINE LOADS 'DBUF' WITH A UNIQUE FLOATING ZERO/ONE PATTERN
:(177776,0,1,0,177775,0,2,0,177773,0,4,0,177767,0,10,0 ETC.)
:IT IS USED WITH MAINT BIT SET (DATI/DATO SEQUENCE) - 200 LOCS
:ARE LOADED WITH THIS PATTERN
:*****

```

```

7066 010410 013703 001542 LDBUF1: MOV DBUF,R3 ;GET BUFFER ADRS
7067 010414 010305 MOV R3,R5 ;SAVE IN R5
7068 010416 062705 000620 ADD #620,R5 ;POINT TO END OF BUFFER
7069 010422 012704 177776 1$: MOV #177776,R4 ;SET UP FLOATING ZERO PATRN
7070 010426 010423 2$: MOV R4,(R3)+ ;LOAD IT (FLOATING 0)
7071 010430 005023 CLR (R3)+ ;ZERO NEXT
7072 010432 005104 COM R4 ;SET UP FLOATING 1
7073 010434 010423 MOV R4,(R3)+ ;LOAD IT
7074 010436 005023 CLR (R3)+ ;ZERO NEXT
7075 010440 020503 CMP R5,R3 ;200 LOCS DONE?
7076 010442 001001 BNE 3$ ;BR IF NOT
7077 010444 000207 RTS PC ;GET OUT
7078 010446 005104 3$: COM R4 ;BACK TO FLOATING ZERO
7079 010450 006304 ASL R4 ;SHIFT LEFT
7080 010452 005204 INC R4 ;KEEP LSB SET
7081 010454 103362 BCC 1$ ;GO RESET FLOATING PATRN
7082 010456 000763 BR 2$ ;GO FLOAT NEXT PATRN
7083
7084
7085
7086
7087
7088

```

```

:*****
:THIS ROUTINE CHECKS 200 LOCATIONS IN 'DBUF' FOR GOOD TRANSFERED
:DATA (#177377) ON 'DATO' TRANSFERS - IF AN ERROR IS DETECTED
:THE RETURN IS TO CALL +2 - IF NO ERROR THE RETURN IS TO CALL +4
:*****

```

```

7089 010460 012701 015316 CKDAT: MOV #DBUF,R1 ;GET BUFFER ADRS
7090 010464 022721 177377 1$: CMP #177377,(R1)+ ;DATA OK?
7091 010470 001410 BEQ 2$ ;BR IF SO
7092 010472 013737 001526 001122 MOV DRVDBR,$BDADR ;SET UP DBR ADRS
7093 010500 014137 001126 MOV -(R1),$BDDAT ;GET ACTUAL DATA XFERED
7094 010504 010137 001120 MOV R1,$GDADR ;GET MEMORY ADRS
7095 010510 000207 RTS PC ;RETURN TO ERROR
7096 010512 022701 016136 2$: CMP #DBUF+620,R1 ;AT END OF 'DBUF'?
7097 010516 001362 BNE 1$ ;BR IF MORE
7098 010520 062716 000002 ADD #2,(SP) ;ADJUST STACK FOR GOOD RETURN
7099 010524 000207 RTS PC ;GET OUT

```

7100
 7101
 7102
 7103
 7104
 7105
 7106
 7107
 7108 010526 012500
 7109 010530 013702 001542
 7110 010534 012701 177776
 7111 010540 005003
 7112 010542 020122
 7113 010544 001010
 7114 010546 020122
 7115 010550 001006
 7116 010552 162700 000002
 7117 010556 003015
 7118 010560 062705 000004
 7119 010564 000205
 7120 010566 014237 001126
 7121 010572 010237 001120
 7122 010576 010137 001124
 7123 010602 013737 001526 001122
 7124 010610 000205
 7125 010612 005101
 7126 010614 005103
 7127 010616 001351
 7128 010620 006301
 7129 010622 005201
 7130 010624 103343
 7131 010626 000745

```

:*****
: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
:*****
CKDAT1: MOV      (R5)+,R0      :GET # OF CHECKS
        MOV      DBUFP,R2    :GET BUFFER ADRS
1$:     MOV      #177776,R1   :SET UP FLOATING ZERO PATRN
        CLR      R3          :R3 SAYS WHEN TO SHIFT PATRN
2$:     CMP      R1,(R2)+     :DATA OK?
        BNE     3$          :BR IF NOT
        CMP      R1,(R2)+     :DATA WRITTEN OK?
        BNE     3$          :BR IF NOT
        SUB     #2,R0        :ACCOUNT FOR TWO ADRS'S
        BGT     4$          :BR IF MORE
        ADD     #4,R5        :ADJUST FOR GOOD RETURN
        RTS     R5          :EXIT
3$:     MOV      -(R2),%BDDAT  :GET BAD DATA
        MOV      R2,%GDADR    :GET MEM ADRS
        MOV      R1,%GDDAT    :LD EXPECTED DATA
        MOV      DRVDBR,%BDADR :SET UP DBR ADRS
        RTS     R5          :RETURN TO ERROR
4$:     COM     R1           :NOW EXPECT COMPLEMENT
        COM     R3           :TIME TO SHIFT?
        BNE     2$          :BR IF NOT
        ASL     R1           :SHIFT LEFT
        INC     R1           :KEEP LSB SET
        BCC     1$          :GO RESET FLOATING PATRN
        BR     2$          :DO NEXT
  
```

```

7133          .SBTTL SYSMAC ROUTINES
7134
7135          .SBTTL TYPE ROUTINE
(1)
(2)
(1)          ::*****
(1)          ::*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)          ::*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)          ::*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)          ::*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHAFACTERS REQUIRED.
(1)          ::*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)          ::*
(1)          ::*CALL:
(1)          ::*1) USING A TRAP INSTRUCTION
(1)          ::*          TYPE          ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)          ::*OR
(1)          ::*          TYPE
(1)          ::*          MESADR
(1)          ::*
(1)          $TYPE:  TSTB          $TFPLG          ;; IS THERE A TERMINAL?
(1)          BPL          1$          ;; BR IF YES
(1)          HALT          ;; HALT HERE IF NO TERMINAL
(1)          BR          3$          ;; LEAVE
(1)          1$:  MOV          RO,-(SP)          ;; SAVE RO
(1)          MOV          @2(SP),RO          ;; GET ADDRESS OF ASCIZ STRING
(1)          CMPB          #APTENV,$ENV          ;; RUNNING IN APT MODE
(1)          BNE          62$          ;; NO,GO CHECK FOR APT CONSOLE
(1)          BITB          #APTSPOOL,$ENVM          ;; SPOOL MESSAGE TO APT
(1)          BEQ          62$          ;; NO,GO CHECK FOR CONSOLE
(1)          MOV          RO,61$          ;; SETUP MESSAGE ADDRESS FOR APT
(1)          JSR          PC,$ATY3          ;; SPOOL MESSAGE TO APT
(1)          61$:  .WORD          0          ;; MESSAGE ADDRESS
(1)          62$:  BITB          #APTCSUP,$ENVM          ;; APT CONSOLE SUPPRESSED
(1)          BNE          60$          ;; YES,SKIP TYPE OUT
(1)          2$:  MOVB          (RO)+,-(SP)          ;; PUSH CHARACTER TO BE TYPED ONTO STACK
(1)          BNE          4$          ;; BR IF IT ISN'T THE TERMINATOR
(1)          TST          (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
(1)          60$:  MOV          (SP)+,RO          ;; RESTORE RO
(1)          3$:  ADD          #2,(SP)          ;; ADJUST RETURN PC
(1)          RTI          ;; RETURN
(1)          4$:  CMPB          #HT,(SP)          ;; BRANCH IF <HT>
(1)          BEQ          8$          ;; BRANCH IF NOT <CRLF>
(1)          CMPB          #CRLF,(SP)          ;; BRANCH IF NOT <CRLF>
(1)          BNE          5$          ;; POP <CR><LF> EQUIV
(1)          TST          (SP)+          ;; TYPE A CR AND LF
(1)          TYPE          ;; TYPE A CR AND LF
(1)          $CRLF          ;;
(1)          CLRB          $CHARCNT          ;; CLEAR CHARACTER COUNT
(1)          BR          2$          ;; GET NEXT CHARACTER
(1)          5$:  JSR          PC,$TYPEC          ;; GO TYPE THIS CHARACTER
(1)          6$:  CMPB          $FILLC,(SP)+          ;; IS IT TIME FOR FILLER CHARS.?
(1)          BNE          2$          ;; IF NO GO GET NEXT CHAR.
(1)          MOV          $NULL,-(SP)          ;; GET # OF FILLER CHARS. NEEDED
(1)          ;; AND THE NULL CHAR.
(1)          7$:  DECB          1(SP)          ;; DOES A NULL NEED TO BE TYPED?
(1)          BLT          6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
    
