VS60

VS60 INST TST PT3

CZVSCC0

AH-9491C-MC

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

IDENTIFICATION

PRODUCT CODE: AC-9490C-MC

PRODUCT NAME: CZVSCC0 VS60 INST TST PT3

DATE: JUNE 1979

MAINTAINER: DIAGNOSTIC ENGINEERING

0.0 TABLE OF CONTENTS

1.0 ABSTRACT

THIS PROGRAM IS ONE OF A SERIES USED TO TEST STRICTLY LOGIC FUNCTIONS OF THE VS60 DISPLAY PROCESSOR. IT IS THE PRIMARY TEST OF THE INTERNAL EIGHT LEVEL STACK AND FOUR LEVEL SILO. THERE ARE ADDITIONAL TESTS WHICH CHECK THE DELTA LENGTH AND TANGENT LOGIC ALONG WITH A MEMORY TEST CONCERNED WITH OPERATING THE VS60 ABOVE 28K OF MEMORY IF AVAILABLE.

*** REV C FIXES THE GREATER-THAN-28K TEST ***

2.0 REQUIREMENTS

2.1 EQUIPMENT

A. PDP-11 COMPUTER WITH AT LEAST 8K OF MEMORY
B. I/O TERMINAL (I.E. ASR33 TTY)
C. VS60 DISPLAY PROCESSOR
D. KT11 FOR MEMORY TEST ONLY

2.2 STORAGE

THIS PROGRAM OCCUPIES LOWER 8K OF MEMORY.

3.0 LOADING PROCEDURE

NORMAL PROCEDURE FOR LOADING A BINARY PROGRAM INTO MEMORY SHOULD BE FOLLOWED.

4.0 STARTING PROCEDURE

4.1 STARTING ADDRESS 200

LOADING ADDRESS 200 AND STARTING WILL IDENTIFY THE TEST, INDICATE THE MEMORY SIZE IN THE SYSTEM, INITIALIZE THE SYSTEM, AND BEGIN TESTING.

4.2 RESTART ADDRES 204

LOADING ADDRESS 204 AND STARTING WILL INITIALIZE THE SYSTEM AND BEGIN TESTING.

5.0 SWITCH REGISTER SETTINGS

SWITCH	FUNCTION
SW15=1	HALT ON ERROR
SW14=1	LOOP ON TEST
SW13=1	INHIBIT ERROR TYPEOUTS
SW12=1	HALT AT END OF DIAGNOSTIC
SW11=1	INHIBIT ITERATIONS - NORMALLY 2000(8) AFTER 1ST PASS
SW10=1	BELL ON ERROR
SW09=1	LOOP ON ERROR
SW08=1	LOOP ON TEST IN SWR<7:0>

6.0 ERROR REPORTING

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC OR FROM THE ERROR TABLE ($ERRTB:) TOWARD THE FRONT OF THE LISTING.

6.2 ERROR DATA

*PC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*TSTNUM	TEST NUMBER WHERE THE ERROR OCCURRED
*BUSADRS	DISPLAY BUS ADDRESS WHERE THE RESULTANT DATA WAS EXPECTED
*EXPCT	DATA THAT WAS EXPECTED
*RCVD	DATA THAT WAS RECEIVED
STK SEL	INDICATES STACK LEVEL WHERE FAILURE OCCURED
GDPC	INDICATES LOW ORDER 16 BIT ADDRESS EXPECTED (MEM TST ONLY)
GDPC-HI	INDICATES HIGH ORDER 2 BIT ADDRESS EXPECTED (MEM TST ONLY)
BDPC	RECEIVED LOW ORDER 16 BIT ADDRESS (MEM TST ONLY)
BDPC-HI	RECEIVED HIGH ORDER 2 BIT ADDRESS (MEM TST ONLY)
VECTOR	INDICATES VECTOR VALUE FROM WHICH DELTA LENGTH AND TANGENT ARE CALCULATED

*NOTE: THESE ARE TYPICALLY TYPED.

7.0 MISCELLANEOUS

7.1 VS60 BUS/VECTOR ADDRESS MODIFICATION:
MODIFY LOCATION '$VECT1' IF BASE VECTOR ADDRESS IS NOT 320.
MODIFY LOCATION '$BASE' IF BASE BUS ADDRESS IS NOT 172000.

