

DRV11-J

DIAG TST PRT2
CNDRDAO

AH-T446A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



Table with multiple columns and rows of data, likely diagnostic test results. The text is very faint and difficult to read, but appears to be organized in a grid format. The table contains several columns of data, possibly representing different test parameters or components. The data is organized in a grid format, with approximately 15 columns and 15 rows of data points. The text is very faint and difficult to read, but appears to be organized in a grid format. The table contains several columns of data, possibly representing different test parameters or components. The data is organized in a grid format, with approximately 15 columns and 15 rows of data points.



IDENTIFICATION

PRODUCT NAME: CNDRDAO DRV11J DIAG TST PRT2
PRODUCT CODE: AC-T445A-MC
PRODUCT DATE: DECEMBER,1982
MAINTAINER: DIAGNOSTICS SERVICES/ISS
AUTHOR: W.HEAVEY

COPYRIGHT (C) 1982,1983
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

```

5685      ;;GPA  .HEADER ^/CNDRDA DRV11J DIAG TST PRT2 /,1982,^/BILL HEAVEY/
5686      .TITLE  CNDRDA DRV11J DIAG TST PRT2
(1)      ;*COPYRIGHT (C) 1982
(1)      ;*DIGITAL EQUIPMENT CORP.
(1)      ;*MAYNARD, MASS. 01754
(1)      ;*
(1)      ;*PROGRAM BY BILL HEAVEY
(1)      ;*
(1)      ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)      ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)
(1)      000001
(1)      160000
5687      $STN=1
5688      $SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP
5689      ;* EDITED TO PERMIT DRV'S TO RUN ON FALCON (KXT11).      ;;GPA
(1)      ;*      G. PASQUANTONIO, JULY '81      ;;GPA
(1)      ;*
(1)      $SWR=165400
5690      $STN=1
(1)      165400
5691      000001
5692      .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)      ;*      9      LOOP ON ERROR
(1)      ;*      8      LOOP ON TEST IN SWR<7:0>
5693      .SBTTL  BASIC DEFINITIONS
(1)      ;*
(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)      ;*
(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)      ;*
(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= 77 ;;GENERAL REGISTER
(1) 000006 SP= 76 ;;STACK POINTER
(1) 000007 PC= 77 ;;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
(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)          000100          ;***** THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED
(1)          000140          LKVEC= 100         ;; LINE CLOCK VECTOR
(1)          000240          BRKVEC= 140        ;; BREAK VECTOR
(1)          000240          PIRQVEC= 240       ;; PROGRAM INTERRUPT REQUEST VECTOR
5694         174160          ABASE= 174160     ;; BASE ADDRESS
5695         000001          ADEVM= 1          ;; DEFAULT TO ONE DRV11J
5696         100000          RDY= BIT15
5697         000400          DIR= BIT8
5698         001000          IE= BIT9

(1)          5700          ;CHIP COMMAND SUMMARY
5701         000020          CIRMR= 20         ;; CLEAR IRR AND IMR
5702         000030          CSIRMR= 30        ;; CLEAR SINGLE IRR AND IMR BIT
5703
5704         000040          CIMR= 40          ;; CLEAR IMR
5705         000050          CSIMR= 50        ;; CLEAR SINGLE IMR BIT
5706         000060          SIMR= 60          ;; SET ALL IMR BITS
5707         000070          SSIMR= 70        ;; SET SINGLE IMR BITS
5708
5709         000100          CIRR= 100        ;; CLEAR IRR
5710         000110          CSIRR= 110       ;; CLEAR SINGLE IRR BITS
5711         000120          SIRR= 120        ;; SET ALL IRR BITS
5712         000130          SSIRR= 130       ;; SET SINGLE IRR BITS
5713
5714         000140          CHPISR= 140       ;; CLEAR HIGHEST PRIORITY ISR BIT
5715         000160          CISR= 160        ;; CLEAR ISR
5716         000170          CSISR= 170       ;; CLEAR SINGLE ISR BIT
5717
5718         000200          LMD04= 200       ;; LOAD MODE BITS M0-M4
5719         000240          LMD57= 240       ;; LOAD MODE BITS M5-M7
  
```

```
5720
5721      ;CHIP MODE BIT PRESELECTION
5722      000240      MISR= 240
5723      000244      MIMR= 244
5724      000250      MIRR= 250
5725      000254      MACR= 254
5726
5727      ;CHIP WRITE PRESELECTION
5728      000300      PACR= 300      ;PRESELECT AUTO CLEAR REG. FOR WRITING
5729      000260      PIMR= 260      ;PRESELECT IMR REG. FOR WRITING
5730      000340      PVMA= 340      ;PRESELECT VECTOR MEMORY ADDRESS
5731
5732      .SBTTL TRAP CATCHER
(1)
(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)
(1)      000174      .=174
(1) 000174 000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000      SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER
(1)
(1) 000200 000137 002476      .SBTTL STARTING ADDRESS(ES)
5733      000100      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
5734 000100 000104 000200 000002      .=100
5735      .WORD 104,200,2      ;IF 'B EVENT' ON Q BUS IS CONNECTED
;IGNORE IT'S INTERRUPT - JUST DO A RTI
```

5737

.SBTTL ACT11 HOOKS

(1)

::*****

(2)

:HOOKS REQUIRED BY ACT11

(1)

000106

SSVPC=.

;SAVE PC

(1)

000046

.=46

(1) 000046

010712

SENDAD

::1)SET LOC.46 TO ADDRESS OF SENDAD III .SEOP

(1)

000052

.=52

(1) 000052

000000

.WORD 0

::2)SET LOC.52 TO ZERO

(1)

000106

SSVPC

:: RESTORE PC

5738

002000

.=2000

5739

;LONGEST TEST TIME

5740

;1ST PASS RUN TIME

5741

;ADDITIONAL RUN TIME

5742

.SBTTL APT PARAMETER BLOCK

(1)

::*****

(2)

:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

(1)

::*****

(2)

(1)

002000

.\$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

002000

\$APTHDR

::POINT TO APT HEADER BLOCK

(1)

002000

=\$X

::RESET LOCATION COUNTER

(2)

::*****

(1)

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

(1)

:INTERFACE SPEC.

(1)

002000

\$APTHD:

(1)

002000

000000

\$HIBTS: .WORD

0

::TWO HIGH BITS OF 18 BIT MAILBOX ADDR.

(1)

002002

002170

\$MBADR: .WORD

\$MAIL

::ADDRESS OF APT MAILBOX (BITS 0-15)

(1)

002004

000014

\$STMT: .WORD

12.

::RUN TIM OF LONGEST TEST

(1)

002006

000036

\$PASTM: .WORD

30.

::RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)

(1)

002010

000036

\$JNITM: .WORD

30.

::ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT

(1)

002012

000031

.WORD

\$ETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

5743

.SBTTL COMMON TAGS

(1)

(2)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

(2)

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

SCMTAG: .=2100

:::START OF COMMON TAGS

.WORD 0
\$TSTNM: .BYTE 0 :::CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 :::CONTAINS ERROR FLAG
\$ICNT: .WORD 0 :::CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 :::CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 :::CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 :::CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 :::CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 :::CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 :::CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 :::CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 :::CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT: .WORD 0 :::CONTAINS 'GOOD' DATA
\$BDDAT: .WORD 0 :::CONTAINS 'BAD' DATA
:::RESERVED--NOT TO BE USED
\$AUTOB: .BYTE 0 :::AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 :::INTERRUPT MODE INDICATOR
\$SWR: .WORD DSWR :::ADDRESS OF SWITCH REGISTER
\$DISPLAY: .WORD DDISP :::ADDRESS OF DISPLAY REGISTER
\$TKS: 177560 :::TTY KBD STATUS
\$TKB: 177562 :::TTY KBD BUFFER
\$TPS: 177564 :::TTY PRINTER STATUS REG. ADDRESS
\$TPB: 177566 :::TTY PRINTER BUFFER REG. ADDRESS
\$NULL: .BYTE 0 :::CONTAINS NULL CHARACTER FOR FILLS
\$FILLS: .BYTE 2
:::CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 :::INSERT FILL CHARS. AFTER A 'LINE FEED'
\$TPFLG: .BYTE 0 ::: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
\$TIMES: 0 :::MAX. NUMBER OF ITERATIONS
\$ESCAPE: 0 :::ESCAPE ON ERROR ADDRESS
\$QUES: .ASCII /?/ :::QUESTION MARK
\$CRLF: .ASCII <15> :::CARRIAGE RETURN
\$LF: .ASCIIZ <12> :::LINE FEED

.SBTTL APT MAILBOX-ETABLE

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

(2) 002206 000000
(2) 002210
(2) 002210 000
(2) 002211 000
(2)
(2) 002212 000000
(2) 002214 000000
(2) 002216 000000
(2)
(2)
(2)
(2)
(2)
(2) 002220 000
(2) 002221 000
(2)
(2)
(2)
(2) 002222 000000
(2)
(2) 002224 000
(2) 002225 000
(2) 002226 000000
(2) 002230 000
(2) 002231 000
(2) 002232 000000
(2) 002234 000
(2) 002235 000
(2) 002236 000000
(2) 002240 000000
(2) 002242 000000
(2) 002244 174160
(2)
(2) 002246 000001
(2) 002250 000000
(2) 002252

\$MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
\$ETABLE: ::APT ENVIRONMENT TABLE
\$ENV: .BYTE AENV ::ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM
::ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG ::APT SWITCH REGISTER
\$USWR: .WORD AUSWR ::USER SWITCHES
\$CPUOP: .WORD ACPUOP ::CPU TYPE,OPTIONS
BITS 15-11=CPU TYPE
11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
11/70=06,PDQ=07,Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT
\$MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS,M.S. BYTE
\$MTYP1: .BYTE AMTYP1 ::MEM. TYPE,BLK#1
MEM.TYPE BYTE -- (HIGH BYTE)
900 NSEC CORE=001
300 NSEC BIPOLAR=002
500 NSEC MOS=003
\$MADR1: .WORD AMADR1 ::HIGH ADDRESS,BLK#1
MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
\$MAMS2: .BYTE AMAMS2 ::HIGH ADDRESS,M.S. BYTE
\$MTYP2: .BYTE AMTYP2 ::MEM. TYPE,BLK#2
\$MADR2: .WORD AMADR2 ::MEM.LAST ADDRESS,BLK#2
\$MAMS3: .BYTE AMAMS3 ::HIGH ADDRESS,M.S.BYTE
\$MTYP3: .BYTE AMTYP3 ::MEM. TYPE,BLK#3
\$MADR3: .WORD AMADR3 ::MEM.LAST ADDRESS,BLK#3
\$MAMS4: .BYTE AMAMS4 ::HIGH ADDRESS,M.S.BYTE
\$MTYP4: .BYTE AMTYP4 ::MEM. TYPE,BLK#4
\$MADR4: .WORD AMADR4 ::MEM.LAST ADDRESS,BLK#4
\$VECT1: .WORD AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
\$VECT2: .WORD AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
\$BASE: .WORD ABASE
::BASE ADDRESS OF EQUIPMENT UNDER TEST
\$DEV: .WORD ADEV ::DEVICE MAP
\$CDW1: .WORD ACDW1 ::CONTROLLER DESCRIPTION WORD#1
\$ETEND:
.MEXIT

5785									
5786			:ERROR	10					
5787	002342	016121		EM10	:CHIP STAT ER				
5788	002344	016333		DH2	:ERRPC TSTNUM	BUSADR	VAM	ADDR	EXPCT RCVD
5789	002346	016552		DT2	:SERRPC TSTNUM	\$BDADR	VECVAL	VECLOC	\$GDDAT \$BDDAT
5790	002350	000000		0					
5791									
5792			:ERROR	11					
5793	002352	016136		EM11	:ILLEGAL INTERRUPT RECEIVED				
5794	002354	016420		DH3	:ERRPC TSTNUM	BUSADR	VAM	ADDR	
5795	002356	016572		DT3	:SERRPC TSTNUM	\$BDADR	VECVAL	VECLOC	
5796	002360	000000		0					
5797									
5798			:ERROR	12					
5799	002362	016166		EM12	:INTERRUPT TEST ERROR				
5800	002364	016333		DH2	:ERRPC TSTNUM	BUSADR	VAM	ADDR	EXPCT RCVD
5801	002366	016552		DT2	:SERRPC TSTNUM	\$BDADR	VECVAL	VECLOC	\$GDDAT \$BDDAT
5802	002370	000000		0					
5803									
5804			:ERROR	13	:MULTIPLE INTERRUPTS RECEIVED				
5805	002372	016213		EM13	:MULTIPLE INTERRUPTS RECEIVED				
5806	002374	016333		DH2	:ERRPC TSTNUM	BUSADR	VAM	ADDR	EXPCT RCVD
5807	002376	016552		DT2	:SERRPC TSTNUM	\$BDADR	VECVAL	VECLOC	\$GDDAT \$BDDAT
5808	002400	000000		0					
5809									
5810			:ERROR	14					
5811	002402	016066		EM7	:ILLEGAL VECTOR ADDR MEM ERROR				
5812	002404	016465		DH4	:ERRPC TSTNUM	TESTPC	BUSADR	VAM	ADDR
5813	002406	016606		DT4	:SERRPC TSTNUM	\$GDADR	\$BDADR	VECVAL	VECLOC
5814	002410	000000		0					
5815									
5816									
5817			: BUS REGISTER ADDRESS POINTERS						
5818									
5819									
5820	002412	174160		DRCSA:	ABASE				
5821	002414	174162		DRDBA:	ABASE+2				
5822	002416	174164		DRCSB:	ABASE+4				
5823	002420	174166		DRDBB:	ABASE+6				
5824	002422	174170		DR CSC:	ABASE+10				
5825	002424	174172		DRDBC:	ABASE+12				
5826	002426	174174		DR CSD:	ABASE+14				
5827	002430	174176		DRDBD:	ABASE+16				
5828									
5829									
5830			:COMMON PROGRAM LOCATION(S)						
5831									
5832	002432	000000		TSTNUM:	0				:CONTAINS TEST NUMBER ON ERROR
5833	002434	000001		DMAP:	1				
5834	002436	000000		INTFLG:	.WORD	0			
5835	002440	000000		XXDP:	.WORD	0			
5836	002442	000000		IMRLOC:	.WORD	0			
5837	002444	000000		ISRLOC:	.WORD	0			
5838	002446	000000		IRRLOC:	.WORD	0			
5839	002450	000000		ACRLOC:	.WORD	0			
5840	002452	000000		VECLOC:	.WORD	0			

CNDRDA DRV11J DIAG TST FRT2
CNDRDA.P11 10-DEC-82 14:44

MACY11 30(1046) 15-DEC-82 15:26 L 1
PAGE 57-9
ERROR POINTER TABLE

SEQ 0011

5841	002454	000000	VECPAT: .WORD	0
5842	002456	000000	VECVL: .WORD	0
5843	002460	000000	SWRSV: .WORD	0
5844	002462	000000	GRPCNT: .WORD	0
5845	002464	000000	BDSV: .WORD	0
5846	002466	000000	LVLCNT: .WORD	0
5847	002470	000000	BYCNT: .WORD	0
5848	002472	000000	BYNUM: .WORD	0
5849	002474	000000	LVLSV: .WORD	0

```
5852 .SBTTL PROGRAM START
5853 002476 START:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 002476 012706 002100 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
(1) 002502 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 002504 022706 002140 CMP #SWR,R6 ;;DONE?
(1) 002510 001374 BNE #-6 ;;LOOP BACK IF NO
(1) 002512 012706 002100 MOV #2100,SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 002516 012737 014342 000020 MOV #$$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 002524 012737 000300 000022 MOV #PR6,@#IOTVEC+2 ;;LEVEL 6
(1) 002532 012737 014014 000030 MOV #$$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 002540 012737 000300 000032 MOV #PR6,@#EMTVEC+2 ;;LEVEL 5
(1) ;;BIT02
(1) 002546 012737 015602 000034 MOV #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 002554 012737 000300 000036 MOV #PR6,@#TRAPVEC+2;LEVEL 6
(1) 002562 012737 015376 000024 MOV #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 002570 012737 000300 000026 MOV #PR6,@#PWRVEC+2 ;;LEVEL 6
(1) 002576 005037 002160 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) 002602 005037 002162 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002606 112737 000001 002115 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
(1) 002614 012737 002614 002106 MOV #.,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002622 012737 002622 002110 MOV #.,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002630 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 002634 012737 002670 000004 MOV #64$,@#ERRVEC ;;SET UP ERROR VECTOR
(2) 002642 012737 177570 002140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 002650 012737 177570 002142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 002656 022777 177777 177254 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
(2) 002664 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;;AND THE HARDWARE SWR IS NOT = -1
(2) 002666 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
(2) 002670 012716 002676 64$: MOV #65$,($P) ;;SET UP FOR TRAP RETURN
(2) 002674 000002 RTI
(2) 002676 012737 000176 002140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
(2) 002704 012737 000174 002142 MOV #DISPREG,DISPLAY
(2) 002712 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002716 005037 002176 CLR $PASS ;;CLEAR PASS COUNT
(2) 002722 132737 000200 002211 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 002730 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
(2) 002732 012737 002212 002140 MOV #$$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
(2) 002740 67$:
5854 002740 005037 002172 CLR $FATAL ;;CLEAR ERROR NUMBER
5855 002744 005037 002170 CLR $MSGTYP ;;CLEAR MESSAGE TYPE
5856 002750 005037 002174 CLR $TESTN ;;CLEAR TEST NUMBER
5857 ;;THE FOLLOWING LINES OF CODE WERE DELETED IN ORDER TO MAKE
5858 ;;THIS DIAGNOSTIC SPECIFIC TO FALCON (11/21).
5859 :CALL FALCON ;; CHECK FOR FALCON (KXT11) ;;GPA
5860 :BEQ 1000$ ;; BR IF NOT KXT11 SYSTEM ;;GPA
5861 :BIC #40,@#22 ;; FALCON, LOWER IOT... ;;GPA
5862 :BIC #40,@#32 ;;...EMT... ;;GPA
5863 :BIC #40,@#36 ;;...AND TRAP TO LEVEL 6. ;;GPA
5864 :CMP $BASE,#ABASE ;; IS $BASE VIRGIN ?? ;;GPA
```

```
5865          ;BNE 1000$          ; BR IF NOT.          ;;GPA
5866          ;MOV #174160,$BASE ; YES, USE ENGINEERING DEFAULT ;;GPA
5867          ;1000$:          ;;GPA
5868          ;CHECK OPERATING ENVIRONMENT
5869 002754 005737 000042      TST @#42          ;ARE WE IN ACT/XXDP AUTO MODE?
5870 002760 001410          BEQ 1$          ;BRANCH IF NO
5871 002762 023737 000042 000046  CMP @#42,@#46    ;IS IT ACT AUTO MODE?
5872 002770 001410          BEQ 2$          ;BRANCH IF YES
5873 002772 012737 177777 002440  MOV #-1,XXDP    ;SET XXDP CHAIN MODE INDICATOR
5874 003000 000404          BR 2$
5875 003002 123727 002210 000001 1$:  CMPB $ENV,#1    ;ARE WE IN APT AUTO MODE?
5876 003010 001003          BNE 3$          ;BRANCH IF NO
5877 003012 112737 000001 002134 2$:  MOVB #1,$AUTOB  ;SET AUTO MODE INDICATOR
5878
5879          ;PRINT TITLE IF NOT IN ACT OR APT AUTO MODE
5880 003020 005227 177777      3$:  INC #-1          ;FIRST TIME?
5881 003024 001014          BNE 5$          ;SKIP TITLE IF NO
5882 003026 004537 011306      JSR R5,TOBUF    ;STORE 0-202 INTO BUFFER AREA
5883 003032 005737 002134      TST $AUTOB     ;ARE WE IN AUTO MODE?
5884 003036 001403          BEQ 4$          ;BRANCH TO TITLE TYPEOUT IF NOT
5885 003040 005737 002440      TST XXDP       ;IS THE AUTO MODE UNDER XXDP?
5886 003044 001404          BEQ 5$          ;SKIP TITLE IF NOT
5887 003046 104401 015662      4$:  TYPE ,TITLED  ;PRINT OUT THE TITLE
5888 003052 104401 015724      TYPE ,TLCABL  ;PRINT DRV11J CABLE REQ'D
5889
5890          ;GET THE VALUE IN THE SOFTWARE SWITCH REGISTER
5891 003056 005737 002134      5$:  TST $AUTOB    ;ARE WE IN AUTOMATIC MODE?
5892 003062 001001          BNE START1    ;BRANCH IF YES
5893 003064 104406          GTSWR         ;ASK FOR SWR INPUT FROM CONSOLE
5894 003066 005037 002202      START1: CLR $UNIT   ;CLEAR UNIT NUMBER
5895 003072 013737 002246 002434  MOV $DEVN,DMAP ;UP TO 4 DRV11J'S
5896 003100 042737 177760 002434  BIC #177760,DMAP ;POSITION OF DRV11J'S
5897 003106 013701 002244      MOV $BASE,R1  ;GET BASE ADDRESS
5898 003112 010137 002412      MOV R1,DRCSA ;MAKE BUS REG.POINTER = BASE
5899 003116 032737 000001 002434  BIT #1,DMAP   ;IS FIRST DRV11-J SELECTED
5900 003124 001002          BNE NEXPAS    ;YES
5901 003126 000137 010570      JMP NXDEV1    ;NO,ADVANCE BASE DRV11J ADDRESS
5902 003132 012700 002412      NEXPAS: MOV #DRCSA,R0 ;SET UP REGISTER ADDRESS POINTERS
5903 003136 010120      NEXPA1: MOV R1,(R0)+ ;LOAD EM
5904 003140 062701 000002      ADD #2,R1    ;
5905 003144 022700 002432      CMP #DRDBD+2,R0 ;ALL DONE?
5906 003150 001372          BNE NEXPA1    ;BR IF NOT
5907 003152 004537 011456      JSR R5,TRPCAT ;EXTEND TRAP CATCHER 204-1774
5908 003156 012706 002100      MOV #2100,SP ;ALWAYS RESET STACK
5909 003162 013737 002202 002200  MOV $UNIT,$DEVCT ;LOAD APT COUNTER WITH UNIT #
5910 003170 123727 002210 000001  CMPB $ENV,#1  ;ARE WE IN APT MODE?
5911 003176 001401          BEQ NEXPA2    ;BR IF YES,SKIP RESET
5912 003200 000005          RESET       ;INITIALIZE
5913 003202 106427 000300      NEXPA2: MTPS #PR6 ;VER:0
5914
```

```

5916
5917
(3)
(3)
(2) 003206 000004
(2)
5918 003210 013700 002412
5919 003214 013701 002416
5920 003220 013702 002412
5921 003224 012737 003276 002110
5922 003232 012737 000002 002462
5923 003240 004537 010746
5924 003244 012703 012570
5925 003250 012737 000050 002442
5926 003256 004537 011026
5927 003262 012737 011576 000300
5928 003270 012737 000300 000302
5929 003276 005037 002436
5930 003302 012706 002100
5931 003306 112710 000340
5932 003312 112711 000060
5933 003316 010037 002122
5934 003322 112710 000060
5935 003326 113710 002442
5936 003332 112710 000202
5937 003336 106427 000000
5938 003342 112712 000241
5939
5940
5941 003346 112710 000241
5942 003352 005737 002436
5943 003356 001401
5944 003360 104011
5945 003362 005037 002436
5946 003366 112710 000120
5947 003372 106427 000300
5948 003376 005737 002436
5949 003402 001401
5950 003404 104011
5951 003406 005037 002436
5952 003412 106427 000000
5953 003416 152762 000002 000001
5954 003424 012737 000001 002124
5955 003432 013737 002436 002126
5956 003440 023737 002124 002126
5957 003446 001401
5958 003450 104012
5959 003452 106427 000300
5960 003456 012737 101710 002124
5961 003464 010237 002122
5962 003470 011237 002126
5963 003474 042737 000007 002126
5964 003502 023737 002124 002126
5965 003510 001401
5966 003512 104002
5967 003514 010137 002122