```

```

(1) 011004 004737 011042      JSR    PC,$TYPEC      ;;GO TYPE A NULL
(1) 011010 105337 011106      DECB   $CHARCNT      ;;DO NOT COUNT AS A COUNT
(1) 011014 000770              BR     7$             ;;LOOP
(1)
(1)
(1)      ;HORIZONTAL TAB PROCESSOR
(1) 011016 112716 000040      8$:   MOVB   #' (SP)      ;;REPLACE TAB WITH SPACE
(1) 011022 004737 011042      9$:   JSR    PC,$TYPEC      ;;TYPE A SPACE
(1) 011026 132737 000007 011106  BITB   #7,$CHARCNT    ;;BRANCH IF NOT AT
(1) 011034 001372              BNE    9$             ;;TAB STOP
(1) 011036 005726              TST    (SP)+          ;;POP SPACE OFF STACK
(1) 011040 000724              BR     2$             ;;GET NEXT CHARACTER
(1) 011042 105777 170102      $TYPEC: TSTB  @STPS      ;;WAIT UNTIL PRINTER IS READY
(1) 011046 100375              BPL    $TYPEC
(1) 011050 116677 000002 170074  MOVB   2(SP),@STPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)
(1) 011056 122766 000015 000002  CMPB   #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
(1) 011064 001003              BNE    1$             ;;BRANCH IF NO
(1) 011066 105037 011106      CLRB   $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
(1) 011072 000406              BR     $TYPEX        ;;EXIT
(1) 011074 122766 000012 000002  1$:   CMPB   #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
(1) 011102 001402              BEQ    $TYPEX        ;;BRANCH IF YES
(1) 011104 105227              INCB   (PC)+          ;;COUNT THE CHARACTER
(1) 011106 000000      $CHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE
(1) 011110 000207      $TYPEX: RTS    PC
(1)
(1)
(1)

```

7136

.SBTTL APT COMMUNICATIONS ROUTINE

```

(1)
(2)
(1) 011112 112737 000001 011356  $ATY1: MOVB   #1,$FFLG      ;;TO REPORT FATAL ERROR
(1) 011120 112737 000001 011354  $ATY3: MOVB   #1,$MFLG      ;;TO TYPE A MESSAGE
(1) 011126 000403              BR     $ATYC
(1) 011130 112737 000001 011356  $ATY4: MOVB   #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
(1) 011136 010046      $ATYC:
(3) 011136 010046      MOV    R0,-(SP)        ;;PUSH R0 ON STACK
(3) 011140 010146      MOV    R1,-(SP)        ;;PUSH R1 ON STACK
(1) 011142 105737 011354      TSTB   $MFLG          ;;SHOULD TYPE A MESSAGE?
(1) 011146 001450              BEQ    5$             ;;IF NOT: BR
(1) 011150 122737 000001 001214  CMPB   #APTENV,$ENV    ;;OPERATING UNDER APT?
(1) 011156 001031              BNE    3$             ;;IF NOT: BR
(1) 011160 132737 000100 001215  BITB   #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 011166 001425              BEQ    3$             ;;IF NOT: BR
(1) 011170 017600 000004              MOV    @4(SP),R0      ;;GET MESSAGE ADDR.
(1) 011174 062766 000002 000004  ADD    #2,4(SP)        ;;BUMP RETURN ADDR.
(1) 011202 005737 001174      1$:   TST    $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
(1) 011206 001375              BNE    1$             ;;IF NOT: WAIT
(1) 011210 010037 001210      MOV    R0,$MSGAD
(1)      ;;PUT ADDR IN MAILBOX
(1) 011214 105720      2$:   TSTB   (R0)+          ;;FIND END OF MESSAGE
(1) 011216 001376              BNE    2$             ;;SUB START OF MESSAGE
(1) 011220 163700 001210      SUB    $MSGAD,R0      ;;GET MESSAGE LNGTH IN WORDS
(1) 011224 006200              ASR    R0              ;;PUT LENGTH IN MAILBOX
(1) 011226 010037 001212      MOV    R0,$MSGGLGT
(1) 011232 012737 000004 001174  MOV    #4,$MSGTYPE    ;;TELL APT TO TAKE MSG.

```



```

(1) 011360 017646 000000          $TYPOS: MOV      @ (SP), -(SP)      :: PICKUP THE MODE
(1) 011364 116637 000001 011603  MOVB     1 (SP), $OFILL  :: LOAD ZERO FILL SWITCH
(1) 011372 112637 011605          MOVB     (SP)+, $OMODE+1  :: NUMBER OF DIGITS TO TYPE
(1) 011376 062716 000002          ADD      #2, (SP)        :: ADJUST RETURN ADDRESS
(1) 011402 000406                    BR      $TYPON
(1) 011404 112737 000001 011603  $TYPOC: MOVB     #1, $OFILL  :: SET THE ZERO FILL SWITCH
(1) 011412 112737 000006 011605  MOVB     #6, $OMODE+1    :: SET FOR SIX(6) DIGITS
(1) 011420 112737 000005 011602  $TYPON: MOVB     #5, $OCNT  :: SET THE ITERATION COUNT
(1) 011426 010346                    MOV      R3, -(SP)      :: SAVE R3
(1) 011430 010446                    MOV      R4, -(SP)      :: SAVE R4
(1) 011432 010546                    MOV      R5, -(SP)      :: SAVE R5
(1) 011434 113704 011605          MOVB     $OMODE+1, R4   :: GET THE NUMBER OF DIGITS TO TYPE
(1) 011440 005404                    NEG      R4
(1) 011442 062704 000006          ADD      #6, R4         :: SUBTRACT IT FOR MAX. ALLOWED
(1) 011446 110437 011604          MOVB     R4, $OMODE     :: SAVE IT FOR USE
(1) 011452 113704 011603          MOVB     $OFILL, R4     :: GET THE ZERO FILL SWITCH
(1) 011456 016605 000012          MOV      12 (SP), R5   :: PICKUP THE INPUT NUMBER
(1) 011462 005003                    CLR      R3             :: CLEAR THE OUTPUT WORD
(1) 011464 006105                    1$:     ROL      R5         :: ROTATE MSB INTO 'C'
(1) 011466 000404                    BR      3$
(1) 011470 006105                    2$:     ROL      R5         :: FORM THIS DIGIT
(1) 011472 006105                    ROL      R5
(1) 011474 006105                    ROL      R5
(1) 011476 010503                    MOV      R5, R3
(1) 011500 006103                    3$:     ROL      R3         :: GET LSB OF THIS DIGIT
(1) 011502 105337 011604          DECB     $OMODE         :: TYPE THIS DIGIT?
(1) 011506 100016                    BPL     7$             :: BR IF NO
(1) 011510 042703 177770          BIC     #177770, R3    :: GET RID OF JUNK
(1) 011514 001002                    BNE     4$             :: TEST FOR 0
(1) 011516 005704                    TST     R4             :: SUPPRESS THIS 0?
(1) 011520 001403                    BEQ     5$             :: BR IF YES
(1) 011522 005204                    4$:     INC      R4         :: DON'T SUPPRESS ANYMORE 0'S
(1) 011524 052703 000060          BIS     #'0, R3        :: MAKE THIS DIGIT ASCII
(1) 011530 052703 000040          5$:     BIS     #' , R3    :: MAKE ASCII IF NOT ALREADY
(1) 011534 110337 011600          MOVB     R3, 8$        :: SAVE FOR TYPING
(1) 011540 104401 011600          TYPE     , 8$         :: GO TYPE THIS DIGIT
(1) 011544 105337 011602          7$:     DECB     $OCNT    :: COUNT BY 1
(1) 011550 C03347                    BGT     2$             :: BR IF MORE TO DO
(1) 011552 002402                    BLT     6$             :: BR IF DONE
(1) 011554 005204                    INC     R4             :: INSURE LAST DIGIT ISN'T A BLANK
(1) 011556 000744                    BR      2$             :: GO DO THE LAST DIGIT
(1) 011560 012605                    6$:     MOV      (SP)+, R5  :: RESTORE R5
(1) 011562 012604                    MOV      (SP)+, R4     :: RESTORE R4
(1) 011564 012603                    MOV      (SP)+, R3     :: RESTORE R3
(1) 011566 016666 000002 000004  MOV      2 (SP), 4 (SP) :: SET THE STACK FOR RETURNING
(1) 011574 012616                    MOV      (SP)+, (SP)
(1) 011576 000002                    RTI
(1) 011600 000                    8$:     .BYTE    0         :: RETURN
(1) 011601 000                    .BYTE    0         :: STORAGE FOR ASCII DIGIT
(1) 011602 000                    $OCNT:  .BYTE    0         :: TERMINATOR FOR TYPE ROUTINE
(1) 011603 000                    $OFILL: .BYTE    0         :: OCTAL DIGIT COUNTER
(1) 011604 000000                    $OMODE: .WORD    0         :: ZERO FILL SWITCH
(1) 011604 000000                    .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