NOTE: A RESTART IS REQUIRED AFTER THE ABOVE ADDRESS MODIFICATION.
THE ABOVE LOCATIONS ARE LOCATED IN THE 'APT MAILBOX-ETABLE'.

7.2 XXDP/APT NOTES:
THIS DIAGNOSTIC IS CHAINABLE UNDER XXDP.
THIS DIAGNOSTIC INCLUDES THE 'APT SOFTWARE HOOKS' HOWEVER, THEY HAVE NOT BEEN TESTED.

7.3 POWER FAIL:
A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT WHICH TIME THE PROGRAM IS RESTARTED AT THE BEGINNING.

7.4 SINGLE VS60 TESTING:
THIS DIAGNOSTIC DOES NOT TEST MULTIPLE VS60'S.

8.0 EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT 3 SECONDS WITH NO ITERATIONS TO ABOUT 2 MINUTES WITH ITERATIONS ENABLED. EXECUTION TIME WILL INCREASE AS MEMORY IS INCREASED ABOVE 8K. AN 'END PASS' MESSAGE IS TYPED EACH PASS THRU THE DIAGNOSTIC.

9.0 PROGRAM TEST DESCRIPTIONS

SEQ 0006

9.1 STACK POINTER TESTS

STACK POINTER TESTS INCLUDE:
DECREMENT (JSR), INCREMENT (POP), STACK OVERFLOW (GREATER THAN 8 JSR'S WITHOUT A POP), STACK UNDERFLOW (GREATER THAN 8 POP'S WITHOUT A JSR), STACK OVERFLOW INTERRUPT AND STACK UNDERFLOW INTERRUPT.

9.2 STACK TESTS - JSR (PUSH)

THE STACK CONSISTS OF 8 LEVELS OF 64 BITS. EACH WRITEABLE BIT IS FORCED TO EACH STATE STARTING FROM THE 'TOP OF STACK' POSITION THRU 8 LEVELS TO THE BOTTOM OF THE STACK. FOR EXAMPLE THE TYPICAL FORMAT MIGHT BE AS FOLLOWS: 'TOP OF STACK' IS SET; AN INSTRUCTION IS SET UP AND EXECUTED WHICH SETS UP THE CONDITION UNDER TEST - FOR EXAMPLE LOAD THE VECTOR SCALE TO SOME VALUE; THEN A JSR (PUSH) INSTRUCTION IS EXECUTED - THIS WILL PUSH OR WRITE ON THE STACK THE VECTOR SCALE AT THE BITS ASSIGNED; THEN THE STACK POINTER IS SET (REG 32) TO POINT AT THE STACK LEVEL AND WORD WHERE THE VECTOR SCALE WAS STORED AND CAN BE READ FROM (REG 26); THEN IT IS CHECKED AGAINST WHAT SHOULD HAVE BEEN STORED AND IF OK THE TEST IS REPEATED AT THE NEXT STACK LEVEL.

9.3 STACK TESTS - POP (RESTORE)

THESE STACK TESTS INCLUDE EXACTLY THE SAME PROCEDURE AS 9.2 BUT INSTEAD OF THE CONDITION UNDER TEST BEING VERIFIED ON THE STACK, IT IS RESTORED VIA A POP INSTRUCTION AND VERIFIED FROM THE BUS. NOTE THAT BEFORE THE POP IS PERFORMED THE COMPLEMENT OF WHAT WAS WRITTEN ON THE STACK IS SENT TO THE FLOP/S (CONDITION) UNDER TEST. THERE ARE SEVERAL CONDITIONS (I.E. STOP INTERRUPT ENABLE, L.P. HIT DISABLE, ETC.) THAT CANNOT BE VERIFIED FROM THE BUS, IN THIS CASE AN ADDITIONAL JSR IS EXECUTED AND THE CONDITION IS CHECKED ON THE STACK.