```

```

*****
*TEST 1 TEST CHIP INTERRUPT ,GROUP 1,GROUP 2
*****
TST1: SCOPE

```

```

MOV DRCSA,R0 ;START WITH GROUP 1
MOV DRCSB,R1
MOV DRCSA,R2 ;REGISTER IS COMMON FOR BOTH GROUPS
MOV #111$, $LPERR ;SET UP LOOP RETURN
MOV #2,GRPCNT ;COUNTER FOR BOTH GROUPS
JSR R5,CLRCSR ;CLEAR ALL CSRS
MOV #BGCHP3,R3 ;EXPECTED GDDAT FOR ISR,IMR
MOV #CSIMR,IMRLOC ;STORE CLEAR SINGLE IMR CODE
JSR R5,CLRIRR ;CLEAR IRR REGS
MOV #INTSR1,300 ;STORE VECTOR ROUTINE
MOV #300,302 ;STORE PSW ;VER:0
111$: CLR INTFLG ;CLEAR INT. FLAG
MOV #2100,SP ;INIT STACK IN CASE OF ILLEGAL INT. LOOP
MOVB #PVMA,(R0) ;PRESELECT VECTOR MEM ADDR
MOVB #60,(R1) ;WRITE VECTOR 300,BY0,LV0
MOV R0,$BDADR ;SAVE BUS ADDRESS
MOVB #SIMR,(R0) ;SET ALL IMR BITS
MOVB IMRLOC,(R0) ;CLEAR MASK ON SINGLE BIT
MOVB #202,(R0) ;LMD04 - COMMON VECTOR
MIPS #PRO
MOVB #241,(R2) ;USED IN TEST OF GROUP 2
;ARM GROUP 1 TO PASS
;ENABLE TO GROUP 2.
;LMD57 - ARM THE CHIP
;CHECK FOR INTERRUPTS
1$: CLR INTFLG ;ERROR,ILLEGAL INTERRUPT
;RESET INT. COUNTER
MOVB #SIRR,(R0) ;SET ALL IRR BITS
MTPS #PR6 ;VER:0
TST INTFLG ;CHECK FOR NO INTERRUPTS
BEQ 2$
ERROR 11 ;ILLEGAL INTERRUPTS
2$: CLR INTFLG ;CLEAR INT. COUNTER
MTPS #PRO
BISB #BIT1,1(R2) ;SET I/E,EXPECT INTERRUPT
MOV #1,$GDDAT ;EXPECT ONE INTERRUPT
MOV INTFLG,$BDDAT ;SHOULD HAVE ONE INTERRUPT
CMP $GDDAT,$BDDAT
BEQ 3$
ERROR 12 ;INTERRUPT TEST ERROR
3$: MTPS #PR6 ;VER:0
MOV #101710,$GDDAT ;RDY,DIR,I/E, CHIP - 31X
MOV R2,$BDADR ;CSRA ADDRESS
MOV (R2),$BDDAT
BIC #7,$BDDAT ;CLEAR UNDEFINED BITS
CMP $GDDAT,$BDDAT
BEQ 4$
ERROR 2 ;CSRA REG ERROR
4$: MOV R1,$BDADR ;READY TO READ ISR REG.

```

CNDRDA DRV11J DIAG TST PRT2
CNDRDA.P11 10-DEC-82 14:44

MACY11 30(1046)
T1

15-DEC-82 15:26 PAGE 58-1
TEST CHIP INTERRUPT ,GROUP 1, GROUP 2

SEQ 0015

5968	003520	005037	002126			CLR	\$BDDAT	;SET UP FOR BYTE READS
5969	003524	005037	002124			CLR	\$GDDAT	;SET UP FOR BYTE EXPECTED
5970	003530	111337	002124			MOVB	(R3), \$GDDAT	;EXPECTED DATA
5971	003534	112710	000240			MOVB	#MISR, (R0)	;LOAD MODE TO READ ISR
5972	003540	111137	002126			MOVB	(R1), \$BDDAT	
5973	003544	023737	002124	002126		CMP	\$GDDAT, \$BDDAT	
5974	003552	001401				BEQ	5\$	
5975	003554	104006				ERROR	6	;ISR ERROR
5976	003556	116337	000001	002124	5\$:	MOVB	1(R3), \$GDDAT	;STORE EXPECTED
5977	003564	112710	000244			MOVB	#MIMR, (R0)	;LOAD MODE FOR IMR
5978	003570	111137	002126			MOVB	(R1), \$BDDAT	
5979	003574	023737	002124	002126		CMP	\$GDDAT, \$BDDAT	
5980	003602	001401				BEQ	6\$	
5981	003604	104005				ERROR	5	;IMR ERROR
5982	003606	116337	000001	002124	6\$:	MOVB	1(R3), \$GDDAT	;EXPECTED IRR
5983	003614	112710	000250			MOVB	#MIRR, (R0)	;LOAD MODE BITS FOR IRR
5984	003620	111137	002126			MOVB	(R1), \$BDDAT	
5985	003624	023737	002124	002126		CMP	\$GDDAT, \$BDDAT	
5986	003632	001401				BEQ	7\$	
5987	003634	104003				ERROR	3	;IRR ERROR
5988	003636	005037	002124		7\$:	CLR	\$GDDAT	
5989	003642	112710	000254			MOVB	#MACR, (R0)	
5990	003646	111137	002126			MOVB	(R1), \$BDDAT	
5991	003652	023737	002124	002126		CMP	\$GDDAT, \$BDDAT	
5992	003660	001401				BEQ	10\$	
5993	003662	104004				ERROR	4	;ACR ERROR
5994	003664	005037	002124		10\$:	CLR	\$GDDAT	
5995	003670	112710	000140			MOVB	#CHPISR, (R0)	;CLEAR HIGHEST PRIOR ISR
5996	003674	112710	000240			MOVB	#MISR, (R0)	;LOAD MODE TO READ ISR
5997	003700	111137	002126			MOVB	(R1), \$BDDAT	
5998	003704	023737	002124	002126		CMP	\$GDDAT, \$BDDAT	
5999	003712	001401				BEQ	11\$	
6000	003714	104006				ERROR	6	;ISR REG ERROR
6001	003716	012737	101710	002124	11\$:	MOV	#101710, \$GDDAT	;RDY, DIR, I/E, CHIP - 31X
6002	003724	010237	002122			MOV	R2, \$BDADR	;CSRA ADDRESS
6003	003730	011237	002126			MOV	(R2), \$BDDAT	
6004	003734	042737	000007	002126		BIC	#7, \$BDDAT	;CLEAR UNDEFINED BITS
6005	003742	023737	002124	002126		CMP	\$GDDAT, \$BDDAT	
6006	003750	001401				BEQ	12\$	
6007	003752	104002				ERROR	2	;CSRA REG ERROR
6008	003754	012737	100710	002124	12\$:	MOV	#100710, \$GDDAT	;RDY, DIR, CHIP - 31X
6009	003762	042712	001000			BIC	#BIT9, (R2)	;CLEAR I/E
6010	003766	011237	002126			MOV	(R2), \$BDDAT	;READ CSRA
6011	003772	042737	000007	002126		BIC	#7, \$BDDAT	;CLEAR UNDEFINED BITS
6012	004000	023737	002124	002126		CMP	\$GDDAT, \$BDDAT	
6013	004006	001401				BEQ	13\$	
6014	004010	104002				ERROR	2	;CSRA REG ERROR
6015	004012	005723			13\$:	TST	(R3)+	;GET NEXT ISR, IMR EXPECTED RESULTS
6016	004014	020327	012610			CMP	R3, #EDCHP3	;CHECK FOR END
6017	004020	001404				BEQ	14\$	
6018	004022	005237	002442			INC	IMRLOC	;INCREMENT MASK BIT TO CLEAR
6019	004026	000137	003276			JMP	111\$;TEST NEXT ISR BIT
6020	004032	005337	002462		14\$:	DEC	GRPCNT	;TESTED BOTH GROUPS?
6021	004036	001406				BEQ	TST2	;BR IF DONE
6022	004040	013700	002422			MOV	DRCSC, R0	;SETUP FOR GROUP 2
6023	004044	013701	002426			MOV	DRCSD, R1	;CSRC = CONTROL GROUP 2


```
6024 ;CSR = DATA GROUP 2
6025 004050 000137 003240 JMP 120$ ;DO GROUP 2
6026
6027
6028 :*****
(3) :*TEST 2 TEST ACR/ISR INTERRUPT ,GROUP 1,GROUP 2
(3) :*****
(2) 004054 000004 TST2: SCOPE
(2)
6029 004056 013700 002412 MOV DRCSA,R0
6030 004062 013701 002416 MOV DRCSB,R1
6031 004066 013702 002412 MOV DRCSA,R2
6032 004072 012737 004144 002110 MOV #111$,$LPER? ;SET UP LOOP RETURN
6033 004100 012737 000002 002462 MOV #2,GRPCNT ;DO TWO GROUPS
6034 004106 004537 010746 120$: JSR R5,CLRCSR ;CLEAR ALL CSRS
6035 004112 012703 012570 MOV #BGCHP3,R3 ;EXPECTED GDDAT FOR ISR
6036 004116 012737 000000 002442 MOV #CSIMR,IMRLOC ;STORE CLEAR SINGLE IMR CODE
6037 004124 004537 011026 JSR R5,CLRIRR ;CLEAR IRR REGS
6038 004130 012737 011576 000300 MOV #INTSR1,300 ;STORE VECTOR ROUTINE
6039 004136 012737 000300 000302 MOV #300,302 ;STORE PSW ;VER:0
6040 004144 005037 002436 111$: CLR INTFLG ;CLEAR INT. FLAG
6041 004150 012706 002100 MOV #2100,SP ;INIT STACK IN CASE OF ILLEGAL INT. LOOP
6042 004154 042712 001000 BIC #BIT9,(R2) ;CLEAR INTERRUPT ENABLE
6043 004160 112712 000242 MOV #242,(R2) ;CLEAR MASTER MASK GP1
6044 004164 112710 000242 MOV #242,(R0) ;CLEAR MASTER MASK GP1,GP2
6045 004170 112710 000300 MOV #PACR,(R0) ;PRESELECT ACR FOR WRITING
6046 004174 111311 MOV #R3,(R1) ;NEXT BIT INTO ACR
6047 004176 112710 000340 MOV #PVMA,(R0) ;PRESELECT VECTOR MEM ADDR
6048 004202 112711 000060 MOV #60,(R1) ;WRITE VECTOR 300,BYO,LVO
6049 004206 010037 002122 MOV R0,$BADDR ;STORE BUS ADDRESS
6050 004212 112710 000060 MOV #SIMR,(R0) ;SET ALL IMR BITS
6051 004216 113710 002442 MOV IMRLOC,(R0) ;CLEAR MASK ON SINGLE BIT
6052 004222 112710 000202 MOV #202,(R0) ;LMD04 - COMMON VECTOR
6053 004226 106427 000000 MTPS #PRO
6054 004232 152762 000002 000001 BISB #BIT1,1(R2) ;SET I/E,NO INTERRUPT
6055 004240 005737 002436 TST INTFLG ;CHECK FOR INTERRUPTS
6056 004244 001401 BEQ 1$
6057 004246 104011 ERROR 11 ;ERROR,ILLEGAL INTERRUPT
6058 004250 005037 002436 1$: CLR INTFLG ;RESET INT. COUNTER
6059 004254 112710 000120 MOV #SIRR,(R0) ;SET ALL IRR BITS
6060 004260 106427 000300 MTPS #PR6 ;VER:0
6061 004264 005737 002436 TST INTFLG ;CHECK FOR NO INTERRUPTS
6062 004270 001401 BEQ 2$
6063 004272 104011 ERROR 11 ;ILLEGAL INTERRUPTS
6064 004274 005037 002436 2$: CLR INTFLG ;CLEAR INT. COUNTER
6065 004300 106427 000000 MTPS #PRO
6066 004304 112712 000241 MOV #241,(R2) ;USED IN TEST OF GROUP 2
6067 ;ARM GROUP 1 CHIP TO PASS
6068 ;ENABLE TO GROUP 2 CHIP.
6069 004310 112710 000241 MOV #241,(R0) ;ARM THE CHIP
6070 004314 012737 000001 002124 MOV #1,$GDDAT ;EXPECT ONE INTERRUPT
6071 004322 013737 002436 002126 MOV INTFLG,$BDDAT ;SHOULD HAVE ONE INTERRUPT
6072 004330 023737 002124 002126 CMP $GDDAT,$BDDAT
6073 004336 001401 BEQ 3$
6074 004340 104012 ERROR 12 ;INTERRUPT TEST ERROR
6075 004342 106427 000300 3$: MTPS #PR6 ;VER:0
```

```

6076 004346 010137 002122 4$: MOV R1,$BADDR ;READY TO READ ISR REG.
6077 004352 005037 002126 CLR $BDDAT ;SET UP FOR BYTE READS
6078 004356 005037 002124 CLR $GDDAT ;SET UP FOR BYTE EXPECTED
6079 004362 112710 000240 MOVB #MISR,(R0) ;LOAD MODE TO READ ISR
6080 004366 111137 002126 MOVB (R1),$BDDAT
6081 004372 023737 002124 002126 CMP $GDDAT,$BDDAT ;ACR SHOULD CLEAR ISR
6082 004400 001401 BEQ 5$
6083 004402 104006 ERROR 6 ;ISR ERROR
6084 004404 116337 000001 002124 5$: MOVB 1(R3),$GDDAT ;STORE EXPECTED
6085 004412 112710 000244 MOVB #MIMR,(R0) ;LOAD MODE FOR IMR
6086 004416 111137 002126 MOVB (R1),$BDDAT
6087 004422 023737 002124 002126 CMP $GDDAT,$BDDAT
6088 004430 001401 BEQ 6$
6089 004432 104005 ERROR 5 ;IMR ERROR
6090 004434 116337 000001 002124 6$: MOVB 1(R3),$GDDAT ;EXPECTED IRR
6091 004442 112710 000250 MOVB #MIRR,(R0) ;LOAD MODE BITS FOR IRR
6092 004446 111137 002126 MOVB (R1),$BDDAT
6093 004452 023737 002124 002126 CMP $GDDAT,$BDDAT
6094 004460 001401 BEQ 7$
6095 004462 104003 ERROR 3 ;IRR ERROR
6096 004464 111337 002124 7$: MOVB (R3),$GDDAT ;EXPECTED ACR
6097 004470 112710 000254 MOVB #MACR,(R0)
6098 004474 111137 002126 MOVB (R1),$BDDAT
6099 004500 023737 002124 002126 CMP $GDDAT,$BDDAT
6100 004506 001401 BEQ 10$
6101 004510 104004 ERROR 4 ;ACR ERROR
6102 004512 005723 10$: TST (R3)+ ;GET NEXT ISR,IMR,IRR EXPECTED RESULTS
6103 004514 020327 012610 CMP R3,#EDCHP3 ;CHECK FOR END
6104 004520 001404 BEQ 11$ ;DONE WITH GROUP TEST
6105 004522 005237 002442 INC IMRLOC ;INCREMENT MASK BIT TO CLEAR
6106 004526 000137 004144 JMP 111$ ;TEST NEXT ISR BIT
6107 004532 005337 002462 11$: DEC GRPCNT ;TESTED BOTH GROUPS?
6108 004536 001406 BEQ TST3 ;:BR IF DONE
6109 004540 013700 002422 MOV DRCSA,R0 ;SETUP FOR GROUP 2
6110 004544 013701 002426 MOV DRCSA,R1 ;CSRC = CONTROL GROUP 2
6111 ;CSR = DATA GROUP 2
6112 004550 000137 004106 JMP 120$ ;DO GROUP 2
6113
6114
6115
6116 ;*****
6117 (3) ;*TEST 3 TEST VECTOR ADDR MEM,LEV 0-7,BY0-3,(204-1774),GRPS 1,2
6118 (3) ;*****
6119 (2) 004554 000004 TST3: SCOPE
6120 (2)
6121 004556 013700 002412 MOV DRCSA,R0
6122 004562 013701 002416 MOV DRCSB,R1
6123 004566 013702 002412 MOV DRCSA,R2
6124 004572 012737 000002 002462 MOV #2,GRPCNT ;DO TWO GROUPS
6125 004600 012737 000001 002472 120$: MOV #1,BYNUM ;INIT FOR BYTE COUNT 0
6126 004606 004537 010746 JSR R5,CLRCSR ;CLEAR ALL CSRS
6127 004612 012737 000340 002456 MOV #PVMA,VECVAL ;START VECTOR LEVEL 0,BYTE COUNT 0
6128 004620 012704 012570 121$: MOV #BGCHP3,R4 ;EXPECTED ISR,IMR PATTERN
6129 004624 012737 000010 002466 MOV #8,LVLCNT ;COUNTER FOR 0-7 LEVELS
6130 004632 012737 000050 002442 MOV #CSIMR,IMRLOC ;STORE SINGLE IMR CODE
6131 004640 012737 000130 002446 MOV #SSIRR,IRRLOC ;STORE SINGLE IRR CODE