7138
 (1)
 (2)
 (1)

 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT

```

(1) ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1) ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1) ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1) ;*REPLACED WITH SPACES.
(1) ;*CALL:
(1) ;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
(1) ;*      TYPDS      ;;GO TO THE ROUTINE
(1)
(1) $TYPDS:
(3) 011606 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 011610 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 011612 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(3) 011614 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
(3) 011616 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
(1) 011620 012746 020200      MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
(1) 011624 016605 000020      MOV      20(SP),R5    ;;GET THE INPUT NUMBER
(1) 011630 100004      BPI      1$          ;;BR IF INPUT IS POS.
(1) 011632 005405      NEG      R5          ;;MAKE THE BINARY NUMBER POS.
(1) 011634 112766 000055 000001      MOVB     #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
(1) 011642 005000      CLR      R0          ;;ZERO THE CONSTANTS INDEX
(1) 011644 012703 012022      MOV      #SDBLK,R3    ;;SETUP THE OUTPUT POINTER
(1) 011650 112723 000040      MOVB     #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
(1) 011654 005002      CLR      R2          ;;CLEAR THE BCD NUMBER
(1) 011656 016001 012012      MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
(1) 011662 160105      SUB      R1,R5        ;;FORM THIS BCD DIGIT
(1) 011664 002402      BLT     4$          ;;BR IF DONE
(1) 011666 005202      INC     R2          ;;INCREASE THE BCD DIGIT BY 1
(1) 011670 000774      BR      3$
(1) 011672 060105      ADD     R1,R5        ;;ADD BACK THE CONSTANT
(1) 011674 005702      TST     R2          ;;CHECK IF BCD DIGIT=0
(1) 011676 001002      BNE     5$          ;;FALL THROUGH IF 0
(1) 011700 105716      TSTB    (SP)        ;;STILL DOING LEADING 0'S?
(1) 011702 100407      BMI     7$          ;;BR IF YES
(1) 011704 106316      ASLB    (SP)        ;;MSD?
(1) 011706 103003      BCC     6$          ;;BR IF NO
(1) 011710 116663 000001 177777      MOVB     1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 011716 052702 000060      BIS     #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1) 011722 052702 000040      BIS     #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 011726 110223      MOVB     R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 011730 005720      TST     (R0)+       ;;JUST INCREMENTING
(1) 011732 020027 000010      CMP     R0,#10      ;;CHECK THE TABLE INDEX
(1) 011736 002746      BLT     2$          ;;GO DO THE NEXT DIGIT
(1) 011740 003002      BGT     8$          ;;GO TO EXIT
(1) 011742 010502      MOV     R5,R2       ;;GET THE LSD
(1) 011744 000764      BR     6$          ;;GO CHANGE TO ASCII
(1) 011746 105726      TSTB    (SP)+       ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 011750 100003      BPL     9$          ;;BR IF NO
(1) 011752 116663 177777 177776      MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 011760 105013      CLRB    (R3)        ;;SET THE TERMINATOR
(3) 011762 012605      MOV     (SP)+,R5    ;;POP STACK INTO R5
(3) 011764 012603      MOV     (SP)+,R3    ;;POP STACK INTO R3
(3) 011766 012602      MOV     (SP)+,R2    ;;POP STACK INTO R2
(3) 011770 012601      MOV     (SP)+,R1    ;;POP STACK INTO R1
(3) 011772 012600      MOV     (SP)+,R0    ;;POP STACK INTO R0
(1) 011774 104401 012022      TYPE    $SDBLK      ;;NOW TYPE THE NUMBER
(1) 012000 016666 000002 000004      MOV     2(SP),4(SP) ;;ADJUST THE STACK

```

(1) 012006 012616
 (1) 012010 000002
 (1) 012012 023420
 (1) 012014 001750
 (1) 012016 000144
 (1) 012020 000012
 (1) 012022 000004

```

MOV      (SP)+,(SP)
RTI
;;RETURN TO USER
$DTBL:   10000.
         1000.
         100.
         10.
$DBLK:   .BLKW 4
.$BTTL   ERROR HANDLER ROUTINE
    
```

7139

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO SWRCK ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW10=1      BELL ON ERROR
*SW09=1      LOOP ON ERROR
*CALL
*          ERROR  N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
    
```

(1) 012032
 (1) 012032 104407
 (2) 012034 104407
 (1) 012036 105237 001103
 (1) 012042 001775
 (1) 012044 013777 001102 167070
 (1) 012052 032777 002000 167060
 (1) 012060 001402
 (1) 012062 104401 001164
 (1) 012066 005237 001112
 (1) 012072 011637 001116
 (1) 012076 162737 000002 001116
 (1) 012104 117737 167006 001114
 (1) 012112 032777 020000 167020
 (1) 012120 001004
 (1) 012122 004737 012222
 (1) 012126 104401 001171
 (1) 012132
 (1) 012132 122737 000001 001214
 (1) 012140 001007
 (1) 012142 113737 001114 012154
 (1) 012150 004737 011130
 (1) 012154 000
 (1) 012155 000
 (1) 012156 000777
 (1) 012160 005777 166754
 (1) 012164 100002
 (1) 012166 000000
 (1) 012170 104407
 (1) 012172 032777 001000 166740
 (1) 012200 001402
 (1) 012202 013716 001110
 (1) 012206 005737 001162
 (1) 012212 001402
 (1) 012214 013716 001162

```

$error:
CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
CKSWR      ;GO LOOK FOR SWR CHANGE
7$: INCB    $ERFLG      ;;SET THE ERROR FLAG
        BEQ    7$        ;;DON'T LET THE FLAG GO TO ZERO
        MOV    $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
        BIT    #BIT10,@SWR  ;;BELL ON ERROR?
        BEQ    1$        ;;NO - SKIP
        TYPE   ,SBELL     ;;RING BELL
1$: INC     $ERTTL      ;;COUNT THE NUMBER OF ERRORS
        MOV    (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
        SUB    #2,$ERRPC
        MOVB  @ $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
        BIT    #BIT13,@SWR  ;;SKIP TYPEOUT IF SET
        BNE   20$        ;;SKIP TYPEOUTS
        JSR   PC,SWRCK    ;;GO TO USER ERROR ROUTINE
        TYPE  , $CRLF
20$: CMPB   #APTENV,$ENV  ;;RUNNING IN APT MODE
        BNE   2$        ;;NO,SKIP APT ERROR REPORT
        MOVB  $ITEMB,21$  ;;SET ITEM NUMBER AS ERROR NUMBER
        JSR   PC,$ATY4    ;;REPORT FATAL ERROR TO APT
21$: .BYTE  0
        .BYTE  0
22$: BR    22$          ;;APT ERROR LOOP
2$: TST   @SWR          ;;HALT ON ERROR
        BPL  3$        ;;SKIP IF CONTINUE
        HALT                ;;HALT ON ERROR!
3$: BIT    #BIT09,@SWR   ;;TEST FOR CHANGE IN SOFT-SWR
        BEQ    4$        ;;LOOP ON ERROR SWITCH SET?
        BR   IF NO
        MOV   $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
        TST   $ESCAPE    ;;CHECK FOR AN ESCAPE ADDRESS
        BEQ   5$        ;;BR IF NONE
        MOV   $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
5$:
    