9.4 STACK TESTS ON DPC BETWEEN 8K & 28K

THESE TWO DPC STACK TESTS ARE IDENTICAL TO THE BASIC JSR & POP DPC TESTS EXCEPT THAT THEY ARE PERFORMED AT THE HIGHEST MEMORY BOUNDRY BELOW 28K. THE INTENTION IS TO STACK PREVIOUSLY UNUSED HIGH ORDER DPC BITS. AT THE START OF THE PROGRAM A SIZER PROGRAM IS USED TO DETERMINE HIGHEST AVAILABLE MEMORY TO 28K FOR THESE TESTS. ON 8K SYSTEMS THESE TESTS ARE ABORTED.

9.5 UPPER MEMORY ADDRESS TEST - ABOVE 28K

SEQ 0007

THIS TEST REQUIRES THAT THE KT11 BE PRESENT OR THE TEST WILL BE ABORTED. THIS TEST WAS INTENDED TO EXERCISE DPC BITS 16 & 17 AT 4K MEMORY INTERVALS ABOVE 28K. THE TEST SEQUENCE FROM EACH 4K BLOCK IS A JSR, POP & LOAD STATUS B INSTRUCTION. THE GRAPH PLOT INCREMENT PORTION OF THE LOAD STATUS B INSTRUCTION CONTAINS A PAGE ADDRESS EQUAL TO ACTUAL 4K MEMORY ADDRESS ABOVE 28K BEING ADDRESSED. REFER TO TEST FOR DETAILED EXPLAINATION.

9.6 DELTA LENGTH TESTS

WITH MAINTENANCE MODE SWITCH 3 SET WHEN EXECUTING A LONG VECTOR INSTRUCTION THE X POSITION REGISTER WILL CONTAIN THE DELTA LENGTH. THE DELTA LENGTH IS THE AMOUNT OF CHANGE FROM THE POINT OF ORIGIN TO THE EDGE OF THE SCREEN NOT TO EXCEED 1777. THESE TESTS WILL VERIFY THAT THE CORRECT DELTA LENGTH FROM THE POINT OF ORIGIN CAN BE CALCULATED WITH DIFFERENT VALUES OF X OR Y.

9.7 TANGENT TESTS

WITH MAINTENANCE MODE SWITCH 3 SET WHEN EXECUTING A LONG VECTOR INSTRUCTION THE Y POSITION REGISTER WILL CONTAIN THE TANGENT. THE TANGENT IS THE OPPOSITE DIVIDED BY THE ADJACENT. THESE TESTS WILL VERIFY THE TANGENT VALUE FOR DIFFERENT VALUES OF X OR Y AT A VECTOR SCALE OF UNITY.

9.8 SILO TESTS

THE SILO TESTS DETERMINE THAT EACH LEVEL OF THE SILO CAN RESTORE THE SAVED STATUS WHEN A LIGHT PEN FLAG INTERRUPT IS FORCED (USING MAINTENTANCE MODE 3). THE ACTUAL TEST SEQUENCE MIGHT LOOK LIKE THIS; SET UP THE STATUS TO SOME UNIQUE VALUE, DO A GRAPHIC DISPLAY INSTRUCTION (THIS FORCES STATUS TO 1ST LEVEL OF SILO), THEN COMPLEMENT STATUS, THEN CAUSE A LIGHT PEN FLAG INTERRUPT (THIS WILL RESTORE THE STATUS FROM THE 1ST LEVEL OF THE SILO), THEN THE PROGRAM WILL VERIFY THAT THE STATUS WAS RESTORED PROPERLY. SINCE THE SILO ADDRESS POINTER IS ALWAYS RESET ON DISPLAY STARTS, THEN THE ABOVE SEQUENCE MUST BE REPEATED UP TO FOUR TIMES (FOUR LEVELS IN SILO) FOR ALL LEVELS TO BE TESTED. THE ONLY DIFFERENCE WILL BE THAT THE LIGHT PEN INTERRUPT WILL BE FORCED AFTER THE 2ND, 3RD, OR 4TH GRAPHIC INSTRUCTION DEPENDING ON THE SILO LEVEL UNDER TEST (EACH GRAPHIC INSTRUCTION LOADS THE SILO AND BUMPS THE SILO ADDRESS POINTER). AN ADDITIONAL STEP IS TAKEN IN SILO TEST #2 IN THE CASE WHERE THE SILO RESTORED STATUS CANNOT BE READ DIRECTLY FROM THE BUS (L.P. HIT DISABLE, STOP INTR ENA, ECT.). TO VERIFY THESE RESTORED CONDITIONS, A JSR IS PREFORMED WHICH SAVES THIS INFORMATION ON THE STACK WHERE IT IS ACCESSIBLE FROM THE BUS.