```

CNRDA DRV11J DIAG TST PRT2
CNRDA.P11 10-DEC-82 14:44

MACY11 30(1046)
T3

15-DEC-82 15:26 PAGE 58-4
TEST VECTOR ADDR MEM,LEV 0-7,BY0-3,(204-1774),GRPS 1,2

SEQ 0018

6128	004646	012737	000170	002444	MOV	#CSISR,ISRLOC	:STORE CLEAR SINGLE ISR CODE
6129	004654	004537	011456		JSR	R5,TRPCAT	:RESTORE TRAP CATCHER
6130	004660	012737	004674	002110	MOV	#112\$,SLPERR	:STORE SCOPE LOOP
6131	004666	012737	000204	002452	MOV	#204,VECLC	:START AT VECTOR 204
6132	004674	106427	000300		MTPS	#300	:VER:0
6133	004700	012706	002100		MOV	#2100,SP	:INIT STACK IN CASE OF ILLEGAL INT. LOOP
6134	004704	042712	001000		BIC	#BIT9,(R2)	:CLEAR INTERRUPT ENABLE
6135	004710	004537	011026		JSR	R5,CLRIRR	:CLEAR IRR REGS
6136	004714	112712	000241		MOVB	#241,(R2)	:USED IN TEST OF GROUP 2
6137							:ARM GROUP 1 CHIP TO PASS
6138							:ENABLE TO GROUP2
6139	004720	013703	002452		MOV	VECLC,R3	:GET VECTOR
6140	004724	010337	002454		MOV	R3,VECPAT	
6141	004730	006037	002454		ROR	VECPAT	:ROTATE VECTOR ADDR TWO RIGHT
6142	004734	006037	002454		ROR	VECPAT	:TO WRITE VECTOR MEM ADDR.
6143	004740	012713	011632		MOV	#INTBY4,(R3)	:STORE VECTOR ROUTINE
6144	004744	012763	000300	000002	MOV	#300,2(R3)	:STORE PSW ;VER:0
6145	004752	005037	002436		CLR	INTFLG	:CLEAR INT. FLAG
6146	004756	013737	002472	002470	MOV	BYNUM,BYCNT	:BYTE COUNTS OF 0-3
6147	004764	113710	002456		MOVB	VECVL,(R0)	:PRESELECT VECTOR MEM ADDR
6148	004770	113711	002454		MOVB	VECPAT,(R1)	:WRITE VECTOR
6149	004774	005337	002470		DEC	BYCNT	:DONE WITH BYTE COUNT VALUE
6150	005000	001373			BNE	113\$:NO,LOAD VECTOR FOR NEXT BYTE COUNT
6151	005002	010037	002122		MOV	R0,\$BDADR	:SAVE BUS ADDRESS
6152	005006	112710	000060		MOVB	#SIMR,(R0)	:SET ALL IMR BITS
6153	005012	113710	002442		MOVB	IMRLOC,(R0)	:CLEAR MASK ON SINGLE BIT
6154	005016	106427	000000		MTPS	#PRO	
6155	005022	112710	000241		MOVB	#241,(R0)	:LMD57 - ARM THE CHIP
6156	005026	005737	002436		TST	INTFLG	:CHECK FOR INTERRUPTS
6157	005032	001401			BEQ	1\$	
6158	005034	104011			ERROR	11	:ERROR,ILLEGAL INTERRUPT
6159	005036	005037	002436		CLR	INTFLG	:RESET INT. COUNTER
6160	005042	113710	002446		MOVB	IRRLOC,(R0)	:SET SINGLE IRR BIT
6161	005046	106427	000300		MTPS	#PR6 ;VER:0	
6162	005052	005737	002436		TST	INTFLG	:CHECK FOR NO INTERRUPTS
6163	005056	001401			BEQ	2\$	
6164	005060	104011			ERROR	11	:ILLEGAL INTERRUPTS
6165	005062	005037	002436		CLR	INTFLG	:CLEAR INT. COUNTER
6166	005066	106427	000000		MTPS	#PRO	
6167	005072	152762	000002	000001	BISB	#BIT1,1(R2)	:SET I/E,EXPECT INTERRUPT
6168	005100	013737	002472	002124	MOV	BYNUM,\$GDDAT	:EXPECT 1 TO 4 INTERRUPTS
6169							:BASED ON BYTE COUNT VALUE
6170	005106	013737	002436	002126	MOV	INTFLG,\$BDDAT	:SHOULD HAVE 1 TO 4 INTERRUPTS
6171							:BASED ON BYTE COUNT VALUE
6172	005114	023737	002124	002126	CMP	\$GDDAT,\$BDDAT	
6173	005122	001401			BEQ	3\$	
6174	005124	104012			ERROR	12	:INTERRUPT TEST ERROR
6175	005126	106427	000300		MTPS	#PR6 ;VER:0	
6176	005132	010137	002122		MOV	R1,\$BDADR	:READY TO READ ISR REG.
6177	005136	005037	002126		CLR	\$BDDAT	:SET UP FOR BYTE READS
6178	005142	005037	002124		CLR	\$GDDAT	:SET UP FOR BYTE EXPECTED
6179	005146	111437	002124		MOVB	(R4),\$GDDAT	:EXPECTED DATA
6180	005152	112710	000240		MOVB	#MISR,(R0)	:LOAD MODE TO READ ISR
6181	005156	111137	002126		MOVB	(R1),\$BDDAT	
6182	005162	023737	002124	002126	CMP	\$GDDAT,\$BDDAT	
6183	005170	001401			BEQ	5\$	

```

6184 005172 104006          ERROR 6          ;ISR ERROR
6185 005174 116437 000001 002124 5$:  MOVB 1(R4),SGDDAT ;STORE EXPECTED
6186 005202 112710 000244          MOVB #MIMR,(R0)   ;LOAD MODE FOR IMR
6187 005206 111137 002126          MOVB (R1),SBDDAT
6188 005212 023737 002124 002126  CMP  SGDDAT,SBDDAT
6189 005220 001401          BEQ  6$
6190 005222 104005          ERROR 5          ;IMR ERROR
6191 005224 005037 002124          CLR  SGDDAT
6192 005230 112710 000250          MOVB #MIRR,(R0)   ;LOAD MODE BITS FOR IRR
6193 005234 111137 002126          MOVB (R1),SBDDAT
6194 005240 023737 002124 002126  CMP  SGDDAT,SBDDAT
6195 005246 001401          BEQ  7$
6196 005250 104003          ERROR 3          ;IRR ERROR
6197 005252 005037 002124          CLR  SGDDAT
6198 005256 112710 000254          MOVB #MACR,(R0)
6199 005262 111137 002126          MOVB (R1),SBDDAT
6200 005266 023737 002124 002126  CMP  SGDDAT,SBDDAT
6201 005274 001401          BEQ  10$
6202 005276 104004          ERROR 4          ;ACR ERROR
6203 005300 005037 002124          CLR  SGDDAT
6204 005304 113710 002444          MOVB ISRLOC,(R0) ;CLEAR SINGLE ISR BIT
6205 005310 112710 000240          MOVB #MISR,(R0) ;LOAD MODE TO READ ISR
6206 005314 111137 002126          MOVB (R1),SBDDAT
6207 005320 023737 002124 002126  CMP  SGDDAT,SBDDAT
6208 005326 001401          BEQ  11$
6209 005330 104006          ERROR 6          ;ISR REG ERROR
6210 005332 004537 011364          JSR  R5,RESTRP   ;RESTORE VECTOR JUST TESTED
6211 005336 005737 017302          TST  KXTFLAG     ; KXT11 ??
6212 005342 001404          BEQ  100$        ; BR IF NOT
6213 005344 022737 000400 002452  CMP  #400,VECLOC ; YES, FINISHED 204-374 ??
6214 005352 000403          403             ; SKIP NEXT TO FIND OUT
6215 005354          100$:
6216 005354 022737 002000 002452  CMP  #2000,VECLOC ;FINISHED?
6217 005362 001402          BEQ  12$        ;FINISHED VECTORS 204-1774?
6218 005364 000137 004674          JMP  112$       ;TEST NEXT VECTOR ADDR
6219 005370 005724          12$:
6220 005372 005237 002442          TST  (R4)+      ;INDEX EXPECTED ISR AND IMR PATTERN
6221 005376 005237 002446          INC  IMRLOC     ;STORE CLEAR FOR NEXT MASK BIT
6222 005402 005237 002444          INC  IRRLOC     ;STORE SET FOR NEXT IRR BIT
6223 005406 005237 002456          INC  ISRLOC     ;STORE CLEAR FOR NEXT ISR BIT
6224 005412 005337 002466          INC  VECVAL     ;SETUP TO TEST NEXT VECTOR LEVEL
6225 005416 001402          DEC  LVLCNT     ;FINISHED LEVELS 0-7
6226 005420 000137 004666          BEQ  13$        ;YES,CHECK BYTE COUNT END
6227 005424 005237 002472          JMP  111$       ;NO,DO NEXT LEVEL
6228 005430 104407          13$:
6229 005432 022737 000400 002456  INC  BYNUM      ;INDEX BYTE COUNT
6230 005440 001402          CKSWR          ;LOOK FOR SWITCH CHANGE REQUEST
6231 005442 000137 004620          CMP  #400,VECVAL ;FINISHED ALL BYTE COUNTS 0-3
6232          14$:
6233 005446 005337 002462          BEQ  14$        ;FINISHED TWO GROUPS?
6234 005452 001406          TST  14$        ;DO NEXT BYTE COUNT FOR
6235 005454 013700 002422          MOV  121$       ;LEVELS 0-7
6236 005460 013701 002426          MOV  111$       ;FINISHED TWO GROUPS?
6237 005464 000137 004600          JMP  120$       ;BR IF DONE
6238          ;CSRC = CONTROL GROUP 2
6239          ;CSRD = DATA GROUP 2
          ;DO GROUP 2 FOR BYTE COUNTS 0-3
          ;AND LEVELS 0-7 FOR EACH BYTE COUNT.

```

```
6240
6241
(3)
(3)
(2) 005470 000004
(2)
6242 005472 013700 002412
6243 005476 013701 002416
6244 005502 013702 002412
6245 005506 012737 000002 002462
6246 005514 012737 000001 002472 120$:
6247 005522 004537 010746
6248 005526 012737 000340 002456
6249 005534 012704 012570 121$:
6250 005540 012737 000010 002466
6251 005546 012737 000050 002442
6252 005554 012737 000130 002446
6253 005562 012737 005574 002110
6254 005570 005037 002452 111$:
6255 005574 106427 000300 112$:
6256 005600 012706 002100
6257 005604 042712 001000
6258 005610 004537 011026
6259 005614 112712 000241
6260
6261
6262 005620 013703 002452
6263 005624 004537 011756
6264 005630 010337 002454
6265 005634 006037 002454
6266 005640 006037 002454
6267 005644 012713 012056
6268 005650 012763 000300 000002
6269 005656 005037 002436
6270 005662 013737 002472 002470
6271
6272 005670 113710 002456
6273 005674 113711 002454 113$:
6274 005700 005337 002470
6275 005704 001373
6276 005706 010037 002122
6277 005712 112710 000060
6278 005716 113710 002442
6279 005722 106427 000000
6280 005726 112710 000241
6281 005732 005737 002436
6282 005736 001405
6283 005740 004537 011326
6284 005744 104011
6285 005746 004537 011756
6286 005752 005037 002436 1$:
6287 005756 113710 002446
6288 005762 106427 000300
6289 005766 005737 002436
6290 005772 001405
6291 005774 004537 011326

:*****
:*TEST 4 TEST VECTOR ADDR MEM,LVLS 0-7,BYO-3,(0-200)GRPS 1,2
:*****
TST4: SCOPE
MOV DRCSA,R0
MOV DRCSB,R1
MOV DRCSA,R2
MOV #2,GRPCNT :DO TWO GROUPS
MOV #1,BYNUM :INIT FOR FIRST BYTE COUNT
JSR R5,CLRCSR :CLEAR ALL CSRS
MOV #PVMA,VECVAL :START VECTOR LEVEL 0,BYTE COUNT 0
MOV #BGCHP3,R4 :ISR,IMR PATTERN
MOV #8,LVLCNT :START LEVEL COUNT 0-7
MOV #CSIMR,IMRLOC :STORE CLEAR SINGLE IMR BIT CODE
MOV #SSIRR,IRRLOC :STORE SET SINGLE IRR BIT CODE
MOV #112$, $LPERR :LOOP RETURN
CLR VECLOC :START AT VECTOR 0
MTPS #300 :VER:0
MOV #2100,SP :INIT STACK IN CASE OF ILLEGAL INT. LOOP
BIC #BIT9,(R2) :CLEAR INTERRUPT ENABLE
JSR R5,CLRIRR :CLEAR IRR REGS
MOVB #241,(R2) :USED IN TEST OF GROUP 2
:ARM GROUP 1 CHIP TO PASS
:ENABLE TO GROUP 2.
:GET VECTOR
:STORE TRAP CATCHER FOR 0-200
MOV VECLOC,R3
JSR R5,CAT200
MOV R3,VECPAT
ROR VECPAT :ROTATE VECTOR ADDR TWO RIGHT
ROR VECPAT :TO WRITE VECTOR MEM ADDR.
MOV #INTSR3,(R3) :STORE VECTOR ROUTINE
MOV #300,2(R3) :STORE PSW;VER:0
CLR INTFLG :CLEAR INT. FLAG
MOV BYNUM,BYCNT :BYTE COUNT OF 0 TO 3
:FOR 1 TO 4 VECTORS.
MOVB VECVAL,(R0) :PRESELECT VECTOR MEM ADDR
MOVB VECPAT,(R1) :WRITE VECTOR
DEC BYCNT :DO 1 TO 4 VECTORS
BNE 113$ :WRITE VECTORS BASED ON BYTE COUNT
MOV R0,$BDADR :SAVE BUS ADDRESS
MOVB #SIMR,(R0) :SET ALL IMR BITS
MOVB IMRLOC,(R0) :CLEAR MASK ON SINGLE BIT
MTPS #PRO
MOVB #241,(R0) :LMD57 - ARM THE CHIP
TST INTFLG :CHECK FOR INTERRUPTS
BEQ 1$
JSR R5,FRMBUF :RESTORE 0-202
ERROR 11 :ERROR,ILLEGAL INTERRUPT
JSR R5,CAT200 :RESTORE TRAP CATCHER
CLR INTFLG :RESET INT. COUNTER
MOVB IRRLOC,(R0) :SET SINGLE IRR BIT
MTPS #PR6 ;VER:0
TST INTFLG :CHECK FOR NO INTERRUPTS
BEQ 2$
JSR R5,FRMBUF :RESTORE 0-202
```

6292	006000	104011			ERROR	11		:ILLEGAL INTERRUPTS
6293	006002	004537	011756		JSR	R5,CAT200		:RESTORE TRAP CATCHER
6294	006006	005037	002436		2\$: CLR	INTFLG		:CLEAR INT. COUNTER
6295	006012	106427	000000		MTPS	#PRO		
6296	006016	152762	000002	000001	BISB	#BIT1,1(R2)		:SET I/E,EXPECT INTERRUPT
6297	006024	013737	002472	002124	MOV	BYNUM,\$GDDAT		:EXPECT NUMBER OF INTERRUPTS
6298								:EQUAL TO BYTE COUNT.
6299	006032	013737	002436	002126	MOV	INTFLG,\$BDDAT		:SHOULD HAVE NUMBER OF INTERRUPTS
6300								:EQUAL TO BYTE COUNT.
6301	006040	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6302	006046	001405			BEQ	3\$		
6303	006050	004537	011326		JSR	R5,FRMBUF		:RESTORE 0-202
6304	006054	104012			ERROR	12		:INTERRUPT TEST ERROR
6305	006056	004537	011756		JSR	R5,CAT200		:RESTORE TRAP CATCHER SUB
6306	006062	106427	000300		3\$: MTPS	#PR6 ;VER:0		
6307	006066	010137	002122		4\$: MOV	R1,\$BDDADR		:READY TO READ ISR REG.
6308	006072	005037	002126		CLR	\$BDDAT		:SET UP FOR BYTE READS
6309	006076	005037	002124		CLR	\$GDDAT		:SET UP FOR BYTE EXPECTED
6310	006102	111437	002124		MOVB	(R4),\$GDDAT		:EXPECTED DATA
6311	006106	112710	000240		MOVB	#MISR,(R0)		:LOAD MODE TO READ ISR
6312	006112	111137	002126		MOVB	(R1),\$BDDAT		
6313	006116	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6314	006124	001405			BEQ	5\$		
6315	006126	004537	011326		JSR	R5,FRMBUF		:RESTORE 0-200
6316	006132	104006			ERROR	6		:ISR ERROR
6317	006134	004537	011756		JSR	R5,CAT200		:RESTORE TRAP CATCHER
6318	006140	116437	000001	002124	5\$: MOVB	1(R4),\$GDDAT		:STORE EXPECTED
6319	006146	112710	000244		MOVB	#MIMR,(R0)		:LOAD MODE FOR IMR
6320	006152	111137	002126		MOVB	(R1),\$BDDAT		
6321	006156	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6322	006164	001405			BEQ	6\$		
6323	006166	004537	011326		JSR	R5,FRMBUF		:RESTORE 0-202
6324	006172	104005			ERROR	5		:IMR ERROR
6325	006174	004537	011756		JSR	R5,CAT200		:RESTORE CATCHER
6326	006200	005037	002124		6\$: CLR	\$GDDAT		
6327	006204	112710	000250		MOVB	#MIRR,(R0)		:LOAD MODE BITS FOR IRR
6328	006210	111137	002126		MOVB	(R1),\$BDDAT		
6329	006214	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6330	006222	001405			BEQ	7\$		
6331	006224	004537	011326		JSR	R5,FRMBUF		:RESTORE 0-202
6332	006230	104003			ERROR	3		:IRR ERROR
6333	006232	004537	011756		JSR	R5,CAT200		:RESTORE TRAP CATCHER
6334	006236	005037	002124		7\$: CLR	\$GDDAT		
6335	006242	112710	000254		MOVB	#MACR,(R0)		
6336	006246	111137	002126		MOVB	(R1),\$BDDAT		
6337	006252	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6338	006260	001405			BEQ	10\$		
6339	006262	004537	011326		JSR	R5,FRMBUF		:RESTORE 0-202
6340	006266	104004			ERROR	4		:ACR ERROR
6341	006270	004537	011756		JSR	R5,CAT200		:RESTORE TRAP CATCHER
6342	006274	005037	002124		10\$: CLR	\$GDDAT		
6343	006300	112710	000160		MOVB	#CISR,(R0)		:CLEAR ISR
6344	006304	112710	000240		MOVB	#MISR,(R0)		:LOAD MODE TO READ ISR
6345	006310	111137	002126		MOVB	(R1),\$BDDAT		
6346	006314	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6347	006322	001405			BEQ	11\$		