```

```

(1) 012220 5$:
(1) 012220 000002 RTI ::RETURN
7140 ::*****
7141 :GO TYPE ERROR
7142 :GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
7143 ::*****
7144 012222 113737 001102 001534 SWRCK: MOV B $TSTNM,TSTNUM :SET UP TEST # ON ER
7145 012230 004737 012240 JSR PC,$ERRTYP :GO TYPE ERROR
7146 012234 104407 CKSWR :GO LOOK FOR SWR CHANGE
7147 012236 000207 RTS PC :RETURN TO ERROR HANDLER
7148 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)
(1) ::*****
(1) :*THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
(1) :*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
(1) :*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1) 012240 $ERRTYP:
(1) 012240 104401 001171 TYPE $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 012244 010046 MOV R0,-(SP) ::SAVE R0
(1) 012246 005000 CLR R0 ::PICKUP THE ITEM INDEX
(1) 012250 153700 001114 BISB @#$ITEMB,R0
(1) 012254 001004 BNE 1$ ::IF ITEM NUMBER IS ZERO, JUST
(1) (2) 012256 013746 001116 MOV $ERRPC,-(SP) ::TYPE THE PC OF THE ERROR
(2) (2) (2) 012262 104402 TYPOC ::SAVE $ERRPC FOR TYPEOUT
(1) 012264 000426 BR 6$ ::ERROR ADDRESS
(1) 012266 005300 1$: DEC R0 ::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 012270 006300 ASL R0 ::GET OUT
(1) 012272 006300 ASL R0 ::ADJUST THE INDEX SO THAT IT WILL
(1) 012274 006300 ASL R0 :: WORK FOR THE ERROR TABLE
(1) 012276 062700 001320 ADD #$ERRTB,R0 ::FORM TABLE POINTER
(1) 012302 012037 012312 MOV (R0)+,2$ ::PICKUP 'ERROR MESSAGE' POINTER
(1) 012306 001404 BEQ 3$ ::SKIP TYPEOUT IF NO POINTER
(1) 012310 104401 TYPE ::TYPE THE 'ERROR MESSAGE'
(1) 012312 000000 2$: .WORD 0 ::'ERROR MESSAGE' POINTER GOES HERE
(1) 012314 104401 001171 TYPE $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 012320 012037 012330 MOV (R0)+,4$ ::PICKUP 'DATA HEADER' POINTER
(1) 012324 001404 BEQ 5$ ::SKIP TYPEOUT IF 0
(1) 012326 104401 TYPE ::TYPE THE 'DATA HEADER'
(1) 012330 000000 4$: .WORD 0 ::'DATA HEADER' POINTER GOES HERE
(1) 012332 104401 001171 TYPE $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 012336 011000 5$: MOV (R0),R0 ::PICKUP 'DATA TABLE' POINTER
(1) 012340 001004 BNE 7$ ::GO TYPE THE DATA
(1) 012342 012600 6$: MOV (SP)+,R0 ::RESTORE R0
(1)
(1) 012344 104401 001171 TYPE $CRLF ::'CARRIAGE RETURN' & 'LINE FEED'
(1) 012350 000207 RTS PC ::RETURN
(1) 012352 7$:
(2) 012352 013046 MOV @ (R0)+,-(SP) ::SAVE @ (R0)+ FOR TYPEOUT
(2) 012354 104402 TYPOC ::GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 012356 005710 TST (R0) ::IS THERE ANOTHER NUMBER?
(1) 012360 001770 BEQ 6$ ::BR IF NO
(1) 012362 104401 012370 TYPE 8$ ::TYPE TWO(2) SPACES
(1) 012366 000771 BR 7$ ::LOOP

```

```

(1) 012370 020040 000 8$: .ASCIZ / / ;;TWO(2) SPACES
(1) 012374 .EVEN
7149 .SBTTL SCOPE HANDLER ROUTINESTARS
(1) ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1) ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) ;*SW14=1 LOOP ON TEST
(1) ;*SW11=1 INHIBIT ITERATIONS
(1) ;*SW09=1 LOOP ON ERROR
(1) ;*SW08=1 LOOP ON TEST IN SWR<5:0>
(1) ;*CALL
(1) ;* SCOPE ;;SCOPE=IOT
(1) $SCOPE:
(1) 012374 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 012376 104407 CKSWR ;;GO LOOK FOR SWR CHANGE
(2) 012376 104407 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
(1) 012400 032777 040000 166532 BNE $OVER ;;YES IF SW14=1
(1) 012406 001117 ;#####START OF CODE FOR THE XOR TESTER#####
(1) 012410 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
(1) 012412 013746 000004 MOV @#ERRVEC,-(SP) ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1) 012416 012737 012436 000004 MOV #5$,@#ERRVEC ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 012424 005737 177060 TST @#177060 ;;SET FOR TIMEOUT
(1) 012430 012637 000004 MOV (SP)+,@#ERRVEC ;;TIME OUT ON XOR?
(1) 012434 000466 BR $SVLAD ;;RESTORE THE ERROR VECTOR
(1) 012436 022626 5$: CMP (SP)+,(SP)+ ;;GO TO THE NEXT TEST
(1) 012440 012637 000004 MOV (SP)+,@#ERRVEC ;;CLEAR THE STACK AFTER A TIME OUT
(1) 012444 000426 BR 7$ ;;RESTORE THE ERROR VECTOR
(1) 012446 6$:;#####END OF CODE FOR THE XOR TESTER##### ;;LOOP ON THE PRESENT TEST
(1) 012446 032777 000400 166464 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
(1) 012454 001407 BEQ 2$ ;;BR IF NO
(1) 012456 017746 166456 MOV @SWR,-(SP) ;;SET DESIRED TEST NUM. FROM SWR
(1) 012462 042716 000300 BIC #$$SWRMK,(SP) ;;STRIP AWAY UNDESIRED BITS
(1) 012466 122637 001102 CMPB (SP)+,$TSTNM ;;ON THE RIGHT TEST?
(1) 012472 001465 BEQ $OVER ;;BR IF YES
(1) 012474 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
(1) 012500 001421 BEQ 3$ ;;BR IF NO
(1) 012502 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 012510 101015 BHI 3$ ;;BR IF NO
(1) 012512 032777 001000 166420 BIT #BIT09,@SWR ;;LOOP ON ERROR?
(1) 012520 001404 BEQ 4$ ;;BR IF NO
(1) 012522 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 012530 000446 BR $OVER
(1) 012532 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
(1) 012536 005037 001160 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 012542 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
(1) 012544 032777 004000 166366 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1) 012552 001011 BNE 1$ ;;BR IF YES
(1) 012554 005737 001202 TST $PASS ;;IF FIRST PASS OF PROGRAM
(1) 012560 001406 BEQ 1$ ;; INHIBIT ITERATIONS
(1) 012562 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 012566 023737 001160 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 012574 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 012576 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
    