10.0 LISTING

SEQ 0008

PROGRAM LISTING FOLLOWS.

```
                                .TITLE  CZVSC-C  VS60 INSTRUCTION TEST PART III
(1)                             ;*COPYRIGHT (C) 1976
(1)                             ;*DIGITAL EQUIPMENT CORP.
(1)                             ;*MAYNARD, MASS. 01754
(1)                             ;*
(1)                             ;*PROGRAM BY R. MOORE
(1)                             ;*
(1)                             ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)                             ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1)                             ;*
                                ;
                                ; REV C -- FIXES TEST 107 TO OPERATE ABOVE 28K (G.P. JUNE '79).
                                ;
                                .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)                             ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)             000000          R0=     %0              ;;GENERAL REGISTER
(1)             000001          R1=     %1              ;;GENERAL REGISTER
(1)             000002          R2=     %2              ;;GENERAL REGISTER
(1)             000003          R3=     %3              ;;GENERAL REGISTER
(1)             000004          R4=     %4              ;;GENERAL REGISTER
(1)             000005          R5=     %5              ;;GENERAL REGISTER
(1)             000006          R6=     %6              ;;GENERAL REGISTER
(1)             000007          R7=     %7              ;;GENERAL REGISTER
(1)             000006          SP=     %6              ;;STACK POINTER
(1)             000007          PC=     %7              ;;PROGRAM COUNTER
(1)
(1)                             ;*PRIORITY LEVEL DEFINITIONS
(1)             000000          PR0=    0               ;;PRIORITY LEVEL 0
(1)             000040          PR1=    40              ;;PRIORITY LEVEL 1
(1)             000100          PR2=    100             ;;PRIORITY LEVEL 2
(1)             000140          PR3=    140             ;;PRIORITY LEVEL 3
(1)             000200          PR4=    200             ;;PRIORITY LEVEL 4
(1)             000240          PR5=    240             ;;PRIORITY LEVEL 5
(1)             000300          PR6=    300             ;;PRIORITY LEVEL 6
(1)             000340          PR7=    340             ;;PRIORITY LEVEL 7
(1)
(1)                             ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
(1)             100000          SW15=   100000
```

```
                                                                K 1
(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)
(1)                                     ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
(1)             000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
```

```
    (1)          000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
    (1)          000014                  TBITVEC=14              ;;''T'' BIT
    (1)          000014                  TRTVEC= 14              ;;TRACE TRAP
    (1)          000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
    (1)          000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
    (1)          000024                  PWRVEC= 24              ;;POWER FAIL
    (1)          000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
    (1)          000034                  TRAPVEC=34              ;;''TRAP'' TRAP
    (1)          000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
    (1)          000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
    (1)          000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
     17          172000                  ABASE=  172000
     18          100320                  AVECT1= 100320
```

```
 20                                     .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)                                     ;*        12            HALT AT END OF DIAGNOSTIC
(1)                                     ;*        11            INHIBIT ITERATIONS
(1)                                     ;*        10            BELL ON ERROR
(1)                                     ;*         9            LOOP ON ERROR
(1)                                     ;*         8            LOOP ON TEST IN SWR<7:0>
 21                                     .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)             000174                          .=174
(1)   000174  000000            DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
(1)   000176  000000            SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
(1)                                     .SBTTL  STARTING ADDRESS(ES)
(1)   000200  000137  002304            JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
 22   000204  000137  003244            JMP     RSTART          ; RESTART.
 23
 24                                     .SBTTL  ACT11 HOOKS
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;HOOKS REQUIRED BY ACT11
(1)             000210                          $SVPC=.                 ;SAVE PC
(1)             000046                          .=46
(1)   000046  024412                    $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)             000052                          .=52
(1)   000052  000000                    .WORD   0               ;;2)SET LOC.52 TO ZERO
(1)             000210                  .=$SVPC                 ;; RESTORE PC
 25
 26             001000                          .=1000
 27                                     .SBTTL  APT PARAMETER BLOCK
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                     ;;**********************************************************************
(1)             001000                  .$X=.   ;;SAVE CURRENT LOCATION
(1)             000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1)   000024  000200                    200     ;;FOR APT START UP
(1)             000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1)   000044  001000                    $APTHDR ;;POINT TO APT HEADER BLOCK
(1)             001000                  .=.$X   ;;RESET LOCATION COUNTER
(2)                                     ;;**********************************************************************
(1)                                     ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)                                     ;INTERFACE SPEC.
(1)
(1)   001000                            $APTHD:
(1)   001000  000000            $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1)   001002  001202            $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1)   001004  000012            $TSTM:  .WORD   10.     ;;RUN TIM OF LONGEST TEST
```

```
    (1)  001006  000036                $PASTM: .WORD   30.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
    (1)  001010  000000                $UNITM: .WORD   0        ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
    (1)  001012  000031                        .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
     28
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                  .WORD   0