6348	006324	004537	011326		JSR	R5,FRMBUF	:RESTORE 0-202
6349	006330	104006			ERROR	6	:ISR REG ERROR
6350	006332	004537	011756		JSR	R5,CAT200	:RESTORE TRAP CATCHER
6351	006336	004537	011714	11\$:	JSR	R5,RES200	:RESTORE VECTOR JUST TESTED
6352	006342	004537	011326		JSR	R5,FRMBUF	:RESTORE 0-200 TO SYSMAC CONTROL
6353	006346	022737	000204	002452	CMP	#204,VECLOC	:FINISHED?
6354	006354	001402			BEQ	12\$	
6355	006356	000137	005574		JMP	112\$:TEST NEXT VECTOR ADDR
6356	006362	005724		12\$:	TST	(R4)+	:INDEX EXPECTED ISR AND IMR PATTERN
6357	006364	005237	002442		INC	IMRLOC	:STORE CLEAR NEXT IMR BIT
6358	006370	005237	002446		INC	IRRLOC	:STORE SET NEXT IRR BIT
6359	006374	005237	002456		INC	VECVL	:SETUP TO TEST NEXT VECTOR LEVEL
6360	006400	005337	002466		DEC	LVLcnt	:FINISHED LEVELS 0-7?
6361	006404	001402			BEQ	13\$:YES,CHECK BYTE COUNT END
6362	006406	000137	005570		JMP	111\$:NO,DO NEXT LEVEL,SAME BYTE COUNT
6363	006412	005237	002472	13\$:	INC	BYNUM	:INDEX BYTE COUNT
6364	006416	104407			CKSWR		:LOOK FOR SWITCH CHANGE
6365	006420	022737	000400	002456	CMP	#400,VECVL	:FINISHED ALL FOUR BYTE COUNTS
6366	006426	001402			BEQ	14\$:YES,FINISHED GROUP
6367	006430	000137	005534		JMP	121\$:NO,DO NEXT BYTE COUNT WITH
6368							:LEVELS 0-7.
6369	006434	005337	002462	14\$:	DEC	GRPCNT	:FINISHED BOTH GROUP 1 AND GROUP 2?
6370	006440	001406			BEQ	15\$:YES,BOTH GROUPS TESTED
6371	006442	013700	002422		MOV	DRCSC,R0	:NO,TEST GROUP 2
6372							:CSRC = CONTROL GROUP 2
6373	006446	013701	002426		MOV	DRCSD,R1	:CSRD = DATA GROUP 2
6374	006452	000137	005514		JMP	120\$:RETURN AND TEST GROUP2
6375							:FOR BYTE COUNTS 0-3 AND
6376							:LEVELS 0-7 FOR EACH BYTE COUNT.
6377	006456	004537	011326	15\$:	JSR	R5,FRMBUF	:RESTORE 0-202,EXIT TEST
6378							
6379							
6380							
(3)							:*****
(3)							:*TEST 5 TEST VECTOR ADDR UNIQUENESS IN FIXED MODE FOR GRPS 1,2
(2)	006462	000004					:*****
(2)							TST5: SCOPE
6381							:BYTE COUNT = 4
6382							:LEVELS = 0-7
6383	006464	004537	010746		JSR	R5,CLRCSR	:CLEAR ALL CSRS
6384	006470	004537	011266		JSR	R5,CLRBF	:CLEAR VECTOR BUFFER AREA
6385	006474	004537	011456		JSR	R5,TRPCAT	:RESTORE TRAP CATCHER
6386	006500	004537	011232		JSR	R5,STRVEC	:STORE VECTOR AREA 300-674
6387							:WITH INT. SERV ROUTINES
6388	006504	012737	000377	002456	MOV	#377,VECVL	:VECTOR MEM FINAL VALUE
6389	006512	004537	011026		JSR	R5,CLRIRR	:CLEAR IRR REGS
6390	006516	012704	017030		MOV	#VBUF,R4	:VECTOR ADDRESS BUFFER
6391	006522	004537	011064		JSR	R5,VECFIL	:FILL VECTOR MEMORY FOR GROUPS 1,2
6392							:WITH VECTORS 300-674
6393	006526	012705	012142		MOV	#SETVEC,R5	: R5 POINTS TO COMMON HANDLER ;;GPA
6394	006532	013700	002412		MOV	DRCSA,R0	:GROUP 1 CSR
6395	006536	013701	002422		MOV	DRCSC,R1	:GROUP 2 CSR
6396	006542	013702	002416		MOV	DRCSE,R2	
6397	006546	013703	002426		MOV	DRCSD,R3	
6398	006552	010037	002122		MOV	R0,\$BDADR	:STORE BUS ADDRESS
6399	006556	112710	000300		MOVB	#PACR,(R0)	:PRESELECT GP1 ACR FOR WRITING

CNDRDA DRV11J DIAG TST PRT2
CNDRDA.P11 10-DEC-82 14:44

MACY11 30(1046)
T5

15-DEC-82 15:26 PAGE 58-9
TEST VECTOR ADDR UNIQUENESS IN FIXED MODE FOR GRPS 1,2

SEQ 0023

6400	006562	112712	000377		MOV B	#377,(R2)	;SET ALL ACR BITS GP1	
6401	006566	112710	000040		MOV B	#CIMR,(R0)	;CLEAR ALL IMR BITS GP1	
6402	006572	005037	002436		CLR	INTFLG	;CLEAR INT. FLAG	
6403	006576	106427	000000		MTPS	#PRO		
6404	006602	112710	000120		MOV B	#SIRR,(R0)	;SET ALL IRR BITS	
6405	006606	112710	000241		MOV B	#241,(R0)	;LMD57 - ARM THE CHIP	
6406	006612	106427	000300		MTPS	#300	;VER:0	
6407	006616	005737	002436		TST	INTFLG	;CHECK FOR NO INTERRUPTS	
6408	006622	001401			BEQ	1\$		
6409	006624	104011			ERROR	11	;ILLEGAL INTERRUPT	
6410	006626	005037	002436	1\$:	CLR	INTFLG		
6411	006632	112711	000300		MOV B	#PACR,(R1)	;PRESELECT GROUP2 FOR WRITING	
6412	006636	112713	000377		MOV B	#377,(R3)	;ALL ACR BITS SET GP2	
6413	006642	112711	000040		MOV B	#CIMR,(R1)	;CLEAR ALL IMR BITS GP2	
6414	006646	106427	000000		MTPS	#PRO		
6415	006652	112711	000120		MOV B	#SIRR,(R1)	;SET ALL IRR BITS	
6416	006656	112711	000241		MOV B	#241,(R1)	;LMD57 - ARM THE CHIP	
6417	006662	106427	000300		MTPS	#PR6 ;VER:0		
6418	006666	005737	002436		TST	INTFLG	;CHECK FOR NO INTERRUPTS	
6419	006672	001401			BEQ	2\$		
6420	006674	104011			ERROR	11	;ILLEGAL INTERRUPT	
6421	006676	005037	002436	2\$:	CLR	INTFLG		
6422	006702	106427	000000		MTPS	#PRO		
6423	006706	152760	000002	000001	BISB	#BIT1,1(R0)	;SET INT. ENABLE,EXPECT INTERRUPTS	
6424	006714	012737	000100	002124	MOV	#64,\$GDDAT	;EXPECT 64 INTERRUPTS	
6425	006722	013737	002436	002126	MOV	INTFLG,\$BDDAT	;SHOULD HAVE 64 INTERRUPTS	
6426	006730	106427	000300		MTPS	#PR6 ;VER:0		
6427	006734	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6428	006742	001401			BEQ	3\$		
6429	006744	104012			ERROR	12	;INTERRUPT TEST ERROR	
6430	006746	012737	101710	002124	3\$:	MOV	#101710,\$GDDAT	;RDY,DIR,I/E,CHIP - 31X
6431	006754	010037	002122		MOV	R0,\$BDADR	;CSRA	
6432	006760	011037	002126		MOV	(R0),\$BDDAT		
6433	006764	042737	000007	002126	BIC	#7,\$BDDAT	;CLEAR UNDEFINED BITS	
6434	006772	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6435	007000	001401			BEQ	4\$		
6436	007002	104002			ERROR	2	;CSRA REG ERROR	
6437	007004	012737	000310	002124	4\$:	MOV	#310,\$GDDAT	;STORE EXPECTED
6438	007012	010137	002122		MOV	R1,\$BDADR	;CSRC	
6439	007016	011137	002126		MOV	(R1),\$BDDAT		
6440	007022	042737	000007	002126	BIC	#7,\$BDDAT	;CLEAR UNDEFINED BITS	
6441	007030	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6442	007036	001401			BEQ	5\$		
6443	007040	104002			ERROR	2	;CSRC ERROR	
6444	007042	010237	002122	5\$:	MOV	R2,\$BDADR	;CSRB ADDRESS	
6445	007046	005037	002126		CLR	\$BDDAT		
6446	007052	005037	002124		CLR	\$GDDAT		
6447	007056	112710	000240		MOV B	#MISR,(R0)	;READ ISR	
6448	007062	111237	002126		MOV B	(R2),\$BDDAT		
6449	007066	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6450	007074	001401			BEQ	6\$		
6451	007076	104006			ERROR	6	;ISR ERROR,GP1	
6452	007100	112710	000244	6\$:	MOV B	#MIMR,(R0)	;MODE BITS FOR IMR	
6453	007104	111237	002126		MOV B	(R2),\$BDDAT		
6454	007110	023737	002124	002126	CMP	\$GDDAT,\$BDDAT		
6455	007116	001401			BEQ	7\$		

6456	007120	104005			ERROR	5		;IMR ERROR,GP1	
6457	007122	112710	000250		7\$:	MOVB	#MIRR,(R0)	;MODE BITS FOR IRR	
6458	007126	111237	002126			MOVB	(R2), \$BDDAT		
6459	007132	023737	002124	002126		CMP	\$GDDAT,\$BDDAT		
6460	007140	001401				BEQ	10\$		
6461	007142	104003				ERROR	3	;IRR ERROR	
6462	007144	012737	000377	002124	10\$:	MOV	#377,\$GDDAT	;EXPECTED ACR	
6463	007152	112710	000254			MOVB	#MACR,(R0)	;READ ACR	
6464	007156	111237	002126			MOVB	(R2), \$BDDAT		
6465	007162	023737	002124	002126		CMP	\$GDDAT,\$BDDAT		
6466	007170	001401				BEQ	11\$		
6467	007172	104004				ERROR	4	;ACR ERROR	
6468	007174	005037	002124		11\$:	CLR	\$GDDAT		
6469	007200	005037	002126			CLR	\$BDDAT		
6470	007204	010337	002122			MOV	R3,\$BDADR	;CSR ADDRESS	
6471	007210	112711	000240			MOVB	#MISR,(R1)	;READ ISR ,GP2	
6472	007214	111337	002126			MOVB	(R3), \$BDDAT		
6473	007220	023737	002124	002126		CMP	\$GDDAT,\$BDDAT		
6474	007226	001401				BEQ	12\$		
6475	007230	104006				ERROR	6	;ISR ERROR,GP1	
6476	007232	112711	000244		12\$:	MOVB	#MIMR,(R1)	;READ IMR GP2	
6477	007236	111337	002126			MOVB	(R3), \$BDDAT		
6478	007242	023737	002124	002126		CMP	\$GDDAT,\$BDDAT		
6479	007250	001401				BEQ	13\$		
6480	007252	104005				ERROR	5	;IMR ERROR,GP2	
6481									
6482	007254	112711	000250		13\$:	MOVB	#MIRR,(R1)	;READ IRR	
6483	007260	111337	002126			MOVB	(R3), \$BDDAT		
6484	007264	023737	002124	002126		CMP	\$GDDAT,\$BDDAT		
6485	007272	001401				BEQ	14\$		
6486	007274	104003				ERROR	3	;IRR ERROR,GP2	
6487	007276	112711	000254		14\$:	MOVB	#MACR,(R1)	;READ ACR	
6488	007302	012737	000377	002124		MOV	#377,\$GDDAT		
6489	007310	111337	002126			MOVB	(R3), \$BDDAT		
6490	007314	023737	002124	002126		CMP	\$GDDAT,\$BDDAT		
6491	007322	001401				BEQ	15\$		
6492	007324	104004				ERROR	4	;ACR ERROR,GP2	
6493	007326	012702	000100		15\$:	MOV	#64,R2		
6494	007332	012704	017030			MOV	#VBUF,R4	;VECTOR BUFFER	
6495	007336	012737	000300	002124		MOV	#300,\$GDDAT	;START WITH FIRST VECTOR 300	
6496	007344	010437	002122		16\$:	MOV	R4,\$BDADR	;VECTOR BUFFER ADDRESS	
6497	007350	011437	002126			MOV	(R4), \$BDDAT		
6498	007354	023737	002124	002126		CMP	\$GDDAT,\$BDDAT		
6499	007362	001401				BEQ	17\$;VECTORED PROPERLY	
6500	007364	104007				ERROR	7	;VECTOR ADDR MEM ERROR	
6501	007366	062737	000004	002124	17\$:	ADD	#4,\$GDDAT		
6502	007374	005737	017302			TST	KXTFLAG	; KXT11 ??	::GPA
6503	007400	001407				BEQ	1000\$; BR IF NGT	::GPA
6504	007402	023727	002124	000400		CMP	\$GDDAT,#400	; YES, REACHED VECTOR 400 ??	::GPA
6505	007410	103403				BLO	1000\$; NOT YET, CONTINUE	::GPA
6506	007412	012737	000300	002124		MOV	#300,\$GDDAT	; YES, RESET EXPECTED TO 300	::GPA
6507	007420				1000\$:				::GPA
6508	007420	062704	000002			ADD	#2,R4	;NEXT BUFFER ADDRESS	
6509	007424	005302				DEC	R2	;COMPLETED 64 BUFFER CHECKS?	
6510	007426	001346				BNE	16\$;CHECK NEXT VECTOR BUFFER LOCATION	
6511								;THAT WAS STORED BY THE INTERRUPT	

```
6512 ;SERVICE ROUTINES.
6513
6514 ;*****
(3) ;*TEST 6 TEST VECTOR ADDR UNIQUENESS IN ROTATING MODE FOR GRPS 1,2
(3) ;*****
(2) 007430 000004 TST6: SCOPE
(2)
6515 ;BYTE COUNT = 4
6516 ;LEVELS = 0-7
6517 007432 004537 010746 JSR R5,CLRCR ;CLEAR ALL CSRS
6518 007436 004537 011266 JSR R5,CLRBF ;CLEAR VECTOR BUFFER AREA
6519 007442 004537 011456 JSR R5,TRPCAT ;RESTORE TRAP CATCHER
6520 007446 004537 011232 JSR R5,STRVEC ;STORE VECTOR AREA 300-674
6521 ;WITH INT. SERV ROUTINES
6522 007452 012737 000377 002456 MOV #377,VECVL ;VECTOR MEM FINAL VALUE
6523 007460 004537 011026 JSR R5,CLRIRR ;CLEAR IRR REGS
6524 007464 012704 017030 MOV #VBUF,R4 ;VECTOR ADDRESS BUFFER
6525 007470 004537 011064 JSR R5,VECFIL ;FILL VECTOR MEMORY FOR GROUPS 1,2
6526 ;WITH VECTORS 300-674
6527 007474 012705 012142 MOV #SETVEC,R5 ; R5 POINTS TO COMMON HANDLER. ;:GPA
6528 007500 013700 002412 MOV DRCSA,R0 ;GROUP 1 CSR
6529 007504 013701 002422 MOV DRCSA,R1 ;GROUP 2 CSR
6530 007510 013702 002416 MOV DRCSB,R2
6531 007514 013703 002426 MOV DRCSB,R3
6532 007520 010037 002122 MOV R0,$BDDADR ;STORE BUS ADDRESS
6533 007524 012737 000010 002474 MOV #8,LVLSAV ;COUNTER FOR 7 LEVELS
6534 007532 112710 000201 MOV #201,(R0) ;ROTATING PRIORITY GROUP 1
6535 007536 112710 000040 MOV #CIMR,(R0) ;CLEAR ALL IMR BITS GP1
6536 007542 005037 002436 CLR INTFLG ;CLEAR INT. FLAG
6537 007546 106427 000000 MTPS #PRO
6538 007552 112710 000120 MOV #SIRR,(R0) ;SET ALL IRR BITS
6539 007556 112710 000241 MOV #241,(R0) ;LMD57 - ARM THE CHIP
6540 007562 106427 000300 MTPS #300 ;VER:0
6541 007566 005737 002436 TST INTFLG ;CHECK FOR NO INTERRUPTS
6542 007572 001401 BEQ 1$
6543 007574 104011 ERROR 11 ;ILLEGAL INTERRUPT
6544 007576 005037 002436 1$: CLR INTFLG
6545 007602 112711 000201 MOV #201,(R1) ;ROTATING PRIORITY FOR GROUP 2
6546 007606 112711 000040 MOV #CIMR,(R1) ;CLEAR ALL IMR BITS GP2
6547 007612 106427 000000 MTPS #PRO
6548 007616 112711 000120 MOV #SIRR,(R1) ;SET ALL IRR BITS
6549 007622 112711 000241 MOV #241,(R1) ;LMD57 - ARM THE CHIP
6550 007626 106427 000300 MTPS #PR6 ;VER:0
6551 007632 005737 002436 TST INTFLG ;CHECK FOR NO INTERRUPTS
6552 007636 001401 BEQ 2$
6553 007640 104011 ERROR 11 ;ILLEGAL INTERRUPT
6554 007642 005037 002436 2$: CLR INTFLG
6555 007646 106427 000000 MTPS #PRO
6556 007652 152760 000002 000001 BISB #BIT1,1(R0) ;SET INT. ENABLE,EXPECT INTERRUPTS
6557 007660 012737 000004 002124 MOV #4,$GDDAT ;EXPECT 4 INTERRUPTS
6558 007666 013737 002436 002126 MOV INTFLG,$BDDAT ;SHOULD HAVE 4 INTERRUPTS
6559 007674 106427 000300 MTPS #PR6 ;VER:0
6560 007700 023737 002124 002126 CMP $GDDAT,$BDDAT
6561 007706 001401 BEQ 100$
6562 007710 104012 ERROR 12 ;INTERRUPT TEST ERROR
6563 007712 112710 000140 100$: MOV #CHPISR,(R0) ;CLEAR HIGHEST PRIORITY ISR
```

```

6564 007716 112710 000120      MOVB    #SIRR,(R0)      ;SET ALL IRR BITS
6565 007722 005337 002474      DEC     LVLSAV          ;DONE 8 LEVELS,GROUP 1
6566 007726 001345                BNE     2$              ;DO NEXT LEVEL IN ROTATION
6567 007730 112710 000100      MOVB    #CIRR,(R0)     ;CLEAR IRR BITS GROUP 1
6568 007734 012737 000010 002474  MOV     #8.,LVLSAV     ;DO 8 LEVELS,GROUP 2
6569 007742 005037 002436      101$:  CLR     INTFLG        ;CLEAR INT FLAG
6570 007746 106427 000000      MTPS    #PRO
6571 007752 012737 000004 002124  MOV     #4,$GDDAT
6572 007760 013737 002436 002126  MOV     INTFLG,$BDDAT ;SHOULD HAVE FOUR INTERRUPTS
6573 007766 106427 000300      MTPS    #PR6 ;VER:0
6574 007772 023737 002124 002126  CMP     $GDDAT,$BDDAT
6575 010000 001401                BEQ     102$
6576 010002 104012                ERROR   12              ;INTERRUPT TEST ERROR
6577 010004 112711 000140 102$:  MOVB    #CHPISR,(R1)   ;CLEAR HIGHEST PRIORITY ISR
6578 010010 112711 000120      MOVB    #SIRR,(R1)     ;SET ALL IRR BITS
6579 010014 005337 002474      DEC     LVLSAV          ;DONE 8 LEVELS?
6580 010020 001350                BNE     101$           ;DO NEXT LEVEL IN ROTATION,GP2
6581 010022 112711 000100      MOVB    #CIRR,(R1)     ;CLEAR IRR BITS,GROUP 2
6582 010026 012737 101750 002124  3$:  MOV     #101750,$GDDAT ;RDY,DIR,I/E,CHIP - 35X
6583 010034 010037 002122      MOV     R0,$BDADR      ;CSRA
6584 010040 011037 002126      MOV     (R0),$BDDAT
6585 010044 042737 000007 002126  BIC     #7,$BDDAT      ;CLEAR UNDEFINED BITS
6586 010052 023737 002124 002126  CMP     $GDDAT,$BDDAT
6587 010060 001401                BEQ     4$
6588 010062 104002                ERROR   2              ;CSRA REG ERROR
6589 010064 012737 000350 002124  4$:  MOV     #350,$GDDAT   ;STORE EXPECTED
6590 010072 010137 002122      MOV     R1,$BDADR      ;CSRC
6591 010076 011137 002126      MOV     (R1),$BDDAT
6592 010102 042737 000007 002126  BIC     #7,$BDDAT      ;CLEAR UNDEFINED BITS
6593 010110 023737 002124 002126  CMP     $GDDAT,$BDDAT
6594 010116 001401                BEQ     5$
6595 010120 104002                ERROR   2              ;CSRC ERROR
6596 010122 010237 002122 5$:  MOV     R2,$BDADR      ;CSRB ADDRESS
6597 010126 005037 002126      CLR     $BDDAT
6598 010132 005037 002124      CLR     $GDDAT
6599 010136 112710 000240      MOVB    #MISR,(R0)     ;READ ISR
6600 010142 111237 002126      MOVB    (R2),$BDDAT
6601 010146 023737 002124 002126  CMP     $GDDAT,$BDDAT
6602 010154 001401                BEQ     6$
6603 010156 104006                ERROR   6              ;ISR ERROR,GP1
6604 010160 112710 000244 6$:  MOVB    #MIMR,(R0)     ;MODE BITS FOR IMR
6605 010164 111237 002126      MOVB    (R2),$BDDAT
6606 010170 023737 002124 002126  CMP     $GDDAT,$BDDAT
6607 010176 001401                BEQ     7$
6608 010200 104005                ERROR   5              ;IMR ERROR,GP1
6609 010202 112710 000250 7$:  MOVB    #MIRR,(R0)     ;MODE BITS FOR IRR
6610 010206 111237 002126      MOVB    (R2),$BDDAT
6611 010212 023737 002124 002126  CMP     $GDDAT,$BDDAT
6612 010220 001401                BEQ     10$
6613 010222 104003                ERROR   3              ;IRR ERROR
6614 010224 112710 000254 10$:  MOVB    #MACR,(R0)     ;READ ACR
6615 010230 111237 002126      MOVB    (R2),$BDDAT
6616 010234 023737 002124 002126  CMP     $GDDAT,$BDDAT
6617 010242 001401                BEQ     11$
6618 010244 104004                ERROR   4              ;ACR ERROR
6619 010246 010337 002122 11$:  MOV     R3,$BDADR      ;CSRD ADDRESS
  