```



```

(1)
(1) 013014 021627 000015      10$:  CMP      (SP),#15      ;; IS IT A <CR>?
(1) 013020 001022                BNE      16$          ;; BRANCH IF NO
(1) 013022 005766 000004      TST      4(SP)        ;; YES, IS IT THE FIRST CHAR?
(1) 013026 001403                BEQ      11$          ;; BRANCH IF YES
(1) 013030 016677 000002 166102  MOV      2(SP),@SWR    ;; SAVE NEW SWR
(1) 013036 062706 000006      11$:  ADD      #6,SP        ;; CLEAR UP STACK
(1) 013042 104401 001171      14$:  TYPE     ,SRLF      ;; ECHO <CR> AND <LF>
(1) 013046 123727 001135 000001  CMPB     $INTAG,#1    ;; RE-ENABLE TTY KBD INTERRUPTS?
(1) 013054 001003                BNE      15$          ;; BRANCH IF NOT
(1) 013056 012777 000100 166060  MOV      #100,@STKS   ;; RE-ENABLE TTY KBD INTERRUPTS
(1) 013064 000002                RTI                      ;; RETURN
(1) 013066 004737 011042      16$:  JSR      PC,$TYPEC    ;; ECHO CHAR
(1) 013072 021627 000060      CMP      (SP),#60     ;; CHAR < 0?
(1) 013076 002420                BLT      18$          ;; BRANCH IF YES
(1) 013100 021627 000067      CMP      (SP),#67     ;; CHAR > 7?
(1) 013104 003015                BGT      18$          ;; BRANCH IF YES
(1) 013106 042726 000060      BIC      #60,(SP)+    ;; STRIP-OFF ASCII
(1) 013112 005766 000002      TST      2(SP)        ;; IS THIS THE FIRST CHAR
(1) 013116 001403                BEQ      17$          ;; BRANCH IF YES
(1) 013120 006316                ASL      (SP)         ;; NO, SHIFT PRESENT
(1) 013122 006316                ASL      (SP)         ;; CHAR OVER TO MAKE
(1) 013124 006316                ASL      (SP)         ;; ROOM FOR NEW ONE.
(1) 013126 005266 000002      17$:  INC      2(SP)        ;; KEEP COUNT OF CHAR
(1) 013132 056616 177776      BIS      -2(SP),(SP)  ;; SET IN NEW CHAR
(1) 013136 000707                BR       7$           ;; GET THE NEXT ONE
(1) 013140 104401 001170      18$:  TYPE     ,SQUES     ;; TYPE ?<CR><LF>
(1) 013144 000720                BR       20$         ;; SIMULATE CONTROL-U
(1) .DSABL  LSB
(1)
(1)
(2)
(1) *****
(1) *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) *CALL:
(1) *      RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
(1) *      RETURN HERE   ;; CHARACTER IS ON THE STACK
(1) *
(1) *
(1) *
(1) *
(1) *
(1) *RDCHR: MOV      (SP),-(SP)  ;; PUSH DOWN THE PC
(1) 013146 011646 000004 000002  MOV      4(SP),2(SP)  ;; SAVE THE PS
(1) 013150 016666 000004 000002  1$:  TSTB     @STKS       ;; WAIT FOR
(1) 013156 105777 165762                BPL      1$           ;; A CHARACTER
(1) 013162 100375                MOVB     @STKB,4(SP)  ;; READ THE TTY
(1) 013164 117766 165756 000004  BIC      #^C<177>,4(SP) ;; GET RID OF JUNK IF ANY
(1) 013172 042766 177600 000004  CMP      4(SP),#23    ;; IS IT A CONTROL-S?
(1) 013200 026627 000004 000023  BNE      3$           ;; BRANCH IF NO
(1) 013206 001013                TSTB     @STKS       ;; WAIT FOR A CHARACTER
(1) 013210 105777 165730      2$:  BPL      2$           ;; LOOP UNTIL ITS THERE
(1) 013214 100375                MOVB     @STKB,-(SP)  ;; GET CHARACTER
(1) 013216 117746 165724  BIC      #^C177,(SP)  ;; MAKE IT 7-BIT ASCII
(1) 013222 042716 177600      CMP      (SP)+,#21    ;; IS IT A CONTROL-Q?
(1) 013226 022627 000021  BNE      2$           ;; IF NOT DISCARD IT
(1) 013232 001366                BR       1$           ;; YES, RESUME
(1) 013234 000750                CMP      4(SP),#140  ;; IS IT UPPER CASE?
(1) 013236 026627 000004 000140  3$:  BLT      4$           ;; BRANCH IF YES
(1) 013244 002407

```



```

(1) 013246 026627 000004 000175      CMP      4(SP),#175      ;; IS IT A SPECIAL CHAR?
(1) 013254 003003                    BGT      4$              ;; BRANCH IF YES
(1) 013256 042766 000040 000004      BIC      #40,4(SP)      ;; MAKE IT UPPER CASE
(1) 013264 000002                    4$: RTI                  ;; GO BACK TO USER
(2)                                     ;; *****
(1)                                     ;; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;; *CALL:
(1)                                     ;; *
(1)                                     ;; *   RDLIN                ;; INPUT A STRING FROM THE TTY
(1)                                     ;; *   RETURN HERE         ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;; *                       ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
(1) 013266 010346                    $RDLIN: MOV      R3,-(SP)  ;; SAVE R3
(1) 013270 012703 013374              1$:  MOV      #$TTYIN,R3  ;; GET ADDRESS
(1) 013274 022703 013404              2$:  CMP      #$TTYIN+8.,R3 ;; BUFFER FULL?
(1) 013300 101405                    BLOS     4$              ;; BR IF YES
(1) 013302 104410                    RDCHR                    ;; GO READ ONE CHARACTER FROM THE TTY
(1) 013304 112613                    MOVB     (SP)+,(R3)      ;; GET CHARACTER
(1) 013306 122713 000177              10$: CMPB     #177,(R3)    ;; IS IT A RUBOUT
(1) 013312 001003                    BNE     3$              ;; SKIP IF NOT
(1) 013314 104401 001170              4$:  TYPE     ,SQUES     ;; TYPE A '?'
(1) 013320 000763                    BR       1$             ;; CLEAR THE BUFFER AND LOOP
(1) 013322 111337 013372              3$:  MOVB     (R3),9$     ;; ECHO THE CHARACTER
(1) 013326 104401 013372              TYPE     ,9$
(1) 013332 122723 000015              CMPB     #15,(R3)+     ;; CHECK FOR RETURN
(1) 013336 001356                    BNE     2$             ;; LOOP IF NOT RETURN
(1) 013340 105063 177777              CLRB    -1(R3)         ;; CLEAR RETURN (THE 15)
(1) 013344 104401 001172              TYPE     ,LF           ;; TYPE A LINE FEED
(1) 013350 012603                    MOV      (SP)+,R3      ;; RESTORE R3
(1) 013352 011646                    MOV      (SP),-(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 013354 016666 000004 000002      MOV      4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
(1) 013362 012766 013374 000004      MOV      #$TTYIN,4(SP)
(1) 013370 000002                    RTI                    ;; RETURN
(1) 013372 000                    9$:  .BYTE    0          ;; STORAGE FOR ASCII CHAR. TO TYPE
(1) 013373 000                    .BYTE    0          ;; TERMINATOR
(1) 013374 000010                    $TTYIN: .BLKB   8.    ;; RESERVE 8 BYTES FOR TTY INPUT
(1) 013404 052536 005015 000          $CNTLU: .ASCIZ  /^U/<15><12> ;; CONTROL 'U'
(1) 013411 136 006507 000012        $CNTLG: .ASCIZ  /^G/<15><12> ;; CONTROL 'G'
(1) 013416 005015 053523 020122        $MSWR:  .ASCIZ  <15><12>/SWR = /
(1) 013424 020075 000
(1) 013427 040 047040 053505        $MNEW:  .ASCIZ  / NEW = /
(1) 013434 036440 000040
7151
(2)                                     ;; .SBTTL READ AN OCTAL NUMBER FROM THE TTY
(1)                                     ;; *****
(1)                                     ;; *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)                                     ;; *CHANGE IT TO BINARY.
(1)                                     ;; *THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
(1)                                     ;; *OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
(1)                                     ;; *FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
(1)                                     ;; *THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
(1)                                     ;; *CALL:
(1)                                     ;; *
(1)                                     ;; *   RDOCT                ;; READ AN OCTAL NUMBER
(1)                                     ;; *   RETURN HERE         ;; LOW ORDER BITS ARE ON TOP OF THE STACK
(1)                                     ;; *BITS ARE IN $HIOCT    ;; HIGH ORDER
(1) 013440 011646                    $RDOCT: MOV      (SP),-(SP) ;; PROVIDE SPACE FOR THE
(1) 013442 016666 000004 000002      MOV      4(SP),2(SP)   ;; INPUT NUMBER

```