001102     000          $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103     000          $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114     000          $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115     001          $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134     000          $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135     000          $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154     000          $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155     002          $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156     012          $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
001157     000          $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001166  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001170  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001172  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001176     077          $QUES:  .ASCII  /?/             ;;QUESTION MARK
001177     015          $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001200  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;**********************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;**********************************************************************
                                .EVEN
001202                  $MAIL:                          ;;APT MAILBOX
001202  000000          $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001204  000000          $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001206  000000          $TESTN: .WORD   ATESTN          ;;TEST NUMBER
```

```
(2)  001210  000000                $PASS:  .WORD   APASS   ;;PASS COUNT
(2)  001212  000000                $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
(2)  001214  000000                $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
(2)  001216  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(2)  001220  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001222                        $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001222      000               $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001223      000               $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001224  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001226  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001230  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                ;*                      BITS 15-11=CPU TYPE
(2)                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                                ;*                              11/70=06,PDQ=07,Q=10
(2)                                ;*                      BIT 10=REAL TIME CLOCK
(2)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001232      000               $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001233      000               $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                ;*                              900 NSEC CORE=001
(2)                                ;*                              300 NSEC BIPOLAR=002
(2)                                ;*                              500 NSEC MOS=003
(2)  001234  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
(2)  001236      000               $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001237      000               $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001240  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001242      000               $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001243      000               $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001244  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001246      000               $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001247      000               $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001250  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001252  100320                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001254  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001256  172000                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001260  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001262  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001264                        $ETEND:
(2)                                .MEXIT
```

```
   (1)                                .SBTTL  ERROR POINTER TABLE
   (1)
   (1)                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
   (1)                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
   (1)                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
   (1)                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
   (1)                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
   (1)
   (1)                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
   (1)                                ;*      DH              ;;POINTS TO THE DATA HEADER
   (1)                                ;*      DT              ;;POINTS TO THE DATA
   (1)                                ;*      DF              ;;POINTS TO THE DATA FORMAT
   (1)
   (1)
   (1)  001264                        $ERRTB:
```

```
                                ;ERROR 1 - MAINT MODE 3 & L.P. FLAG FAILED TO CAUSE AN INTERRUPT

001264  027554                         EM1                     ;L.P. FLAG INTR FAILURE
001266  031763                         DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001270  032310                         DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001272  000000                         0                       ;ALL ARE OCTAL

                                ;ERROR 2 - SILO FAILED TO RESTORE AT BUSADRS & LEVEL INDICATED

001274  027603                         EM2                     ;SILO FAILURE AT LEVEL INDICATED
001276  032232                         DH5                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    LEVEL
001300  032324                         DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001302  000000                         0

                                ;ERROR 3 - STACK POINTER FAILED TO DECREMENT PROPERLY ON JSR INSTR

001304  027643                         EM3                     ;STK PTR DECREMENT ERROR
001306  031763                         DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001310  032310                         DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001312  000000                         0

                                ;ERROR 4 - STACK OVERFLOW BIT FAILED TO SET OR RESET

001314  027662                         EM4                     ;STACK OVERFLOW ERROR
001316  031763                         DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001320  032310                         DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001322  000000                         0

                                ;ERROR 5 - JSR RELATIVE INSTR FAILED TO PUSH DPC ONTO STACK

001324  027704                         EM5                     ;DPC STACKING ERROR
001326  032030                         DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001330  032324                         DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001332  000000                         0