```


6676
6677
6678
6679
6680 010564 013701 002412
6681 010570 000241
6682 010572 006037 002434
6683 010576 001412
6684 010600 162701 000020
6685 010604 005237 002202
6686 010610 032737 000001 002434
6687 010616 001764
6688 010620 000137 003132
6689
6690
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 010624
(2) 010624 000240
(1) 010626 005037 002102
(1) 010632 005037 002160
(1) 010636 005237 002176
(1) 010642 042737 100000 002176
(1) 010650 005327
(1) 010652 000001
(1) 010654 003022
(1) 010656 012737
(1) 010660 000001
(1) 010662 010652
(1) 010664 104401 010731
(2) 010670 013746 002176
(2) 010674 104405
(1) 010676 104401 010726
(1) 010702 013700 000042
(1) 010706 001405
(1) 010710 000005
(1) 010712 004710
(1) 010714 000240
(1) 010716 000240
(1) 010720 000240
(1) 010722
(1) 010722 000137
(1) 010724 003066
(1)
(1)
(1) 010726 377 377 000
(1) 010731 015 042412 042116
(1) 010736 050040 051501 020123
(1) 010744 000043

;SERVICE ROUTINES.
;DON'T REPORT 'END OF PASS' UNTIL ALL SELECTED DRV11J'S HAVE BEEN TESTED.
NXDEV: MOV DRCSA,R1 ;INIT TO SETUP DRV11J ADDRESS
NXDEV1: CLC ;CLEAR CARRY FOR DEVICE MAP
ROR DMAP ;LOOK FOR NEXT DRV11J
BEQ \$EOP ;BR IF ALL TESTED
SUB #20,R1 ;NEXT DRV11J STARTS -20 FROM PAST CSR
INC \$UNIT ;UPDAT UNIT #
BIT #1,DMAP ;IS UNIT SELECTED?
BEQ NXDEV1 ;BR IF NOT
JMP NEXPAS ;TEST NEXT DRV11J
;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 \$TSTNM ;:ZERO THE TEST NUMBER
CLR \$TIMES ;:ZERO THE NUMBER OF ITERATIONS
INC \$PASS ;:INCREMENT THE PASS NUMBER
BIC #100000,\$PASS ;:DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;:LOOP?
\$EOPCT: .WORD 1
BGT \$DOAGN ;:YES
MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
\$ENDCT: .WORD 1
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
\$SENDMG: .ASCIIZ <15><12>/END PASS #/

6692
6693
6694
6695
6696
6697
6698
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6740
6741
6742
6743
6744
6745
6746
6747

010746 010046
010750 106427 000300
010754 012737 000340 002456
010762 012737 000300 002452
010770 013700 002412
010774 005037 002124
011000 005037 002126
011004 005010
011006 105060 000005
011012 005060 000010
011016 105060 000015
011022 012600
011024 000205

011026 010046
011030 013700 002412
011034 105060 000011
011040 112760 000001 000001
011046 005060 000002
011052 105010
011054 105060 000010
011060 012600
011062 000205

011064 013700 002412
011070 013701 002416
011074 012702 000002
011100 012737 000060 002454
011106 012737 000370 002456
011114 012737 000010 002466
011122 113710 002456
011126 012737 000004 002470
011134 113711 002454
011140 005237 002454
011144 005737 017302
011150 001407
011152 023727 002454 000100
011160 103403
011162 012737 000060 002454

011170
011170 005337 002470
011174 001357
011176 005337 002466

.SBTTL PROGRAM SUBROUTINES

:CLEAR ALL CONTROL/STATUS REGISTERS

```
CLRCSR: MOV R0,-(SP) ;SAVE R0
MTPS #PR6 ;VER:0 ;PSW = 300
MOV #PUMA,VECVAL ;INIT VECTOR ADDR MEM = 340
MOV #300,VEVCLOC ;INIT VECTOR ADDR TO 300
MOV DRCSA,R0 ;START OF CSR ADDRESS
CLR $GDDAT ;CLEAR EXPECTED
CLR $BDDAT ;CLEAR REC'D
CLR (R0) ;CLEAR CSRA;CHIP RESET GROUP 1
CLRB 5(R0) ;CLEAR HIGH BYTE CSRB
CLR 10(R0) ;CLEAR CSRC;CHIP RESET GROUP 2
CLRB 15(R0) ;CLEAR HIGH BYTE CSRD
MOV (SP)+,R0 ;RETURN R0
RTS R5
```

:CLEAR IRR REGISTERS, GROUP 1, GROUP 2 WITH CHIP RESET

```
CLRIRR: MOV R0,-(SP)
MOV DRCSA,R0 ;START OF CSR ADDRESS
CLRB 11(R0) ;CSRC TO INPUT MODE
MOVB #BIT0,1(R0) ;CSRA TO OUTPUT MODE
CLR 2(R0) ;CLEAR DBRA
CLRB (R0) ;CHIP RESET OF GROUP1
CLRB 10(R0) ;CHIP RESET OF GROUP 2
MOV (SP)+,R0 ;RESTORE REGISTER
RTS R5 ;EXIT
```

:ROUTINE TO FILL VECTOR MEMORY FOR VECTOR UNIQUENESS TEST

```
:300-474 GROUP 1
:500-674 GROUP 2
VECFIL: MOV DRCSA,R0 ;START GROUP 1
MOV DRCSB,R1
MOV #2,R2 ;TWO GROUPS
MOV #60,VECPAT ;VECTOR START 300
1$: MOV #370,VECVAL ;START VECTOR LEVEL 0,BY4
MOV #8.,LVLCNT ;LEVELS 0-7
2$: MOVB VECVAL,(R0)
MOV #4,BYCNT ;DO FOUR BYTE COUNTS
3$: MOVB VECPAT,(R1) ;WRITE VECTOR
INC VECPAT ;SETUP FOR NEXT VECTOR
TST KXTFLAG ;KXT11 ??
BEQ 1000$ ;BR IF NOT ;:GPA
CMP VECPAT,#100 ;YES, REACHED VECTOR 400 ?? ;:GPA
BLO 1000$ ;NOT YET, CONTINUE ;:GPA
MOV #60,VECPAT ;YES, WRAP-AROUND TO 300 AGAIN ;:GPA
;WE'LL END UP WITH 300-374... ;:GPA
;...REPLICATED 4 TIMES. ;:GPA

1000$: DEC BYCNT ;DONE FOUR BYTE COUNTS?
BNE 3$ ;DO NEXT VECTOR
4$: DEC LVLCNT ;DONE ALL LEVELS IN GROUP? ;:GPA
```

```

6748 011202 001403          BEQ      5$
6749 011204 005237 002456  INC      VECVAL      ;INC NEXT LEVEL
6750 011210 000744          BR       2$           ;DO NEXT LEVEL
6751 011212 005302          5$: DEC      R2           ;DONE BOTH GROUPS?
6752 011214 001405          BEQ      6$           ;DONE,EXIT
6753 011216 013700 002422  MOV      DRCSC,R0     ;DO GROUP 2
6754 011222 013701 002426  MOV      DRCSD,R1
6755 011226 000727          BR       1$
6756 011230 000205          6$: RTS      R5           ;EXIT
6757
6758          ;ROUTINE TO STORE VECTOR AREA 300-674 WITH INTERRUPT SERVICE
6759 011232 012700 000300  STRVEC: MOV      #300,R0      ;START AT VECTOR 300
6760 011236 012701 012170  MOV      #INT0,R1         ;FIRST SERVICE ROUTINE
6761 011242 012702 000100  MOV      #64.,R2         ;STORE 64 SERVICE ROUTINES
6762 011246 010120          1$: MOV      R1,(R0)+
6763 011250 012720 000300  MOV      #300,(R0)+
6764          ;;GPA ADD      #6,R1           ;VER:0
6765 011254 062701 000004  ADD      #4,R1           ;SETUP FOR NEXT VECTOR STORAGE
6766 011260 005302          ADD      #4,R1           ; SETUP NEXT VETOR.
6767 011262 001371          DEC      R2           ;DONE WITH 64 VECTOR ROUTINES?
6768 011264 000205          BNE     1$
6769          RTS      R5           ;EXIT
6770          ;ROUTINE TO CLEAR VECTOR BUFFER
6771 011266 012700 017030  CLRBF: MOV      #VBUF,R0     ;VECTOR BUFFER START
6772 011272 012701 000100  MOV      #64.,R1
6773 011276 005020          1$: CLR      (R0)+
6774 011300 005301          DEC      R1
6775 011302 001375          BNE     1$
6776 011304 000205          RTS      R5
6777
6778          ;ROUTINE TO STORE 0-202 INTO BUFFER AREA TO ALLOW
6779          ;VECTOR TESTS TO 0-200.
6780 011306 005000          TOBUF: CLR      R0
6781 011310 012701 016624  MOV      #DBUF,R1         ;BUFFER START
6782 011314 012021          1$: MOV      (R0)+,(R1)+     ;STORE 0-202 INTO BUFFER
6783 011316 022700 000204  CMP      #204,R0         ;FINISHED?
6784 011322 001374          BNE     1$             ;NO
6785 011324 000205          RTS      R5           ;RETURN
6786
6787          ;STORE BUFFER AREA INTO LOCATIONS 0-202
6788          ;AFTER VECTOR TESTS
6789 011326 010046          FRMBUF: MOV      R0,-(SP)     ;SAVE R0
6790 011330 010146          MOV      R1,-(SP)
6791 011332 005000          CLR      R0
6792 011334 012701 016624  MOV      #DBUF,R1         ;BUFFER START
6793 011340 012120          1$: MOV      (R1)+,(R0)+     ;STORE BUFFER INTO LOCS 0-202
6794 011342 022700 000204  CMP      #204,R0         ;FINISHED?
6795 011346 001374          BNE     1$             ;NO
6796 011350 013777 002460 170562  MOV      SWRSV,@SWR       ;STORE LOCATION 176,SOF1 SWR
6797 011356 012601          MOV      (SP)+,R1        ;RESTORE R1
6798 011360 012600          MOV      (SP)+,R0        ;RESTORE R0
6799 011362 000205          RTS      R5           ;RETURN
6800
6801          ;RESTORE VECTOR JUST TESTED TO ORIGINAL TRAP FOR 204-1774
6802 011364 022737 000001 002210  RESTRP: CMP      #1,$ENV     ;CHECK FOR APT
6803 011372 001421          BEQ     1$             ;YES,BRANCH

```

```

6804 011374 032777 010000 170536 BIT #SW12,@SWR ;CHECK TYPE OF TRAP RETURN
6805 011402 001412 BEQ 10$ ;RESTORE ILLEGAL VECTOR ROUTINE
6806 011404 062737 000002 002452 ADD #2,VECLOC ;RESTORE TRAP
6807 011412 013723 002452 MOV VECLOC,(R3)+ ;TO VECTOR
6808 011416 005013 CLR (R3) ;RESTORE HALT
6809 011420 062737 000002 002452 ADD #2,VECLOC ;SETUP FOR NEXT
6810 011426 000412 BR 2$ ;RETURN
6811 011430 012723 011704 10$: MOV #TRPALL,(R3)+ ;RESTORE ILLEGAL TRAP ROUTINE
6812 011434 000402 BR 11$ ;RESTORE PSW
6813 011436 012723 011666 1$: MOV #TRPOUT,(R3)+ ;RESTORE APT SUB TRAP
6814 ;IF ON APT
6815 011442 012713 000300 11$: MOV #300,(R3) ;RESTORE PSW SAVE :VER:0
6816 011446 062737 000004 002452 ADD #4,VECLOC ;UPDATE FOR NEXT VECTOR
6817 011454 000205 2$: RTS R5 ;RETURN
6818
6819 ;ROUTINE TO SET UP TRAP CATCHER 204-1774 IN STANDALONE MODE OR
6820 ;A COMMON VECTOR ROUTINE SETUP UNDER APT.
6821 ;VECTOR ROUTINE IS USED ONLY IN AN APT ENVIRONMENT AND WILL STORE
6822 ;A COMMON SUBROUTINE IN VECTOR AREA 204-1774.
6823
6824 011456 010046 TRPCAT: MOV R0,-(SP) ;SAVE R0
6825 011460 010146 MOV R1,-(SP)
6826 011462 022737 000001 002210 CMP #1,$ENV ;CHECK FOR APT
6827 011470 001426 BEQ APTVEC ;YES,SET UP TRAPS FOR APT
6828 011472 012700 000204 MOV #204,R0 ;START CATCHER AT 204
6829 011476 032777 010000 170434 BIT #SW12,@SWR ;TEST IF SW12 IS SET
6830 011504 000101 BNE 1$ ;YES,BR AND STORE REGULAR TRAP
6831 ;CATCHER(+2,HALT) IN LOCATIONS
6832 ;204-1774
6833 011506 012720 011704 11$: MOV #TRPALL,(R0)+ ;NO,PLACE ILLEGAL VECTOR RETURN
6834 011512 012720 000300 MOV #300,(R0)+ ;ROUTINES IN LOCATIONS 204-1774;VER:0
6835 011516 022700 002000 CMP #2000,R0 ;STORED ALL VECTORS?
6836 011522 001371 BNE 11$
6837 011524 000421 BR ENDTRP ;RETURN AFTER VECTOR STORAGE
6838 011526 010001 1$: MOV R0,R1 ;DRV11J CAN VECTOR 0-1774
6839 011530 005721 TST (R1)+
6840 011532 010120 MOV R1,(R0)+
6841 011534 005020 CLR (R0)+
6842 011536 022701 001776 CMP #1776,R1 ;CHECK FOR VECTOR END
6843 011542 001371 BNE 1$
6844 011544 000411 BR ENDTRP ;RETURN IN STAND ALONE MODE
6845 011545 012701 000204 APTVEC: MOV #204,R1 ;STARTING VECTOR ADDRESS
6846 011552 012721 011666 1$: MOV #TRPOUT,(R1)+ ;ILL INTERRUPT ROUTINE
6847 011556 012721 000300 MOV #300,(R1)+ ;PSW NEXT;VER:0
6848 011562 022701 002000 CMP #2000,R1 ;DONE?
6849 011566 001371 BNE 1$ ;DO NEXT VECTOR
6850 011570 012601 ENDTRP: MOV (SP)+,R1
6851 011572 012600 MOV (SP)+,R0
6852 011574 000205 RTS R5 ;EXIT
6853
6854 ;INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 204-1774
6855 011576 005237 002436 INTSR1: INC INTFLG ;COUNT INTERRUPT
6856 011602 012737 000001 002124 MOV #1,$GDDAT ;STORE EXPECTED
6857 011610 013737 002436 002126 MOV INTFLG,$BDDAT ;SAVE INT. COUNT
6858 011616 023737 002124 002126 CMP $GDDAT,$BDDAT ;SHOULD BE ONE INTERRUPT
6859 011624 001401 BEQ 1$
  
```



```
6860 011626 104013          ERROR 13          :MULTIPLE INTERRUPTS RECEIVED
6861 011630 000002          1$: RTI          :RETURN FROM INT.
6862
6863          :INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 204-1774
6864          :FOR A BYTE COUNT OF 1 TO 4 VECTORS IN VECTOR ADDR MEMORY.
6865 011632 005237 002436          INTBY4: INC INTFLG          :COUNT INTERRUPT
6866 011636 013737 002472 002124          MOV BYNUM,$GDDAT          :STORE EXPECTED
6867 011644 013737 002436 002126          MOV INTFLG,$BDDAT          :SAVE INT. COUNT
6868 011652 023737 002124 002126          CMP $GDDAT,$BDDAT          :INTERRUPTS SHOULD NOT EXCEED BYTE COUNT
6869 011660 002001          BGE 1$
6870 011662 104013          ERROR 13          :MULTIPLE INTERRUPTS RECEIVED
6871          :MORE INTERRUPTS THAN BYTE COUNT
6872 011664 000002          1$: RTI          :RETURN FROM INT.
6873
6874          :COMMON ILLEGAL VECTOR RETURN ROUTINE FOR APT MODE.
6875 011666 011637 002120          TRPOUT: MOV (SP),$GDADR          :SAVE ADDRESS OF TEST
6876 011672 004537 011326          JSR R5,FRMBUF          :RETURN LOCATIONS 0-202
6877 011676 104014          ERROR 14          :ILLEGAL VECTOR ADDR MEM ERROR
6878 011700 000000          HALT          :CANNOT CONTINUE
6879 011702 000000          HALT          :CANNOT CONTINUE
6880
6881          :COMMON ILLEGAL VECTOR RETURN ROUTINE FOR VECTOR AREA
6882          :204-1774
6883 011704 011637 002120          TRPALL: MOV (SP),$GDADR          :SAVE PC OF TEST
6884 011710 104014          ERROR 14          :ILLEGAL VECTOR ADDR MEM ERROR
6885 011712 000002          RTI          :RETURN
6886
6887          :RESTORE VECTOR JUST TESTED TO ERROR SUBROUTINE FOR 0-202
6888 011714 022737 000001 002210          RES200: CMP #1,$ENV          :CHECK FOR APT
6889 011722 001405          BEQ 1$          :YES,BRANCH
6890 011724 012723 012122          MOV #TRP200,(R3)+          :ERROR TRAP SUB. FOR 0-200
6891 011730 012713 000300          MOV #300,(R3)          :PSW;VER:0
6892 011734 000404          BR 2$          :RETURN
6893 011736 012723 011666          1$: MOV #TRPOUT,(R3)+          :RESTORE APT SUB TRAP
6894          :IF ON APT
6895 011742 012713 000300          MOV #300,(R3)          :RESTORE PSW SAVE;VER:0
6896 011746 062737 000004 002452          2$: ADD #4,VECLOC          :UPDATE FOR NEXT VECTOR
6897 011754 000205          RTS R5          :RETURN
6898
6899          :ROUTINE TO SET UP TRAP SUBROUTINE 0-200 IN STANDALONE MODE OR
6900          :A COMMON ERROR VECTOR ROUTINE SETUP UNDER APT.
6901
6902 011756 010046          CAT200: MOV R0,-(SP)          :SAVE R0
6903 011760 017737 170154 002460          MOV @SWR,SWRSAV          :SAVE LOCATION 176,SOFT SWR
6904 011766 022737 000001 002210          CMP #1,$ENV          :CHECK FOR APT
6905 011774 001411          BEQ APT200          :YES,SET UP TRAPS FOR APT
6906 011776 005000          CLR R0          :START CATCHER SUB AT 0
6907 012000 012720 012122          1$: MOV #TRP200,(R0)+          :STORE VECTOR TRAP SUBROUTINE
6908 012004 012720 000300          MOV #300,(R0)+          :PSW;VER:0
6909 012010 022700 000204          CMP #204,R0          :FINISHED 0-202
6910 012014 001371          BNE 1$
6911 012016 000410          BR END200          :RETURN IN STAND ALONE MODE
6912 012020 005000          APT200: CLR R0          :START AT VECTOR 0
6913 012022 012720 011666          1$: MOV #TRPOUT,(R0)+          :VECTOR ERROR ROUTINE
6914 012026 012720 000300          MOV #300,(R0)+          :PSW NEXT;VER:0
```

```

6916 012032 022700 000204          CMP      #204,R0          :DONE?
6917 012036 001371          BNE      1$             :DO NEXT VECTOR
6918 012040 012600          END200: MOV      (SP)+,R0
6919 012042 012713 012056          MOV      #INTSR3,(R3)  :STORE VECTOR ROUTINE
6920 012046 012763 000300 000002          MOV      #300,2(R3)   :STORE PSW;VER:0
6921 012054 000205          RTS       R5           :EXIT
6922
6923
6924          :INTERRUPT SERVICE ROUTINE USED TO VERIFY INTERRUPTS 0-200,BYTE COUNT 0-3
6925          :FOR BYTE COUNTS OF 1 TO 4 VECTORS IN VECTOR ADDRESS MEMORY.
6926 012056 005237 002436          INTSR3: INC      INTFLG          :COUNT INTERRUPT
6927 012062 013737 002472 002124          MOV      BYNUM,$GDDAT  :STORE EXPECTED
6928 012070 013737 002436 002126          MOV      INTFLG,$BDDAT :SAVE INT. COUNT
6929 012076 023737 002124 002126          CMP      $GDDAT,$BDDAT :INTERRUPTS SHOULD NOT EXCEED BYTE COUNT
6930 012104 002005          BGE      1$
6931 012106 004537 011326          JSR      R5,FRMBUF     :RESTORE 0-202 BEFORE ERROR
6932 012112 104013          ERROR   13           :MULTIPLE INTERRUPTS RECEIVED
6933          :MORE INTERRUPTS THAN BYTE COUNT
6934 012114 004537 011756          JSR      R5,CAT200     :RESTORE VECTOR SUBROUTINES
6935 012120 000002          1$:      RTI           :RETURN FROM INT.
6936
6937          :COMMON ILLEGAL VECTOR ERROR RETURN ROUTINE FOR 0-200
6938 012122 011637 002120          TRP200: MOV      (SP),$GADR  :SAVE PC OF TEST
6939 012126 004537 011326          JSR      R5,FRMBUF     :RETURN LOC 0-200
6940 012132 104014          ERROR   14           :ILLEGAL VECTOR ADDR MEM ERROR
6941 012134 004537 011756          JSR      R5,CAT200     :RESTORE VECTOR SUBROUTINES FOR PROCEED
6942 012140 000002          RTI
6943
6944          :VECTOR SERVICE ROUTINE FOR VECTOR UNIQUENESS TEST
6945          :SETVEC: MOV      (R5)+,(R4)+  :STORE VALUE IN BUFFER  ;;GPA
6946 012142 017624 000000          SETVEC: MOV      @ (SP),(R4)+  :STORE VECTOR NUMBER IN BUFFER ;;GPA
6947 012146 005726          TST      (SP)+        :RESTORE STACK FOR RTI
6948 012150 005237 002436          INC      INTFLG          :COUNT INTERRUPT
6949 012154 022737 000100 002436          CMP      #64.,INTFLG
6950 012162 002001          BGE      1$           :64 EXPECTED INTERRUPTS?
6951 012164 104013          ERROR   13           :ERROR ,MORE THAN 64 INTERRUPTS
6952 012166 000002          1$:      RTI           :RETURN FOR NEXT INTERRUPT
6953
6954          :INTERRUPT SERVICE ROUTINES THAT ARE STORED FROM 300-674
6955          :IN THE VECTOR UNIQUENESS TEST.
6956          : R5 POINTS TO "SETVEC:" DURING TESTS 5 AND 6, AND CALLS          ;;GPA
6957          : CHANGED FROM "JSR R5,SETVEC" TO "JSR PC,(R5)" TO CONSERVE      ;;GPA
6958          : SPACE AND REMAIN UNDER 4KW TOTAL PROGRAM SIZE.              ;;GPA
6959 012170 004715          INTO:   JSR      PC,(R5) ;STORE 300 INTO VBUF,TO BE          ;;GPA
6960 012172 000300          300          :CHECKED AFTER ALL INTERRUPTS.
6961 012174 004715          INT1:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6962 012176 000304          304
6963 012200 004715          INT2:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6964 012202 000310          310
6965 012204 004715          INT3:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6966 012206 000314          314
6967 012210 004715          INT4:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6968 012212 000320          320
6969 012214 004715          INT5:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6970 012216 000324          324
6971 012220 004715          INT6:   JSR      PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA

```

6972	012222	000330		330	
6973	012224	004715	INT7:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6974	012226	000334		334	
6975	012230	004715	INT8:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6976	012232	000340		340	
6977	012234	004715	INT9:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6978	012236	000344		344	
6979	012240	004715	INT10:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6980	012242	000350		350	
6981	012244	004715	INT11:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6982	012246	000354		354	
6983	012250	004715	INT12:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6984	012252	000360		360	
6985	012254	004715	INT13:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6986	012256	000364		364	
6987	012260	004715	INT14:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6988	012262	000370		370	
6989	012264	004715	INT15:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6990	012266	000374		374	
6991	012270	004715	INT16:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6992	012272	000400		400	
6993	012274	004715	INT17:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6994	012276	000404		404	
6995	012300	004715	INT18:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6996	012302	000410		410	
6997	012304	004715	INT19:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
6998	012306	000414		414	
6999	012310	004715	INT20:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7000	012312	000420		420	
7001	012314	004715	INT21:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7002	012316	000424		424	
7003	012320	004715	INT22:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7004	012322	000430		430	
7005	012324	004715	INT23:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7006	012326	000434		434	
7007	012330	004715	INT24:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7008	012332	000440		440	
7009	012334	004715	INT25:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7010	012336	000444		444	
7011	012340	004715	INT26:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7012	012342	000450		450	
7013	012344	004715	INT27:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7014	012346	000454		454	
7015	012350	004715	INT28:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7016	012352	000460		460	
7017	012354	004715	INT29:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7018	012356	000464		464	
7019	012360	004715	INT30:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7020	012362	000470		470	
7021	012364	004715	INT31:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7022	012366	000474		474	
7023	012370	004715	INT32:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7024	012372	000500		500	
7025	012374	004715	INT33:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7026	012376	000504		504	
7027	012400	004715	INT34:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA

7028	012402	000510		510	
7029	012404	004715	INT35:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7030	012406	000514		514	
7031	012410	004715	INT36:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7032	012412	000520		520	
7033	012414	004715	INT37:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7034	012416	000524		524	
7035	012420	004715	INT38:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7036	012422	000530		530	
7037	012424	004715	INT39:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7038	012426	000534		534	
7039	012430	004715	INT40:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7040	012432	000540		540	
7041	012434	004715	INT41:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7042	012436	000544		544	
7043	012440	004715	INT42:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7044	012442	000550		550	
7045	012444	004715	INT43:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7046	012446	000554		554	
7047	012450	004715	INT44:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7048	012452	000560		560	
7049	012454	004715	INT45:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7050	012456	000564		564	
7051	012460	004715	INT46:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7052	012462	000570		570	
7053	012464	004715	INT47:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7054	012466	000574		574	
7055	012470	004715	INT48:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7056	012472	000600		600	
7057	012474	004715	INT49:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7058	012476	000604		604	
7059	012500	004715	INT50:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7060	012502	000610		610	
7061	012504	004715	INT51:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7062	012506	000614		614	
7063	012510	004715	INT52:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7064	012512	000620		620	
7065	012514	004715	INT53:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7066	012516	000624		624	
7067	012520	004715	INT54:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7068	012522	000630		630	
7069	012524	004715	INT55:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7070	012526	000634		634	
7071	012530	004715	INT56:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7072	012532	000640		640	
7073	012534	004715	INT57:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7074	012536	000644		644	
7075	012540	004715	INT58:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7076	012542	000650		650	
7077	012544	004715	INT59:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7078	012546	000654		654	
7079	012550	004715	INT60:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7080	012552	000660		660	
7081	012554	004715	INT61:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
7082	012556	000664		664	
7083	012560	004715	INT62:	JSR	PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA

CNDRDA DRV11J DIAG TST PRT2
CNDRDA.P11 10-DEC-82 14:44

MACY11 30(1046) 15-DEC-82 15:26 K 3 PAGE 59-3
PROGRAM SUBROUTINES

SEQ 0036

7084 012562 000670
7085 012564 004715
7086 012566 000674
7087
7088
7089
7090
7091
7092
7093
7094
7095 012570 001 376
7096 012572 002 375
7097 012574 004 373
7098 012576 010 367
7099 012600 020 357
7100 012602 040 337
7101 012604 100 277
7102 012606 200 177
7103 012610 000000
7104
7105

670
INT63: JSR PC,(R5) ;VECTOR UNIQUENESS TEST, INT. SERV RTN ;;GPA
674

.SBTTL PATTEKNS FOR REGISTER R/W
:
:PATTERNS USED FOR LOADING/READING REGISTERS
:ISR INTERRUPT SERVICE REGISTER
:IRR INTERRUPT REQUEST REGISTER
:IMR INTERRUPT MASK REGISTER

BGCHP3: .BYTE 1,376
.BYTE 2,375
.BYTE 4,373
.BYTE 10,367
.BYTE 20,357
.BYTE 40,337
.BYTE 100,277
.BYTE 200,177
EDCHP3: 000000

7107
7108
7109

.SBTTL SYSMAC ROUTINES

.SBTTL TYPE ROUTINE

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

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

(1)	012612	105737	002157	\$TYPE:	TSTB	\$TPFLG	:: IS THERE A TERMINAL?
(1)	012616	100002			BPL	1\$:: BR IF YES
(1)	012620	000C30			HALT		:: HALT HERE IF NO TERMINAL
(1)	012622	000430			BR	3\$:: LEAVE
(1)	012624	010046		1\$:	MOV	RO,-(SP)	:: SAVE RO
(1)	012626	017600	000002		MOV	@2(SP),RO	:: GET ADDRESS OF ASCIZ STRING
(1)	012632	122737	000001 002210		CMPB	#APTENV,\$ENV	:: RUNNING IN APT MODE
(1)	012640	001011			BNE	62\$:: NO,GO CHECK FOR APT CONSOLE
(1)	012642	132737	000100 002211		BITB	#APTSPOOL,\$ENVM	:: SPOOL MESSAGE TO APT
(1)	012650	001405			BEQ	62\$:: NO,GO CHECK FOR CONSOLE
(1)	012652	010037	012662		MOV	RO,61\$:: SETUP MESSAGE ADDRESS FOR APT
(1)	012656	004737	013102		JSR	PC,\$ATY3	:: SPOOL MESSAGE TO APT
(1)	012662	000000		61\$:	.WORD	0	:: MESSAGE ADDRESS
(1)	012664	132737	000040 002211	62\$:	BITB	#APTCSUP,\$ENVM	:: APT CONSOLE SUPPRESSED
(1)	012672	001003			BNE	60\$:: YES,SKIP TYPE CJT
(1)	012674	112046		2\$:	MOVB	(RO)+,-(SP)	:: PUSH CHARACTER TO BE TYPED ONTO STACK
(1)	012676	001005			BNE	4\$:: BR IF IT ISN'T THE TERMINATOR
(1)	012700	005726			TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK
(1)	012702	012600		60\$:	MOV	(SP)+,RO	:: RESTORE RO
(1)	012704	062716	000002	3\$:	ADD	#2,(SP)	:: ADJUST RETURN PC
(1)	012710	000002			RTI		:: RETURN
(1)	012712	122716	000011	4\$:	CMPB	#HT,(SP)	:: BRANCH IF <HT>
(1)	012716	001430			BEQ	8\$	
(1)	012720	122716	000200		CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
(1)	012724	001006			BNE	5\$	
(1)	012726	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
(1)	012730	104401			TYPE		:: TYPE A CR AND LF
(1)	012732	002165			\$CRLF		
(1)	012734	105037	013070		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
(1)	012740	000755			BR	2\$:: GET NEXT CHARACTER
(1)	012742	004737	013024	5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
(1)	012746	123726	002156	6\$:	CMPB	\$FILLC,(SP)+	:: IS IT TIME FOR FILLER CHARS.?
(1)	012752	001350			BNE	2\$:: IF NO GO GET NEXT CHAR.
(1)	012754	013746	002154		MOV	\$NULL,-(SP)	:: GET # OF FILLER CHARS. NEEDED
(1)							:: AND THE NULL CHAR.
(1)	012760	105366	000001	7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
(1)	012764	002770			BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK

```
(1) 012766 004737 013024 JSR PC,$TYPEC ;:GO TYPE A NULL
(1) 012772 105337 013070 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
(1) 012776 000770 BR 7$ ;:LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 013000 112716 000040 8$: MOVB #' ,(SP) ;:REPLACE TAB WITH SPACE
(1) 013004 004737 013024 9$: JSR PC,$TYPEC ;:TYPE A SPACE
(1) 013010 132737 000007 013070 BITB #7,$CHARCNT ;:BRANCH IF NOT AT
(1) 013016 001372 BNE 9$ ;:TAB STOP
(1) 013020 005726 TST (SP)+ ;:POP SPACE OFF STACK
(1) 013022 000724 BR 2$ ;:GET NEXT CHARACTER
(1) 013024 105777 167120 $TYPEC: TSTB @STPS ;:WAIT UNTIL PRINTER IS READY
(1) 013030 100375 BPL $TYPEC
(1) 013032 116677 000002 167112 MOVB 2(SP),@$TPB ;:LOAD CHAR TO BE TYPED INTO DATA REG.
(1)
(1) 013040 122766 000015 000002 CMPB #CR,2(SP) ;:IS CHARACTER A CARRIAGE RETURN?
(1) 013046 001003 BNE 1$ ;:BRANCH IF NO
(1) 013050 105037 013070 CLRB $CHARCNT ;:YES--CLEAR CHARACTER COUNT
(1) 013054 000406 BR $TYPEX ;:EXIT
(1) 013056 122766 000012 000002 1$: CMPB #LF,2(SP) ;:IS CHARACTER A LINE FEED?
(1) 013064 001402 BEQ $TYPEX ;:BRANCH IF YES
(1) 013066 105227 INCB (PC)+ ;:COUNT THE CHARACTER
(1) 013070 000000 $CHARCNT: WORD 0 ;:CHARACTER COUNT STORAGE
(1) 013072 000207 $TYPEX: RTS PC
```

7110

.SBTTL APT COMMUNICATIONS ROUTINE

```
(1)
(1)
(1) ;*****
(2)
(1) 013074 112737 000001 013340 $ATY1: MOVB #1,$FFLG ;:TC REPORT FATAL ERROR
(1) 013102 112737 000001 013336 $ATY3: MOVB #1,$MFLG ;:TO TYPE A MESSAGE
(1) 013110 000403 BR $ATYC
(1) 013112 112737 000001 013340 $ATY4: MOVB #1,$FFLG ;:TO ONLY REPORT FATAL ERROR
(1) 013120 $ATYC:
(3) 013120 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
(3) 013122 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
(1) 013124 105737 013336 TSTB $MFLG ;:SHOULD TYPE A MESSAGE?
(1) 013130 001450 BEQ 5$ ;:IF NOT: BR
(1) 013132 122737 000001 002210 CMPB #APTENV,$ENV ;:OPERATING UNDER APT?
(1) 013140 001031 BNE 3$ ;:IF NOT: BR
(1) 013142 132737 000100 002211 BITB #APTPOOL,$ENVM ;:SHOULD SPOOL MESSAGES?
(1) 013150 001425 BEQ 3$ ;:IF NOT: BR
(1) 013152 017600 000004 MOV @4(SP),R0 ;:GET MESSAGE ADDR.
(1) 013156 062766 000002 000004 ADD #2,4(SP) ;:BUMP RETURN ADDR.
(1) 013164 005737 002170 1$: TST $MSGTYPE ;:SEE IF DONE W/ LAST XMISSION?
(1) 013170 001375 BNE 1$ ;:IF NOT: WAIT
(1) 013172 010037 002204 MOV R0,$MSGAD ;:PUT ADDR IN MAILBOX
(1)
(1) 013176 105720 2$: TSTB (R0)+ ;:FIND END OF MESSAGE
(1) 013200 001376 BNE 2$
(1) 013202 163700 002204 SUB $MSGAD,R0 ;:SUB START OF MESSAGE
(1) 013206 006200 ASR R0 ;:GET MESSAGE LNGTH IN WORDS
(1) 013210 010037 002206 MOV R0,$MSGGLT ;:PUT LENGTH IN MAILBOX
(1) 013214 012737 000004 002170 MOV #4,$MSGTYPE ;:TELL APT TO TAKE MSG.
```

```
(1) 013222 000413  
(1) 013224 017637 000004 013250 3$: BR 5$  
(1) 013232 062766 000002 000004 MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE  
(3) 013240 013746 177776 ADD #2,4(SP) ;;BUMP RETURN ADDRESS  
(1) 013244 004737 012612 MOV 177776,-(SP) ;;PUSH 177776 ON STACK  
(1) 013250 000000 JSR PC,$TYPE ;;CALL TYPE MACRO  
(1) 013252 .WORD 0  
(1) 013252 105737 013340 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?  
(1) 013256 001416 BEQ 12$ ;;IF NOT: BR  
(1) 013260 005737 002210 TST $ENV ;;RUNNING UNDER APT?  
(1) 013264 001413 BEQ 12$ ;;IF NOT: BR  
(1) 013266 005737 002170 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?  
(1) 013272 001375 BNE 11$ ;;IF NOT: WAIT  
(1) 013274 017637 000004 002172 MOV @4(SP),$FATAL ;;GET ERROR #  
(1) 013302 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.  
(1) 013310 005237 002170 INC $MSGTYPE ;;TELL APT TO TAKE ERROR  
(1) 013314 105037 013340 12$: CLRB $FFLG ;;CLEAR FATAL FLAG  
(1) 013320 105037 013337 CLRB $LFLG ;;CLEAR LOG FLAG  
(1) 013324 105037 013336 CLRB $MFLG ;;CLEAR MESSAGE FLAG  
(3) 013330 012601 MOV (SP)+,R1 ;;POP STACK INTO R1  
(3) 013332 012600 MOV (SP)+,R0 ;;POP STACK INTO R0  
(1) 013334 000207 RTS PC ;;RETURN  
(1) 013336 000 $MFLG: .BYTE 0 ;;MESSG. FLAG  
(1) 013337 000 $LFLG: .BYTE 0  
(1) 013340 000 $FFLG: .BYTE 0 ;;LOG FLAG  
(1) 013342 .EVEN ;;FATAL FLAG  
(1) 000200 APTSIZE=200  
(1) 000001 APTENV=001  
(1) 000100 APTSPool=100  
(1) 000040 APTCSUP=040  
7111 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE  
(1)  
(2) *****  
(1) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT  
(1) *OCTAL (ASCII) NUMBER AND TYPE IT.  
(1) *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE  
(1) *CALL:  
(1) * MOV NUM,-(SP) ;;NUMBER TO BE TYPED  
(1) * TYPOS ;;CALL FOR TYPEOUT  
(1) * .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE  
(1) * .BYTE M ;;M=1 OR 0  
(1) * ;;1=TYPE LEADING ZEROS  
(1) * ;;0=SUPPRESS LEADING ZEROS  
(1) *$STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST  
(1) *$TYPOS OR $TYPOC  
(1) *CALL:  
(1) * MOV NUM,-(SP) ;;NUMBER TO BE TYPED  
(1) * TYPON ;;CALL FOR TYPEOUT  
(1) *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER  
(1) *CALL:  
(1) * MOV NUM,-(SP) ;;NUMBER TO BE TYPED  
(1) * TYPOC ;;CALL FOR TYPEOUT  
(1)
```



```

(1) 013342 017646 000000
(1) 013346 116637 000001
(1) 013354 112637 013567
(1) 013360 062716 000002
(1) 013364 000406
(1) 013366 112737 000001
(1) 013374 112737 000006
(1) 013402 112737 000005
(1) 013410 010346
(1) 013412 010446
(1) 013414 010546
(1) 013416 113704 013567
(1) 013422 005404
(1) 013424 062704 000006
(1) 013430 110437 013566
(1) 013434 113704 013565
(1) 013440 016605 000012
(1) 013444 005003
(1) 013446 006105
(1) 013450 000404
(1) 013452 006105
(1) 013454 006105
(1) 013456 006105
(1) 013460 010503
(1) 013462 006103
(1) 013464 105337 013566
(1) 013470 100016
(1) 013472 042703 177770
(1) 013476 001002
(1) 013500 005704
(1) 013502 001403
(1) 013504 005204
(1) 013506 052703 000060
(1) 013512 052703 000040
(1) 013516 110337 013562
(1) 013522 104401 013562
(1) 013526 105337 013564
(1) 013532 003347
(1) 013534 002402
(1) 013536 005204
(1) 013540 000744
(1) 013542 012605
(1) 013544 012604
(1) 013546 012603
(1) 013550 016666 000002 000004
(1) 013556 012616
(1) 013560 000002
(1) 013562 000
(1) 013563 000
(1) 013564 000
(1) 013565 000
(1) 013566 000000
7112
(1)
(2)
(1)