```
(3) 013450 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3) 013452 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3) 013454 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(1) 013456 104411      1$: RDLIN      ;;READ AN ASCIZ LINE
(1) 013460 012600      MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
(1) 013462 010037 013566 MOV      R0,5$        ;;AND SAVE IT
(1) 013466 005001      CLR      R1           ;;CLEAR DATA WORD
(1) 013470 005002      CLR      R2
(1) 013472 112046      2$: MOV      (R0)+,-(SP) ;;PICKUP THIS CHARACTER
(1) 013474 001420      BEQ      3$          ;;IF ZERO GET OUT
(1) 013476 122716 000060 CMPB     #'0,(SP)     ;;MAKE SURE THIS CHARACTER
(1) 013502 003026      BGT      4$          ;;IS AN OCTAL DIGIT
(1) 013504 122716 000067 CMPB     #'7,(SP)
(1) 013510 002423      BLT      4$
(1) 013512 006301      ASL      R1           ;;*2
(1) 013514 006102      ROL      R2
(1) 013516 006301      ASL      R1           ;;*4
(1) 013520 006102      ROL      R2
(1) 013522 006301      ASL      R1           ;;*8
(1) 013524 006102      ROL      R2
(1) 013526 042716 177770 BIC      #'C7,(SP)   ;;STRIP THE ASCII JUNK
(1) 013532 062601      ADD      (SP)+,R1    ;;ADD IN THIS DIGIT
(1) 013534 000756      BR       2$          ;;LOOP
(1) 013536 005726      3$: TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
(1) 013540 010166 000012 MOV      R1,12(SP)   ;;SAVE THE RESULT
(1) 013544 010237 013576 MOV      R2,$HIOCT
(3) 013550 012602      MOV      (SP)+,R2    ;;POP STACK INTO R2
(3) 013552 012601      MOV      (SP)+,R1    ;;POP STACK INTO R1
(3) 013554 012600      MOV      (SP)+,R0    ;;POP STACK INTO R0
(1) 013556 000002      RTI
(1) 013560 005726      4$: TST      (SP)+      ;;CLEAN PARTIAL FROM STACK
(1) 013562 105010      CLR      (R0)        ;;SET A TERMINATOR
(1) 013564 104401      TYPE
(1) 013566 000000      5$: .WORD     0         ;;TYPE UP THRU THE BAD CHAR.
(1) 013570 104401 001170 TYPE     $QUES
(1) 013574 000730      BR       1$          ;;TRY AGAIN
(1) 013576 000000      $HIOCT: .WORD     0   ;;HIGH ORDER BITS GO HERE
(1) 013600
7152
(1)
(2)
(1)
(1) 013600 012737 013744 000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 013606 012737 000300 000026 MOV     #PR6,@#PWRVEC+2 ;;PRIO:6
(3) 013614 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
(3) 013616 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
(3) 013620 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
(3) 013622 010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
(3) 013624 010446      MOV     R4,-(SP)      ;;PUSH R4 ON STACK
(3) 013626 010546      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
(3) 013630 017746 165304 MOV     @SWR,-(SP)    ;;PUSH @SWR ON STACK
(1) 013634 010637 013750 MOV     SP,$SAVR6     ;;SAVE SP
(1) 013640 012737 013652 000024 MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 013646 000000      HALT
(1) 013650 000776      BR     .-2          ;;HANG UP
```

```

(2)
(1)
(1) 013652 012737 013744 000024 $PWRUP: MOV    #SILLUP,@PWRVEC ;;SET FOR FAST DOWN
(1) 013660 013706 013750          MOV    $SAVR6,SP      ;;GET SP
(1) 013664 005037 013750          CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
(1) 013670 005237 013750 1$:    INC    $SAVR6        ;;WAIT FOR THE INC
(1) 013674 001375          BNE   1$            ;;OF WORD
(3) 013676 012677 165236          MOV    (SP)+,@SWR    ;;POP STACK INTO @SWR
(3) 013702 012605          MOV    (SP)+,R5     ;;POP STACK INTO R5
(3) 013704 012604          MOV    (SP)+,R4     ;;POP STACK INTO R4
(3) 013706 012603          MOV    (SP)+,R3     ;;POP STACK INTO R3
(3) 013710 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
(3) 013712 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
(3) 013714 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
(1) 013716 012737 013600 000024 MOV    #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 013724 012737 000300 000026 MOV    #PR6,@PWRVEC+2 ;;PRIO:6
(1) 013732 104401          TYPE   PWRMSG        ;;REPORT T.IE POWER FAILURE
(1) 013734 013752 $PWRMG: .WORD PWRMSG   ;;POWER FAIL MESSAGE POINTER
(1) 013736 012716 $PWRAD: MOV    (PC)+,(SP) ;;RESTART AT RESTRT
(1) 013740 002406 $PWRAD: .WORD RESTRT  ;;RESTART ADDRESS
(1) 013742 000002          RTI
(1) 013744 000000 $SILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
(1) 013746 000776          BR    .-2           ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 013750 000000 $SAVR6: 0            ;;PUT THE SP HFRE
7153 013752 005015 042522 052123 PWRMSG: .ASCIZ <15><12>/RESTARTED FROM PWR FAIL/
      013760 051101 042524 020104
      013766 051106 046517 050040
      013774 051127 043040 044501
      014002 000114

7154 .EVEN
7155 .SBTTL TRAP DECODER
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 014004 010046 $TRAP: MOV    R0,-(SP)   ;;SAVE R0
(1) 014006 016600 000002 MOV    2(SP),R0      ;;GET TRAP ADDRESS
(1) 014012 005740 TST   -(R0)          ;;BACKUP BY 2
(1) 014014 111000 MOV   (R0),R0        ;;GET RIGHT BYTE OF TRAP
(1) 014016 006300 ASL   R0             ;;POSITION FOR INDEXING
(1) 014020 016000 014040 MOV   $TRPAD(R0),R0  ;;INDEX TO TABLE
(1) 014024 000200 RTS    R0            ;;GO TO ROUTINE
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 014026 011646 $TRAP2: MOV   (SP),-(SP) ;;MOVE THE PC DOWN
(1) 014030 016666 000004 000002 MOV   4(SP),2(SP)    ;;MOVE THE PSW DOWN
(1) 014036 000002 RTI                    ;;RESTORE THE PSW
(1)
(3) .SBTTL TRAP TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
    
```

				: *BY THE "TRAP" INSTRUCTION.	
				: ROUTINE	
				:-----	
(3)				\$TRPAD:	.WORD \$TRAP2
(3)	014040	014026		\$TYPE	::CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
(3)	014042	010630		\$TYPOC	::CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)	014044	011404		\$TYPOS	::CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)	014046	011360		\$TYPON	::CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3)	014050	011420		\$TYPDS	::CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(3)	014052	011606			
(1)				\$GTSWR	::CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
(3)	014054	012734			
(1)				\$CKSWR	::CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
(3)	014056	012664		\$RDCHR	::CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3)	014060	013146		\$RDLIN	::CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3)	014062	013266		\$RDOCT	::CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
(3)	014064	013440			
7156				.SBTTL	ASCII MESSAGES
7157				EM1:	.ASCIZ /REG TIMEOUT ER/
7158	014066	042522	020107	044524	
	014074	042515	052517	020124	
	014102	051105	000		
7159	014105	122	043505	051040	EM2: .ASCIZ 'REG READ/WRITE ER'
	014112	040505	027504	051127	
	014120	052111	020105	051105	
	014126	000			
7160	014127	102	051525	051040	EM3: .ASCIZ /BUS RESET ER/
	014134	051505	052105	042440	
	014142	000122			
7161	014144	047106	052103	041040	EM4: .ASCIZ /FNCT BITS FAILED TO SET STAT BITS/
	014152	052111	020123	040506	
	014160	046111	042105	052040	
	014166	020117	042523	020124	
	014174	052123	052101	041040	
	014202	052111	000123		
7162	014206	042522	042101	020131	EM5: .ASCIZ /READY INTR FAILURE/
	014214	047111	051124	043040	
	014222	044501	052514	042522	
	014230	000			
7163	014231	122	040505	054504	EM6: .ASCIZ /READY CLR OR SET ER/
	014236	041440	051114	047440	
	014244	020122	042523	020124	
	014252	051105	000		
7164	014255	123	040524	052524	EM7: .ASCIZ /STATUS ER UN XFER/
	014262	020123	051105	047440	
	014270	020116	043130	051105	
	014276	000			
7165	014277	127	051117	020104	EM10: .ASCIZ /WORD COUNT ER ON XFER/
	014304	047503	047125	020124	
	014312	051105	047440	020116	
	014320	043130	051105	000	
7166	014325	102	043125	042506	EM11: .ASCIZ /BUFFER ADRS ER ON XFER/
	014332	020122	042101	051522	
	014340	042440	020122	047117	
	014346	054040	042506	000122	
7167	014354	040504	040524	042440	EM12: .ASCIZ /DATA ER FROM MEM/

	014362	020122	051106	046517					
	014370	046440	046505	000					
7168	014375	104	052101	020101	EM13:	.ASCIZ	/DATA ER TO MEM/		
	014402	051105	052040	020117					
	014410	042515	000115						
7169	014414	044523	043516	042514	EM14:	.ASCIZ	/SINGLE CYCLE OFF DID NOT LOCK OUT CPU/		
	014422	041440	041531	042514					
	014430	047440	043106	042040					
	014436	042111	047040	052117					
	014444	046040	041517	020113					
	014452	052517	020124	050103					
	014460	000125							
7170	014462	044523	043516	042514	EM15:	.ASCIZ	/SINGLE CYCLE ON LOCKED OUT CPU/		
	014470	041440	041531	042514					
	014476	047440	020116	047514					
	014504	045503	042105	047440					
	014512	052125	041440	052520					
	014520	000							
7171	014521	015	050012	042514	WARN:	.ASCII	<15><12>/PLEASE DISABLE 'REV11' MEMORY REFRESH OPTION/		
	014526	051501	020105	044504					
	014534	040523	046102	020105					
	014542	051042	053105	030461					
	014550	020042	042515	047515					
	014556	054522	051040	043105					
	014564	042522	044123	047440					
	014572	052120	047511	116					
7172	014577	015	040412	042116		.ASCIZ	<15><12>/AND ENABLE PROCESSOR MEMORY REFRESH /		
	014604	042440	040516	046102					
	014612	020105	051120	041517					
	014620	051505	047523	020122					
	014626	042515	047515	054522					
	014634	051040	043105	042522					
	014642	044123	020040	000040					
7173	014650	042516	020130	047514	EM16:	.ASCIZ	/NEX LOGIC ER/		
	014656	044507	020103	051105					
	014664	000							
7174	014665	103	041531	042514	EM17:	.ASCIZ	/CYCLE FAILED TO CLK DBR (IN)/		
	014672	043040	044501	042514					
	014700	020104	047524	041440					
	014706	045514	042040	051102					
	014714	024040	047111	000051					
7175	014722	040504	040524	042440	EM20:	.ASCIZ	'DATA ER FROM I/O PAGE (XCSR)''		
	014730	020122	051106	046517					
	014736	044440	047457	050040					
	014744	043501	020105	054050					
	014752	051503	024522	000					
7176	014757	105	051122	041520	DH1:	.ASCIZ	/ERRPC TSTNUM BUSADR EXPCT RCVD/		
	014764	020040	052040	052123					
	014772	052516	020115	041040					
	015000	051525	042101	020122					
	015006	042440	050130	052103					
	015014	020040	051040	053103					
	015022	000104							
7177	015024	051105	050122	020103	DH2:	.ASCIZ	/ERRPC TSTNUM BUSADR ADRS EXPCT RCVD/		
	015032	020040	051524	047124					
	015040	046525	020040	052502					

	015046	040523	051104	020040	
	015054	042101	051522	020040	
	015062	020040	054105	041520	
	015070	020124	020040	041522	
	015076	042126	000		
7178	015101	105	051122	041520	DH3: .ASCIZ /ERRPC TSTNUM BUSADR/
	015106	020040	052040	052123	
	015114	052516	020115	041040	
	015122	051525	042101	000122	
7179					
7180					.EVEN
7181	015130	001116	001534	001122	DT1: \$ERRPC,TSTNUM,\$BDADR,\$GDDAT,\$BDDAT,0
	015136	001124	001126	000000	
7182	015144	001116	001534	001122	DT2: \$ERRPC,TSTNUM,\$BDADR,\$GDADR,\$GDDAT,\$BDDAT,0
	015152	001120	001124	001126	
	015160	000000			
7183	015162	001116	001534	001122	DT3: \$ERRPC,TSTNUM,\$BDADR,0
	015170	000000			
7184	015172	005015	051511	052040	KDF11B: .ASCIZ <15><12>/IS THE PROCESSOR KDF11-B? /
	015200	042510	050040	047522	
	015206	042503	051523	051117	
	015214	045440	043104	030461	
	015222	041055	020077	000040	
7185	015230	005015	051124	047101	IOTEST: .ASCIZ <15><12>.TRANSFERS FOR I/O PAGE? .
	015236	043123	051105	020123	
	015244	047506	020122	027511	
	015252	020117	040520	042507	
	015260	020077	000040		
7186	015264	005015	042101	051104	IOADR: .ASCIZ <15><12>\ADDRESS OF I/O PAGE? \
	015272	051505	020123	043117	
	015300	044440	047457	050040	
	015306	043501	037505	020040	
	015314	000			
7187		015316			.EVEN
7188					*****
7189					: 'DBUF' IS THE WORKING AREA IN EACH 4K MEM FOR ALL
7190					: NPR OPERATIONS - IT IS 200 WORDS LONG
7191					*****
7192	015316	000000			DBUF: 0 ;1ST ADRS OF DATA BUFFER