```

```

$TYPOS: MOV @ (SP), -(SP) ;; PICKUP THE MODE
MOV 1 (SP), $OFILL ;; LOAD ZERO FILL SWITCH
MOV (SP)+, $OMODE+1 ;; NUMBER OF DIGITS TO TYPE
ADD #2, (SP) ;; ADJUST RETURN ADDRESS
BR $TYPON
$TYPOC: MOV #1, $OFILL ;; SET THE ZERO FILL SWITCH
MOV #6, $OMODE+1 ;; SET FOR SIX(6) DIGITS
$TYPON: MOV #5, $OCNT ;; SET THE ITERATION COUNT
MOV R3, -(SP) ;; SAVE R3
MOV R4, -(SP) ;; SAVE R4
MOV R5, -(SP) ;; SAVE R5
MOV $OMODE+1, R4 ;; GET THE NUMBER OF DIGITS TO TYPE
NEG R4
ADD #6, R4 ;; SUBTRACT IT FOR MAX. ALLOWED
MOV R4, $OMODE ;; SAVE IT FOR USE
MOV $OFILL, R4 ;; GET THE ZERO FILL SWITCH
MOV 12 (SP), R5 ;; PICKUP THE INPUT NUMBER
CLR R3 ;; CLEAR THE OUTPUT WORD
1$: ROL R5 ;; ROTATE MSB INTO 'C'
BR 3$ ;; GO DO MSB
2$: ROL R5 ;; FORM THIS DIGIT
ROL R5
MOV R5, R3
3$: ROL R3 ;; GET LSB OF THIS DIGIT
DECB $OMODE ;; TYPE THIS DIGIT?
BPL 7$ ;; BR IF NO
BIC #177770, R3 ;; GET RID OF JUNK
BNE 4$ ;; TEST FOR 0
TST R4 ;; SUPPRESS THIS 0?
BEQ 5$ ;; BR IF YES
4$: INC R4 ;; DON'T SUPPRESS ANYMORE 0'S
BIS #'0, R3 ;; MAKE THIS DIGIT ASCII
5$: BIS #'1, R3 ;; MAKE ASCII IF NOT ALREADY
MOV R3, 8$ ;; SAVE FOR TYPING
TYPE 8$ ;; GO TYPE THIS DIGIT
7$: DECB $OCNT ;; COUNT BY 1
BGT 2$ ;; BR IF MORE TO DO
BLT 6$ ;; BR IF DONE
INC R4 ;; INSURE LAST DIGIT ISN'T A BLANK
BR 2$ ;; GO DO THE LAST DIGIT
6$: MOV (SP)+, R5 ;; RESTORE R5
MOV (SP)+, R4 ;; RESTORE R4
MOV (SP)+, R3 ;; RESTORE R3
MOV 2 (SP), 4 (SP) ;; SET THE STACK FOR RETURNING
MOV (SP)+, (SP)
RTI ;; RETURN
8$: .BYTE 0 ;; STORAGE FOR ASCII DIGIT
.BYTE 0 ;; TERMINATOR FOR TYPE ROUTINE
$OCNT: .BYTE 0 ;; OCTAL DIGIT COUNTER
$OFILL: .BYTE 0 ;; ZERO FILL SWITCH
$OMODE: .WORD 0 ;; NUMBER OF DIGITS TO TYPE
.SBTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

```

*****
*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) 013570
(3) 013570 010046
(3) 013572 010146
(3) 013574 010246
(3) 013576 010346
(3) 013600 010546
(1) 013602 012746 020200
(1) 013606 016605 000020
(1) 013612 100004
(1) 013614 005405
(1) 013616 112766 000055 000001
(1) 013624 005000 1$: CLR R0 ;;ZERO THE CONSTANTS INDEX
(1) 013626 012703 014004 MOV #SDBLK,R3 ;;SETUP THE OUTPUT POINTER
(1) 013632 112723 000040 MOV #',(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
(1) 013636 005002 2$: CLR R2 ;;CLEAR THE BCD NUMBER
(1) 013640 016001 013774 MOV $DTBL(R0),R1 ;;GET THE CONSTANT
(1) 013644 160105 3$: SUB R1,R5 ;;FORM THIS BCD DIGIT
(1) 013646 002402 BLT 4$ ;;BR IF DONE
(1) 013650 005202 INC R2 ;;INCREASE THE BCD DIGIT BY 1
(1) 013652 000774 BR 3$
(1) 013654 060105 4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
(1) 013656 005702 TST R2 ;;CHECK IF BCD DIGIT=0
(1) 013660 001002 BNE 5$ ;;FALL THROUGH IF 0
(1) 013662 105716 TSTB (SP) ;;STILL DOING LEADING 0'S?
(1) 013664 100407 BMI 7$ ;;BR IF YES
(1) 013666 106316 5$: ASLB (SP) ;;MSD?
(1) 013670 103003 BCC 6$ ;;BR IF NO
(1) 013672 116663 000001 177777 MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 013700 052702 000060 6$: BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII
(1) 013704 052702 000040 7$: BIS #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 013710 110223 MOVB R2,(R3)+ ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 013712 005720 TST (R0)+ ;;JUST INCREMENTING
(1) 013714 020027 000010 CMP R0,#10 ;;CHECK THE TABLE INDEX
(1) 013720 002746 BLT 2$ ;;GO DO THE NEXT DIGIT
(1) 013722 003002 BGT 8$ ;;GO TO EXIT
(1) 013724 010502 MOV R5,R2 ;;GET THE LSD
(1) 013726 000764 BR 6$ ;;GO CHANGE TO ASCII
(1) 013730 105726 8$: TSTB (SP)+ ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 013732 100003 BPL 9$ ;;BR IF NO
(1) 013734 116663 177777 177776 9$: MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 013742 105013 CLRB (R3) ;;SET THE TERMINATOR
(3) 013744 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
(3) 013746 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 013750 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 013752 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 013754 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 013756 104401 014004 TYPE $SDBLK ;;NOW TYPE THE NUMBER
(1) 013762 016666 000002 000004 MOV 2(SP),4(SP) ;;ADJUST THE STACK

```

(1) 013770 012616
(1) 013772 000002
(1) 013774 023420
(1) 013776 001750
(1) 014000 000144
(1) 014002 000012
(1) 014004 000004

7113

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

(1) 014014
(1) 014014 104407
(2) 014016 104407
(1) 014020 105237 002103
(1) 014024 001775
(1) 014026 013777 002102 166106
(1) 014034 005237 002112
(1) 014040 011637 002116
(1) 014044 162737 000002 002116
(1) 014052 117737 166040 002114
(1) 014060 032777 000000 166052
(1) 014066 001004
(1) 014070 004737 014170
(1) 014074 104401 002165
(1) 014100
(1) 014100 122737 000001 002110
(1) 014106 001007
(1) 014110 113737 002114 014122
(1) 014116 004737 013112
(1) 014122 000
(1) 014123 000
(1) 014124 000777
(1) 014126 005777 166006
(1) 014132 100002
(1) 014134 000000
(1) 014136 104407
(1) 014140 032777 001000 165772
(1) 014146 001402
(1) 014150 013716 002110
(1) 014154 005737 002162
(1) 014160 001402
(1) 014162 013716 002162
(1) 014166
(1) 014166 000002

7114
7115

```
MOV (SP)+,(SP)
RTI ;:RETURN TO USER
SDTBL: 10000.
      1000.
      100.
      10.
SDBLK: .BLKW 4
.SBTTL ERROR HANDLER ROUTINE

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

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

20$: CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
BNE 2$ ;:NO,SKIP APT ERROR REPORT
MOVB $ITEMB,21$ ;:SET ITEM NUMBER AS ERROR NUMBER
JSR PC,$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!
CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
3$: BIT #BIT09,@SWR ;:LOOP ON ERROR SWITCH SET?
BEQ 4$ ;:BR IF NO
MOV $LPERR,(SP) ;:FUDGE RETURN FOR LOOPING
4$: TS' $ESCAPE ;:CHECK FOR AN ESCAPE ADDRESS
BEJ 5$ ;:BR IF NONE
MCV $ESCAPE,(SP) ;:FUDGE RETURN ADDRESS FOR ESCAPE

5$: RTI ;:RETURN

:*****
:GO TYPE ERROR
```

```
7116 ;GO UPDATE SOFTWARE SWR IF 'CNTRL/G'  
7117 ;*****  
7118 014170 113737 002102 002432 SWRCK: MOV B $TSTNM,TSTNUM ;SET UP TEST # ON ER  
7119 014176 004737 014206 JSR PC,$ERRTYP ;GO TYPE ERROR  
7120 014202 104407 CKSWR ;GO LOOK FOR SWR CHANGE  
7121 014204 000207 RTS PC ;RETURN TO ERROR HANDLER  
7122 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE  
(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1) 014206 $ERRTYP:  
(1) 014206 104401 002165 TYPE , $CRLF ;:'CARRIAGE RETURN' & 'LINE FEED'  
(1) 014212 010046 MOV RO,-(SP) ;:SAVE RO  
(1) 014214 005000 CLR RO ;:PICKUP THE ITEM INDEX  
(1) 014216 153700 002114 BISB @#$ITEMB,RO  
(1) 014222 001004 BNE 1$ ;:IF ITEM NUMBER IS ZERO, JUST  
(1) ;:TYPE THE PC OF THE ERROR  
(2) 014224 013746 002116 MOV $ERRPC,-(SP) ;:SAVE $ERRPC FOR TYPEOUT  
(2) ;:ERROR ADDRESS  
(2) 014230 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)  
(1) 014232 000426 BR 6$ ;:GET OUT  
(1) 014234 005300 1$: DEC RO ;:ADJUST THE INDEX SO THAT IT WILL  
(1) 014236 006300 ASL RO ;: WORK FOR THE ERROR TABLE  
(1) 014240 006300 ASL RO  
(1) 014242 006300 ASL RO  
(1) 014244 062700 002252 ADD #$ERRTB,RO ;:FORM TABLE POINTER  
(1) 014250 012037 014260 MOV (RO)+,2$ ;:PICKUP 'ERROR MESSAGE' POINTER  
(1) 014254 001404 BEQ 3$ ;:SKIP TYPEOUT IF NO POINTER  
(1) 014256 104401 TYPE ;:TYPE THE 'ERROR MESSAGE'  
(1) 014260 000000 2$: .WORD 0 ;:'ERROR MESSAGE' POINTER GOES HERE  
(1) 014262 104401 002165 TYPE , $CRLF ;:'CARRIAGE RETURN' & 'LINE FEED'  
(1) 014266 012037 014276 3$: MOV (RO)+,4$ ;:PICKUP 'DATA HEADER' POINTER  
(1) 014272 001404 BEQ 5$ ;:SKIP TYPEOUT IF 0  
(1) 014274 104401 TYPE ;:TYPE THE 'DATA HEADER'  
(1) 014276 000000 4$: .WORD 0 ;:'DATA HEADER' POINTER GOES HERE  
(1) 014300 104401 002165 TYPE , $CRLF ;:'CARRIAGE RETURN' & 'LINE FEED'  
(1) 014304 011000 5$: MOV (RO),RO ;:PICKUP 'DATA TABLE' POINTER  
(1) 014306 001004 BNE 7$ ;:GO TYPE THE DATA  
(1) 014310 012600 6$: MOV (SP)+,RO ;:RESTORE RO  
(1)  
(1) 014312 104401 002165 TYPE , $CRLF ;:'CARRIAGE RETURN' & 'LINE FEED'  
(1) 014316 000207 RTS PC ;:RETURN  
(1) 7$:  
(2) 014320 013046 MOV @(RO)+,-(SP) ;:SAVE @(RO)+ FOR TYPEOUT  
(2) 014322 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)  
(1) 014324 005710 TST (RO) ;:IS THERE ANOTHER NUMBER?  
(1) 014326 001770 BEQ 6$ ;:BR IF NO  
(1) 014330 104401 014336 TYPE , 8$ ;:TYPE TWO(2) SPACES  
(1) 014334 000771 BR 7$ ;:LOOP  
(1) 014336 020040 000 8$: .ASCIZ / / ;:TWO(2) SPACES  
(1) .EVEN  
7123 .SBTTL SCOPE HANDLER ROUTINE STARS  
(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<7:0>  
(1) ;*CALL  
(1) ;* SCOPE ;:SCOPE=IOT  
(1) $SCOPE:  
(1) 014342 104407 CKSWR ;:TEST FOR CHANGE IN SOFT-SWR  
(1) 014342 032777 040000 165566 1$: BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?  
(1) 014352 001114 BNE $OVER ;:YES IF SW14=1  
(1) ;#####START OF CODE FOR THE XOR TESTER#####  
(1) 014354 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE 'XOR' TESTER CHANGE  
(1) ;THIS INSTRUCTION TO A 'NOP' (NOP=240)  
(1) 014356 013746 000004 MOV @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR  
(1) 014362 012737 014402 000004 MOV #5$,@#ERRVEC ;:SET FOR TIMEOUT  
(1) 014370 005737 177060 TST @#177060 ;:TIME OUT ON XOR?  
(1) 014374 012637 000004 MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR  
(1) 014400 000463 BR $SVLAD ;:GO TO THE NEXT TEST  
(1) 014402 022626 5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT  
(1) 014404 012637 000004 MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR  
(1) 014410 000423 BR 7$ ;:LOOP ON THE PRESENT TEST  
(1) 014412 6$;#####END OF CODE FOR THE XOR TESTER#####  
(1) 014412 032777 000400 165520 BIT #BIT08,@SWR ;:LOOP ON SPEC. TEST?  
(1) 014420 001404 BEQ 2$ ;:BR IF NO  
(1) 014422 127737 165512 002102 CMPB @SWR,$TSTNM ;:ON THE RIGHT TEST? SWR<7:0>  
(1) 014430 001465 BEQ $OVER ;:BR IF YES  
(1) 014432 105737 002103 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?  
(1) 014436 001421 BEQ 3$ ;:BR IF NO  
(1) 014440 123737 002115 002103 CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?  
(1) 014446 101015 BEQ 3$ ;:BR IF NO  
(1) 014450 032777 001000 165462 BIT #BIT09,@SWR ;:LOOP ON ERROR?  
(1) 014456 001404 BEQ 4$ ;:BR IF NO  
(1) 014460 013737 002110 002106 7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE  
(1) 014466 000466 BR $OVER  
(1) 014470 105037 002103 4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG  
(1) 014474 005037 002160 CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE  
(1) 014500 000415 BR 1$ ;:ESCAPE TO THE NEXT TEST  
(1) 014502 032777 004000 165430 3$: BIT #BIT11,@SWR ;:INHIBIT ITERATIONS?  
(1) 014510 001011 BNE 1$ ;:BR IF YES  
(1) 014512 005737 002176 TST $PASS ;:IF FIRST PASS OF PROGRAM  
(1) 014516 001406 BEQ 1$ ;: INHIBIT ITERATIONS  
(1) 014520 005237 002104 INC $ICNT ;:INCREMENT ITERATION COUNT  
(1) 014524 023737 002160 002104 CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE  
(1) 014532 002024 BGE $OVER ;:BR IF MORE ITERATION REQUIRED  
(1) 014534 012737 000001 002104 1$: MOV #1,$ICNT ;:REINITIALIZE THE ITERATION COUNTER  
(1) 014542 013737 014620 002160 MOV $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO  
(1) 014550 105237 002102 $SVLAD: INCB $TSTNM ;:COUNT TEST NUMBERS  
(1) 014554 113737 002102 002174 MOVB $TSTNM,$TESTN ;:SET TEST NUMBER IN APT MAILBOX  
(1) 014562 011637 002106 MOV (SP),$LPADR ;:SAVE SCOPE LOOP ADDRESS  
(1) 014566 011637 002110 MOV (SP),$LPERR ;:SAVE ERROR LOOP ADDRESS  
(1) 014572 005037 002162 CLR $ESCAPE ;:CLEAR THE ESCAPE FROM ERROR ADDRESS  
(1) 014576 112737 000001 002115 MOVB #1,$ERMAX ;:ONLY ALLOW ONE(1) ERROR ON NEXT TEST
```

```
(1) 014604 013777 002102 165330 COVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 014612 013716 002106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(1) 014616 000002 RTI ;;FIXES PS
(1) 014620 000001 $MXCNT: 1. ;;MAX. NUMBER OF ITERATIONS
7124 .SBTTL TTY INPUT ROUTINE
(1)
(2) ;;*****
(1)
(1)
(1) .ENABL LSB
(1)
(2) ;;*****
(1) ;;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ;;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ;;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1) ;;*WHEN OPERATING IN TTY FLAG MODE.
(1) 014622 022737 000176 002140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
(1) 014630 001074 BNE 15$ ;;BRANCH IF NO
(1) 014632 105777 165306 TSTB @STKS ;;CHAR THERE?
(1) 014636 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
(1) 014640 117746 165302 MOVB @STKB,-(SP) ;;SAVE THE CHAR
(1) 014644 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
(1) 014650 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
(1) 014654 001062 BNE 15$ ;;NO, RETURN TO USER
(1) 014656 123727 002134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
(1) 014664 001456 BEQ 15$ ;;BRANCH IF YES
(1)
(1) 014666 104401 015347 $GTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (^G)
(1) 014672 104401 015354 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
(2) 014676 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
(2) 014702 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 014704 104401 015365 TYPE , $MNEW ;;PROMPT FOR NEW SWR
(1) 014710 005046 19$: CLR -(SP) ;;CLEAR COUNTER
(1) 014712 005046 CLR -(SP)
(1)
(1) 014714 105777 165224 7$: TSTB @STKS ;;CHAR THERE?
(1) 014720 100375 BPL 7$ ;;IF NOT TRY AGAIN
(1)
(1) 014722 117746 165220 MOVB @STKB,-(SP) ;;PICK UP CHAR
(1) 014726 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1) 014732 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
(1) 014736 001005 BNE 10$ ;;BRANCH IF NOT
(1) 014740 104401 015342 TYPE , $CNTLU ;;YES, ECHO CONTROL-U (^U)
(1) 014744 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
(1) 014750 000757 BR 19$ ;;LET'S TRY IT AGAIN
(1)
(1)
(1) 014752 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
(1) 014756 001022 BNE 16$ ;;BRANCH IF NO
(1) 014760 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
(1) 014764 001403 BEQ 11$ ;;BRANCH IF YES
(1) 014766 016677 000002 165144 MOV 2(SP),@SWR ;;SAVE NEW SWR
(1) 014774 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
```

```
(1) 015000 104401 002165 14$: TYPE $SRLF ;;ECHO <CR> AND <LF>
(1) 015004 123727 002135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
(1) 015012 001003 BNE 15$ ;;BRANCH IF NOT
(1) 015014 012777 000100 165122 MOV #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
(1) 015022 000002 15$: RTI ;;RETURN
(1) 015024 004737 013024 16$: JSR PC,$TYPEC ;;ECHO CHAR
(1) 015030 021627 000050 CMP (SP),#60 ;;CHAR < 0?
(1) 015034 002420 BLT 18$ ;;BRANCH IF YES
(1) 015036 021627 000067 CMP (SP),#67 ;;CHAR > 7?
(1) 015042 003015 BGT 18$ ;;BRANCH IF YES
(1) 015044 042726 000060 BIC #60,(SP)+ ;;STRIP-OFF ASCII
(1) 015050 005766 000002 TST 2(SP) ;;IS THIS THE FIRST CHAR
(1) 015054 001403 BEQ 17$ ;;BRANCH IF YES
(1) 015056 006316 ASL (SP) ;;NO, SHIFT PRESENT
(1) 015060 006316 ASL (SP) ;; CHAR OVER TO MAKE
(1) 015062 006316 ASL (SP) ;; ROOM FOR NEW ONE.
(1) 015064 005266 000002 17$: INC 2(SP) ;;KEEP COUNT OF CHAR
(1) 015070 056616 177776 BIS -2(SP),(SP) ;;SET IN NEW CHAR
(1) 015074 000707 BR 7$ ;;GET THE NEXT ONE
(1) 015076 104401 002164 18$: TYPE $QUES ;;TYPE ?<CR><LF>
(1) 015102 000720 BR 20$ ;;SIMULATE CONTROL-U
(1) .DSABL LSB

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

(1) 015104 011646 $RDCHR: MOV (SP),-(SP) ;;PUSH DOWN THE PC
(1) 015106 016666 000004 000002 MOV 4(SP),2(SP) ;;SAVE THE PS
(1) 015114 105777 165024 1$: TSTB @STKS ;;WAIT FOR
(1) 015120 100375 BPL 1$ ;;A CHARACTER
(1) 015122 117766 165020 000004 MOVB @STKB,4(SP) ;;READ THE TTY
(1) 015130 042766 177600 000004 BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
(1) 015136 026627 000004 000023 CMP 4(SP),#23 ;;IS IT A CONTROL-S?
(1) 015144 001013 BNE 3$ ;;BRANCH IF NO
(1) 015146 105777 164772 2$: TSTB @STKS ;;WAIT FOR A CHARACTER
(1) 015152 100375 BPL 2$ ;;LOOP UNTIL ITS THERE
(1) 015154 117746 164766 MOVB @STKB,-(SP) ;;GET CHARACTER
(1) 015160 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 015164 022627 000021 CMP (SP)+,#21 ;;IS IT A CONTROL-Q?
(1) 015170 001366 BNE 2$ ;;IF NOT DISCARD IT
(1) 015172 000750 BR 1$ ;;YES, RESUME
(1) 015174 026627 000004 000140 3$: CMP 4(SP),#140 ;;IS IT UPPER CASE?
(1) 015202 002407 BLT 4$ ;;BRANCH IF YES
(1) 015204 026627 000004 000175 CMP 4(SP),#175 ;;IS IT A SPECIAL CHAR?
(1) 015212 003003 BGT 4$ ;;BRANCH IF YES
(1) 015214 042766 000040 000004 BIC #40,4(SP) ;;MAKE IT UPPER CASE
(1) 015222 000002 4$: RTI ;;GO BACK TO USER