```

7194      ;;THE FOLLOWING MACRO CALL TO CNMAC2.SML WAS ADDED TO MAKE
7195      ;;THIS DIAGNOSTIC SPECIFIC TO 11/21 PROCESSOR AND WAS RENAMED
7196      ;;FROM CVDRABO TO CNDRAAO. ALSO ASSEMBLED WITH CNMAC2.SML.
7197      POINT=.          ;SAVE POINTER
(1)      . =100
(1) 000100 015320      $CLKVEC          :LKVEC HANDLER
(1) 000102 000300      300              :INTERRUPT HANDLER PRI
(1)      . =140          :BRKVEC
(1) 000140 170000      170000          :ODT START ADDRESS
(1) 000142 000300      300              :PRIORITY
(1)      . =POINT        :RESTORE POINTER
(1) 015320 104401 015326 $CLKVEC:      TYPE,CLKMES
(1) 015324 000000      HALT
(1) 015326 005015 045514 042526 CLKMES: .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
(1) 015334 020103 047111 042524
(1) 015342 051122 050125 020124
(1) 015350 020055 044504 041523
(1) 015356 047117 042516 052103
(1) 015364 046040 041524 000040
7198      000001      .END
  
```


ABASE = 174600	6050#	6073	6176	6177	6178	6179
ACDW1 = 000000	6073					
ACDW2 = 000000	6073					
ACPUOP = 000000	6073					
ADDW0 = 000000	6073					
ADDW1 = 000000	6073					
ADDW10 = 000000	6073					
ADDW11 = 000000	6073					
ADDW12 = 000000	6073					
ADDW13 = 000000	6073					
ADDW14 = 000000	6073					
ADDW15 = 000000	6073					
ADDW2 = 000000	6073					
ADDW3 = 000000	6073					
ADDW4 = 000000	6073					
ADDW5 = 000000	6073					
ADDW6 = 000000	6073					
ADDW7 = 000000	6073					
ADDW8 = 000000	6073					
ADDW9 = 000000	6073					
ADEVCT = 000000	6073					
ADEVN = 000001	6062#	6073				
AENV = 000000	6073					
AENVN = 000000	6073					
AFATAL = 000000	6073					
ALTERN = 007452	6882	6923#				
AMADR1 = 000000	6073					
AMADR2 = 000000	6073					
AMADR3 = 000000	6073					
AMADR4 = 000000	6073					
AMAMS1 = 000000	6073					
AMAMS2 = 000000	6073					
AMAMS3 = 000000	6073					
AMAMS4 = 000000	6073					
AMSGAD = 000000	6073					
AMSGLG = 000000	6073					
AMSGTY = 000000	6073					
AMTYP1 = 000000	6073					
AMTYP2 = 000000	6073					
AMTYP3 = 000000	6073					
AMTYP4 = 000000	6073					
APASS = 000000	6073					
APRIOR = 000000	6073					
APTCSU = 000040	7135	7136#				
APTENV = 000001	7135	7136#	7139			
APTSIZ = 000200	6197	7136#				
APTSP0 = 000100	7135	7136#				
ASWREG = 000000	6073					
ATESTN = 000000	6073					
AUNIT = 000000	6073					
AUSWR = 000000	6073					
AVECT1 = 000210	6061#	6073	6183	6184		
AVECT2 = 000000	6073					
BIT0 = 000001	6059#					
BIT00 = 000001	6059#	6293	6298	6347	7027	
BIT01 = 000002	6059#					

BIT02 = 000004	6059#													
BIT03 = 000010	6059#													
BIT04 = 000020	6059#													
BIT05 = 000040	6059#													
BIT06 = 000100	6059#													
BIT07 = 000200	6059#	6222												
BIT08 = 000400	6059#	7149												
BIT09 = 001000	6059#	7139	7149											
BIT11 = 000002	6059#													
BIT10 = 002000	6059#	7139												
BIT11 = 004000	6059#	7149												
BIT12 = 010000	6059#													
BIT13 = 020000	6059#	6738	7139											
BIT14 = 040000	6059#	7149												
BIT15 = 100000	6059#													
BIT2 = 000004	6059#													
BIT3 = 000010	6059#													
BIT4 = 000020	6059#													
BIT5 = 000040	6059#													
BIT6 = 000100	6059#	6429												
BIT7 = 000200	6059#													
BIT8 = 000400	6059#													
BIT9 = 001000	6059#													
BPTVEC= 000014	6059#													
BRKVEC= 000140	6059#													
CKDAT 010460	6642	7089#												
CKDAT1 010526	6744	6835	6873	7108#										
CKSTAT 010156	6513	6556	6598	6632	6726	6825	6863	6899	6941	7009#				
CKSWR = 104407	7139	7146	7149	7155#										
CLKMES 015326	7197#													
CORSZ 001540	6190#	6223	6234*	6845										
CORSZR 002304	6224	6229#												
CR = 000015	6059#	7135												
CRLF = 000200	6059#	6211	7135											
DBUF 015316	6191	6236	6501	6502*	6522	6545	6566	6568	6587	6607	6620	6653	6682	
	6714	6810	6878	6916	6958	7047	7047	7052	7089	7096	7192#			
DBUFP 001542	6191#	6236*	6843*	6844*	6845	6848	6878*	6886*	6916*	6928*	6958*	7025	7066	
	7109													
DDISP = 177570	6059#	6073	6197											
DH1 014757	6076	6082	6088	6094	6100	6106	6112	6118	6124	6154	6160	7176#		
DH2 015024	6130	6136	6169	7177#										
DH3 015101	6142	6148	7178#											
DISPLA 001142	6073#	6197*	7139*	7149*										
DISPRE 000174	6065#	6197												
DMAP 001536	6189#	6238*	6239*	6241	6966*	6975								
DRVBAR 001522	6177#	6288	6291*	6292	6339*	6346	6349	6463	6465*	6466	6471	6501*	6545*	
	6587*	6620*	6653*	6682*	6714*	6755*	6781	6785	6810*	6848*	6887*	6929*	6969*	
	7026	7030												
DRVCSR 001524	6178#	6361	6365*	6366	6376	6382	6384*	6386	6392	6407	6410*	6411	6424	
	6429*	6431	6435	6439	6447	6449*	6450	6454*	6456	6462*	6470*	6475*	6481*	
	6487*	6503*	6504*	6506	6508	6509*	6547*	6548*	6550	6552	6589*	6590*	6591	
	6593	6594*	6623*	6624*	6625	6627	6628*	6647	6655*	6656*	6663	6664*	6676	
	6684*	6685*	6690	6691*	6700	6717*	6718*	6719	6721	6722*	6757*	6758*	6759	
	6761	6762*	6766	6770	6790*	6791	6795	6799	6803	6813*	6814*	6818	6820	
	6821*	6851*	6852*	6856	6858	6859*	6890*	6891*	6892	6894	6895*	6932*	6933*	
	6934	6936	6937*	6975*	6997*	7009	7010*	7014						

\$CDW2	001256	6073#																		
\$CHARC	011106	7135#*																		
\$CKSWR	012664	7150#	7155																	
\$CLKVE	015320	7197#																		
\$CMTAG	001100	6073#	6197																	
\$CM3 =	000000	6073#																		
\$CNTLG	013411	7150#																		
\$CNTLU	013404	7150#																		
\$CPUOP	001222	6073#																		
\$CRLF	001171	6073#	7135	7139	7148	7150	7151													
\$DBLK	012022	7138#																		
\$DDW0	001260	6073#	6249*																	
\$DDW1	001262	6073#																		
\$DDW10	001304	6073#																		
\$DDW11	001306	6073#																		
\$DDW12	001310	6073#																		
\$DDW13	001312	6073#																		
\$DDW14	001314	6073#																		
\$DDW15	001316	6073#																		
\$DDW2	001264	6073#																		
\$DDW3	001266	6073#																		
\$DDW4	001270	6073#																		
\$DDW5	001272	6073#																		
\$DDW6	001274	6073#																		
\$DDW7	001276	6073#																		
\$DDW8	001300	6073#																		
\$DDW9	001302	6073#																		
\$DEVCT	001204	6073#	6246*																	
\$DEVM	001252	6073#	6238																	
\$DOAGN	010066	6978#																		
\$DTBL	012012	7138#																		
\$ENDAD	010056	6070	6978#																	
\$ENDCT	010024	6978#																		
\$ENDMG	010075	6978#																		
\$ENULL	010072	6978#																		
\$ENV	001214	6073#	6211	7135	7136	7139														
\$ENVM	001215	6073#	6197	7135	7136															
\$EOP	007770	6967	6978#																	
\$EOPCT	010016	6978#																		
\$ERFLG	001103	6073#	7139*	7149*																
\$ERMAX	001115	6073#	6197*	7149*																
\$ERROR	012032	6197	7139#																	
\$ERRPC	001116	6073#	7139*	7148	7181	7182	7183													
\$ERRTB	001320	6073#	7148																	
\$ERRTY	012240	7145	7148#																	
\$ERTTL	001112	6073#	7139*																	
\$ESCAP	001162	6073#	6197*	7139	7149*															
\$ETABL	001214	6073#																		
\$ETEND	001320	6072	6073#																	
\$FATAL	001176	6073#	7136*																	
\$FFLG	011356	7136#*																		
\$FILLC	001156	6073#	7135																	
\$FILLS	001155	6073#	7135																	
\$GDADR	001120	6073#	6522*	6566*	6607*	6914*	6956*	7094*	7121*	7182										
\$GDDAT	001124	6073#	6254*	6273*	6274	6276	6279*	6282*	6283*	6290*	6291	6294	6297*	6298*						
		6301*	6303*	6310*	6311	6313	6316*	6319*	6320*	6327*	6332	6337*	6363*	6364*						

.\$PTB	5109#	6052#	6073#
.\$PTH	5370#	6052#	6072
.\$PTY	5547#	6052#	7136
.\$STA	5417#		
.\$CATC	932#	6050#	6065
.\$CMTA	1047#	6050#	6073
.\$DB2D	4686#		
.\$DB2O	4812#		
.\$DIV	4587#		
.\$EOP	2214#	6051#	6978
.\$ERRO	2700#	6051#	7139
.\$ERRT	2896#	6051#	7148
.\$MULT	4523#		
.\$POWE	4229#	6050#	7152
.\$RAND	4307#		
.\$RDDE	3891#		
.\$RDOC	3797#	6049#	7151
.\$READ	3395#	6051#	7150
.\$R2AZ	4958#		
.\$SAVE	3969#		
.\$SB2D	4771#		
.\$SB2O	4874#		
.\$SCOP	2454#	6051#	7149
.\$SIZE	4361#		
.\$SUPR	4913#		
.\$TRAP	4073#	6049#	7155
.\$TYPB	3287#		
.\$TYPD	3209#	6051#	7138
.\$TYPE	2985#	6051#	7135
.\$TYPO	3112#	6050#	7137
.\$4OCA	972#		

. ABS. 015372 000

ERRORS DETECTED: 0

CNDRAA,CNDRAA/CRF/NL:TOC=CNMAC2.SML,CNDRAA.P11
 RUN-TIME: 13 13 .9 SECONDS
 RUN-TIME RATIO: 53/28=1.8
 CORE USED: 33K (66 PAGES)