*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
```

```
(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) 015224 010346 $RDLIN: MOV R3,-(SP) ;; SAVE R3
(1) 015226 012703 015332 1$: MOV #$TTYIN,R3 ;; GET ADDRESS
(1) 015232 022703 015342 2$: CMP #$TTYIN+8.,R3 ;; BUFFER FULL?
(1) 015236 101405 BLOS 4$ ;; BR IF YES
(1) 015240 104410 RDCHR ;; GO READ ONE CHARACTER FROM THE TTY
(1) 015242 112613 MOVB (SP)+,(R3) ;; GET CHARACTER
(1) 015244 122713 000177 10$: CMPB #177,(R3) ;; IS IT A RUBOUT
(1) 015250 001003 BNE 3$ ;; SKIP IF NOT
(1) 015252 104401 002164 4$: TYPE ,SQLES ;; TYPE A '?'
(1) 015256 000763 BR 1$ ;; CLEAR THE BUFFER AND LOOP
(1) 015260 111337 015330 3$: MOVB (R3),9$ ;; ECHO THE CHARACTER
(1) 015264 104401 015330 TYPE ,9$
(1) 015270 122723 000015 CMPB #15,(R3)+ ;; CHECK FOR RETURN
(1) 015274 001356 BNE 2$ ;; LOOP IF NOT RETURN
(1) 015276 105063 177777 CLR B -1(R3) ;; CLEAR RETURN (THE 15)
(1) 015302 104401 002166 TYPE ,SLF ;; TYPE A LINE FEED
(1) 015306 012603 MOV (SP)+,R3 ;; RESTORE R3
(1) 015310 011646 MOV (SP),-(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 015312 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 015320 012766 015332 000004 MOV #$TTYIN,4(SP)
(1) 015326 000002 RTI ;; RETURN
(1) 015330 000 9$: .BYTE 0 ;; STORAGE FOR ASCII CHAR. TO TYPE
(1) 015331 000 .BYTE 0 ;; TERMINATOR
(1) 015332 000010 $TTYIN: .BLKB 8 ;; RESERVE 8 BYTES FOR TTY INPUT
(1) 015342 052536 005015 000 $CNTLU: .ASCIZ /^U/<15><12> ;; CONTROL 'U'
(1) 015347 136 006507 00C012 $CNTLG: .ASCIZ /^G/<15><12> ;; CONTROL 'G'
(1) 015354 005015 053523 C20122 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 015362 020075 000
(1) 015365 040 047040 053505 $MNEW: .ASCIZ / NEW = /
(1) 015372 036440 000040
7125 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)
(1) *****
(1) :POWER DOWN ROUTINE
(1) 015376 012737 015542 000024 $PWRDN: MOV #$ILLUP,@PWRVEC ;; SET FOR FAST UP
(1) 015404 012737 000300 000026 MOV #PR6,@PWRVEC+2 ;; PRIO:6
(3) 015412 010046 MOV R0,-(SP) ;; PUSH R0 ON STACK
(3) 015414 010146 MOV R1,-(SP) ;; PUSH R1 ON STACK
(3) 015416 010246 MOV R2,-(SP) ;; PUSH R2 ON STACK
(3) 015420 010346 MOV R3,-(SP) ;; PUSH R3 ON STACK
(3) 015422 010446 MOV R4,-(SP) ;; PUSH R4 ON STACK
(3) 015424 010546 MOV R5,-(SP) ;; PUSH R5 ON STACK
(3) 015426 017746 164506 MOV @SWR,-(SP) ;; PUSH @SWR ON STACK
(1) 015432 010637 015546 MOV SP,$SAVR6 ;; SAVE SP
(1) 015436 012737 015450 000024 MOV #$PWRUP,@PWRVEC ;; SET UP VECTOR
(1) 015444 000000 HALT
(1) 015446 000776 BR .-2 ;; HANG UP
(1)
(2)
(1) *****
(1) :POWER UP ROUTINE
(1) 015450 012737 015542 000024 $PWRUP: MOV #$ILLUP,@PWRVEC ;; SET FOR FAST DOWN
(1) 015456 013706 015546 MOV $SAVR6,SP ;; GET SP
```



```

(1) 015462 005037 015546          CLR    $SAVR6          ;;WAIT LOOP FOR THE TTY
(1) 015466 005237 015546          1$:  INC    $SAVR6          ;;WAIT FOR THE INC
(1) 015472 001375                   BNE    1$              ;;OF WORD
(3) 015474 012677 164440          MOV    (SP)+,@SWR      ;;POP STACK INTO @SWR
(3) 015500 012605                   MOV    (SP)+,R5        ;;POP STACK INTO R5
(3) 015502 012604                   MOV    (SP)+,R4        ;;POP STACK INTO R4
(3) 015504 012603                   MOV    (SP)+,R3        ;;POP STACK INTO R3
(3) 015506 012602                   MOV    (SP)+,R2        ;;POP STACK INTO R2
(3) 015510 012601                   MOV    (SP)+,R1        ;;POP STACK INTO R1
(3) 015512 012600                   MOV    (SP)+,R0        ;;POP STACK INTO R0
(1) 015514 012737 015376 000024    MOV    #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 015522 012737 000300 000026    MOV    #PR6,@PWRVEC+2 ;;PRIO:6
(1) 015530 104401                   TYPE                                ;;REPORT THE POWER FAILURE
(1) 015532 015550          SPWRMG: .WORD  PWRMSG      ;;POWER FAIL MESSAGE POINTER
(1) 015534 012716          MOV    (PC)+,(SP)      ;;RESTART AT START1
(1) 015536 003066          SPWRAD: .WORD  START1   ;;RESTART ADDRESS
(1) 015540 000002          RTI
(1) 015542 000000          $ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
(1) 015544 000776          BR    .-2             ;;BEFORE THE POWER DOWN WAS COMPLETE
(1) 015546 000000          $SAVR6: 0              ;;PUT THE SP HERE
7126 015550 005015 042522 052123    PWRMSG: .ASCIZ <15><12>/RESTARTED FROM PWR FAIL/
      015556 051101 042524 020104
      015564 051106 046517 050040
      015572 051127 043040 044501
      015600 000114

7127          .EVEN
7128          .SBTTL TRAP DECODER
(1)          ;;*****
(2)          ;;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)          ;;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)          ;;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)          ;;*GO TO THAT ROUTINE.
(1)          STRAP: MOV    R0,-(SP)          ;;SAVE R0
(1) 015602 010046          MOV    2(SP),R0        ;;GET TRAP ADDRESS
(1) 015604 016600 000002          TST    -(R0)           ;;BACKUP BY 2
(1) 015610 005740          MOVB   (R0),R0         ;;GET RIGHT BYTE OF TRAP
(1) 015612 111000          ASL    R0              ;;POSITION FOR INDEXING
(1) 015614 006300          MOV    STRPAD(R0),R0   ;;INDEX TO TABLE
(1) 015616 016000 015636          RTS    R0              ;;GO TO ROUTINE
(1) 015622 000200
(1)          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)          STRAP2: MOV   (SP),-(SP)        ;;MOVE THE PC DOWN
(1) 015624 011646          MOV   4(SP),2(SP)     ;;MOVE THE PSW DOWN
(1) 015626 016666 000004 000002          RTI    ;;RESTORE THE PSW
(1) 015634 000002
(3)          .SBTTL TRAP TABLE
(3)          ;;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)          ;;*BY THE "TRAP" INSTRUCTION.
(3)          :
(3)          : ROUTINE
(3)          : -----

```

(3)	015636	015624			\$TRPAD: .WORD \$TRAP2				
(3)	015640	012612			\$TYPE	::CALL=TYPE	TRAP+1(104401)	TTY TYPEOUT ROUTINE	
(3)	015642	013366			\$TYPOC	::CALL=TYPOC	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)	
(3)	015644	013342			\$TYPOS	::CALL=TYPOS	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)	
(3)	015646	013402			\$TYPON	::CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)	
(3)	015650	013570			\$TYPDS	::CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WIT' SIGN)	
(1)									
(3)	015652	014672			\$GTSWR	::CALL=GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING	
(1)									
(3)	015654	014622			\$CKSWR	::CALL=CKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR	
(3)	015656	015104			\$RDCHR	::CALL=RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE	
(3)	015660	015224			\$RDLIN	::CALL=RDLIN	TRAP+11(104411)	TTY TYPEIN STRING ROUTINE	
7129									
7130					.SBTTL	ASCII MESSAGES			
7131					::GPA TITLED:	.ASCIZ <15><12>/CNDRDA DRV11J DIAG TEST PART 2/<15><12>			
7132	015662	041600	042116	042122	TITLED:	.ASCIZ <200>/CNDRDA DRV11J DIAG TEST PART 2/<200>		::GPA	
	015670	020101	042040	053122					
	015676	030461	020112	044504					
	015704	043501	052040	051505					
	015712	020124	040520	052122					
	015720	031040	000200						
7133					::GPA TLCABL:	.ASCIZ <15><12>/DRV11J CABLE REQ'D/<15><12>			
7134	015724	042200	053122	030461	TLCABL:	.ASCIZ <200>/DRV11J CABLE REQ'D/<200>		::GPA	
	015732	020112	040503	046102					
	015740	020105	042522	023521					
	015746	100104	000						
7135									
7136	015751	122	043505	052040	EM1:	.ASCIZ /REG TIMEOUT ER/			
	015756	046511	047505	052125					
	015764	042440	000122						
7137	015770	042522	020107	042522	EM2:	.ASCIZ 'REG READ/WRITE ER'			
	015776	042101	053457	044522					
	016004	042524	042440	000122					
7138	016012	051111	020122	042522	EM3:	.ASCIZ /IRR REG ER/			
	016020	020107	051105	000					
7139	016025	101	051103	051040	EM4:	.ASCIZ /ACR REG ER/			
	016032	043505	042440	000122					
7140	016040	046511	020122	042522	EM5:	.ASCIZ /IMR REG ER/			
	016046	020107	051105	000					
7141	016053	111	051123	051040	EM6:	.ASCIZ /ISR REG ER/			
	016060	043505	042440	000122					
7142	016066	046111	042514	040507	EM7:	.ASCIZ /ILLEGAL VECTOR MEM ADDR ER/			
	016074	020114	042526	052103					
	016102	051117	046440	046505					
	016110	040440	042104	020122					
	016116	051105	000						
7143	016121	103	044510	020120	EM10:	.ASCIZ /CHIP STAT ER/			
	016126	052123	052101	042440					
	016134	000122							
7144					::GPA EM11:	.ASCIZ /ILLEGAL INTERRUPT RECEIVED/			
7145	016136	046111	042514	040507	EM11:	.ASCIZ /ILLEGAL INTERRUPT REC'D/		::GPA	
	016144	020114	047111	042524					
	016152	051122	050125	020124					
	016160	042522	023503	000104					
7146	016166	047111	042524	051122	EM12:	.ASCIZ /INTERRUPT TEST ERROR/			
	016174	050125	020124	042524					

```

016202 052123 042440 051122
016210 051117 000
7147
7148 016213 115 046125 044524 ::GPA EM13: .ASCIZ /MULTIPLE INTERRUPTS RECEIVED/
016220 046120 020105 047111 EM13: .ASCIZ /MULTIPLE INTERRUPTS REC'D/ ::GPA
016226 042524 051122 050125
016234 051524 051040 041505
016242 042047 000
7149 016245 126 041505 020124 EM14: .ASCIZ /VECT ADDR MEM ER/
016252 042101 051104 046440
016260 046505 042440 000122
7150 016266 051105 050122 020103 DH1: .ASCIZ /ERRPC TSTNUM BUSADR EXPCT RCVD/
016274 020040 051524 047124
016302 046525 020040 052502
016310 040523 051104 020040
016316 054105 041520 020124
016324 020040 041522 042126
016332 000
7151 016333 105 051122 041520 DH2: .ASCIZ /ERRPC TSTNUM BUSADR VAM ADDR EXPCT RCVD/
016340 020040 052040 052123
016346 052516 020115 041040
016354 051525 042101 020122
016362 053040 046501 020040
016370 020040 040440 042104
016376 020122 020040 042440
016404 050130 052103 020040
016412 051040 053103 000104
7152 016420 051105 050122 020103 DH3: .ASCIZ /ERRPC TSTNUM BUSADR VAM ADDR/
016426 020040 051524 047124
016434 046525 020040 052502
016442 040523 051104 020040
016450 040526 020115 020040
016456 020040 042101 051104
016464 000
7153 016465 105 051122 041520 DH4: .ASCIZ /ERRPC TSTNUM TESTPC BUSADR VAM ADDR/
016472 052011 052123 052516
016500 004515 042524 052123
016506 041520 041011 051525
016514 042101 020122 053040
016522 046501 020040 020040
016530 040440 042104 000122
7154
7155
7156 016536 002116 002432 002122 DT1: .EVEN
$ERRPC, TSTNUM, $BDADR, $GDDAT, $BDDAT, 0
016544 002124 002126 000000
7157 016552 002116 002432 002122 DT2: $ERRPC, TSTNUM, $BDADR, VECVAL, VECLOC, $GDDAT, $BDDAT, 0
016560 002456 002452 002124
016566 002126 000000
7158 016572 002116 002432 002122 DT3: $ERRPC, TSTNUM, $BDADR, VECVAL, VECLOC, 0
016600 002456 002452 000000
7159 016606 002116 002432 002120 DT4: $ERRPC, TSTNUM, $GDADR, $BDADR, VECVAL, VECLOC, 0
016614 002122 002456 002452
016622 000000
7160
7161
7162
;:*****

```

CNRDA DRV11J DIAG TST PRT2
CNRDA.P11 10-DEC-82 14:44

MACY11 30(1046) 15-DEC-82 15:26 M 4 PAGE 59-18
ASCII MESSAGES

SEQ 0051

7163
7164
7165
7166
7167 016624 000102
7168
7169
7170
7171
7172
7173 017030 000100
7174 017230

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

7263
7264
(1) 000100 017230
(1) 000102 000300
(1) 000140 000140
(1) 000140 170000
(1) 000142 000300
(1) 017230 017230
(1) 017230 104401 017236
(1) 017234 000000
(1) 017236 005015 045514 042526
(1) 017244 020103 047111 042524
(1) 017252 051122 050125 020124
(1) 017260 020055 044504 041523
(1) 017266 047117 042516 052103
(1) 017274 046040 041524 000040
7265
7266 017302 000001
7267 000001

;THE FOLLOWING CALL WAS INCLUDED TO INIT BRKVEC AND LKVEC
POINT=. ;SAVE POINTER
.=100
\$CLKVEC ;LKVEC HANDLER
300 ;INTERRUPT HANDLER PRI
.=140 ;BRKVEC
170000 ;ODT START ADDRESS
300 ;PRIORITY
.=POINT ;RESTORE POINTER
\$CLKVEC: TYPE,CLKMES
HALT
CLKMES: .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /

;THE FOLLOWING FLAG WAS INCLUDED TO INDICATE FALCON
KXTFLAG:1
.END

INT15	012264	6989#
INT16	012270	6991#
INT17	012274	6993#
INT18	012300	6995#
INT19	012304	6997#
INT2	012200	6963#
INT20	012310	6999#
INT21	012314	7001#
INT22	012320	7003#
INT23	012324	7005#
INT24	012330	7007#
INT25	012334	7009#
INT26	012340	7011#
INT27	012344	7013#
INT28	012350	7015#
INT29	012354	7017#
INT3	012204	6965#
INT30	012360	7019#
INT31	012364	7021#
INT32	012370	7023#
INT33	012374	7025#
INT34	012400	7027#
INT35	012404	7029#
INT36	012410	7031#
INT37	012414	7033#
INT38	012420	7035#
INT39	012424	7037#
INT4	012210	6967#
INT40	012430	7039#
INT41	012434	7041#
INT42	012440	7043#
INT43	012444	7045#
INT44	012450	7047#
INT45	012454	7049#
INT46	012460	7051#
INT47	012464	7053#
INT48	012470	7055#
INT49	012474	7057#
INT5	012214	6969#
INT50	012500	7059#
INT51	012504	7061#
INT52	012510	7063#
INT53	012514	7065#
INT54	012520	7067#
INT55	012524	7069#
INT56	012530	7071#
INT57	012534	7073#
INT58	012540	7075#
INT59	012544	7077#
INT6	012220	6971#
INT60	012550	7079#
INT61	012554	7081#
INT62	012560	7083#
INT63	012564	7085#
INT7	012224	6973#
INT8	012230	6975#

STRVEC	011232	6386	6520	6759#										
SWR	002140	5743#	5853*	6796*	6804	6829	6903	7113	7123	7124*	7125*			
SWRCK	014170	7113	7118#											
SWREG	000176	5732#	5853	7124										
SWRSV	002460	5843#	6796	6903*										
SW0	= 000001	5693#												
SW00	= 000001	5693#												
SW01	= 000002	5693#												
SW02	= 000004	5693#												
SW03	= 000010	5693#												
SW04	= 000020	5693#												
SW05	= 000040	5693#												
SW06	= 000100	5693#												
SW07	= 000200	5693#												
SW08	= 000400	5693#												
SW09	= 001000	5693#												
SW1	= 000002	5693#												
SW10	= 002000	5693#												
SW11	= 004000	5693#												
SW12	= 010000	5693#	6804	6829										
SW13	= 020000	5693#												
SW14	= 040000	5693#												
SW15	= 100000	5693#												
SW2	= 000004	5693#												
SW3	= 000010	5693#												
SW4	= 000020	5693#												
SW5	= 000040	5693#												
SW6	= 000100	5693#												
SW7	= 000200	5693#												
SW8	= 000400	5693#												
SW9	= 001000	5693#												
TBITVE	= 000014	5693#												
TITLED	015662	5887	7132#											
TKVEC	= 000060	5693#												
TLCABL	015724	5888	7134#											
TOBUF	011306	5882	6780#											
TPVEC	= 000064	5693#												
TRAPVE	= 000034	5693#	5853*											
TRPALL	011704	6811	6833	6883#										
TRPCAT	011456	5907	6129	6385	6519	6824#								
TRPOUT	011666	6813	6846	6875#	6893	6913								
TRP200	012122	6890	6907	6938#										
TRTVEC	= 000014	5693#												
TSTNUM	002432	5832#	7118*	7156	7157	7158	7159							
TST1	003206	5917#												
TST2	004054	6021	6028#											
TST3	004554	6108	6116#											
TST4	005470	6234	6241#											
TST5	006462	6380#												
TST6	007430	6514#												
TYPDS	= 104405	6690	7128#											
TYPE	= 104401	5887	5888	6690	7109	7111	7112	7113	7122	7124	7125	7128#	7264	
TYPOC	= 104402	7122	7124	7128#										
TYPON	= 104404	7128#												
TYPOS	= 104403	7128#												
VBUF	017030	6390	6494	6524	6658	6771	7173#							

CNDRDA DRV11J DIAG TST PRT2
CNDRDA.P11 10-DEC-82 14:44

MACY11 30(1046) 15-DEC-82 15:26 M 5
PAGE 62-1
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0064

.SEOP	2214#	5682#	6690
.SERRO	2700#	5682#	7113
.SERRT	2896#	5682#	7122
.SMULT	4523#		
.SPOWE	4229#	5681#	7125
.SRAND	4307#		
.SRDDE	3891#		
.SRDOC	3797#	5680#	
.SREAD	3395#	5682#	7124
.SR2AZ	4958#		
.SSAVE	3969#		
.SSB2D	4771#		
.SSB2O	4874#		
.SSCOP	2454#	5682#	7123
.SSIZE	4361#		
.SSUPR	4913#		
.STRAP	4073#	5680#	7128
.STYPB	3287#		
.STYPD	3209#	5682#	7112
.STYPE	2985#	5682#	7109
.STYPO	3112#	5681#	7111
.S4OCA	972#		

. ABS. 017304 000

ERRORS DETECTED: 0

CNDRDA,CNDRDA/CR/NL:TOC=CNMAC2.SML,CNDRDA.P11
RUN-TIME: 11 12 1 SECONDS
RUN-TIME RATIO: 56/25=2.2
CORE USED: 33K (66 PAGES)