```

```
                                ;ERROR 6 - DPU NAME REG FAILED ON STACKING

001334  027722                          EM6                     ;DPU NAME STACKING ERROR
001336  032030                          DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001340  032324                          DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001342  000000                          0

                                ;ERROR 7 - MODE BITS FAILED ON STACKING

001344  027745                          EM7                     ;MODE STACKING ER
001346  032030                          DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001350  032324                          DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001352  000000                          0

                                ;ERROR 10 - VECTOR SCALE BITS FAILED ON STACKING

001354  027764                          EM10                    ;VECTOR SCALE STACKING ERROR
001356  032030                          DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001360  032324                          DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001362  000000                          0

                                ;ERROR 11 - CHARACTER SCALE FAILED ON STACKING

001364  030013                          EM11                    ;CHARACTER SCALE STACKING ERROR
001366  032030                          DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001370  032324                          DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001372  000000                          0

                                ;ERROR 12 - CHARACTER ROTATE FAILED ON STACKING

001374  030045                          EM12                    ;CHARACTER ROTATE STACKING ERROR
001376  032030                          DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001400  032324                          DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001402  000000                          0

                                ;ERROR 13 - INTENSITY LEVEL FAILED ON STACKING

001404  030100                          EM13                    ;INTENSITY LEVEL STACKING ERROR
001406  032030                          DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001410  032324                          DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001412  000000                          0

                                ;ERROR 14 - COLOR LEVEL FAILED ON STACKING

001414  030132                          EM14                    ;COLOR LEVEL STACKING-ER
001416  032030                          DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001420  032324                          DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001422  000000                          0
```

```
                                ;ERROR 15 - ITALICS FAILED ON STACKING

001424  030160                        EM15                    ;ITALICS STACKING ER
001426  032030                        DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001430  032324                        DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001432  000000                        0

                                ;ERROR 16 - MENU FAILED ON STACKING

001434  030202                        EM16                    ;MENU STACKING ER
001436  032030                        DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001440  032324                        DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001442  000000                        0

                                ;ERROR 17 - BLINK ENA FAILED ON STACKING

001444  030221                        EM17                    ;BLINK ENA STACKING-ER
001446  032030                        DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001450  032324                        DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001452  000000                        0

                                ;ERROR 20 - STOP INT ENA FAILED ON STACKING

001454  030245                        EM20                    ;STOP INT ENA STACKING ER
001456  032030                        DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001460  032324                        DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001462  000000                        0

                                ;ERROR 21 - L.P. HIT FAILED ON STACKING

001464  030274                        EM21                    ;LP HIT STACKING ER
001466  032030                        DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001470  032324                        DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001472  000000                        0

                                ;ERROR 22 - LOAD STATUS A OR B' INSTR FAILED WHEN ENA=0 (VERIFIED ON STACK)

001474  030325                        EM22                    ;LD STATUS STKING ER
001476  032030                        DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001500  032324                        DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001502  000000                        0

                                ;ERROR 23 - EDGE INT ENA FAILED ON STACKING

001504  030355                        EM23                    ;EDGE INT ENA STACKING ER
001506  032030                        DH2                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
001510  032324                        DT2                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
001512  000000                        0
```

```
                                                H 2
165                             ;ERROR 24 - CHARACTER STRING ESCAPE FAILED ON STACKING
166
167  001514  030404                     EM24                ;CHAR STRING ESCAPE STACKING ER
168  001516  032030                     DH2                 ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
169  001520  032324                     DT2                 ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
170  001522  000000                     0
171
172                             ;ERROR 25 - DEPTH QUEING OR DEPTH QUE CONTROL FAILED ON STACKING
173
174  001524  030441                     EM25                ;DEPTH QUE STACKING ER
175  001526  032030                     DH2                 ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
176  001530  032324                     DT2                 ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
177  001532  000000                     0
178
179                             ;ERROR 26 - LINE TYPE FAILED ON STACKING
180
181  001534  030465                     EM26                ;LINE TYPE STACKING ER
182  001536  032030                     DH2                 ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
183  001540  032324                     DT2                 ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
184  001542  000000                     0
185
186                             ;ERROR 27 - INTENSITY ENABLED FAILED ON STACKING
187
188  001544  030511                     EM27                ;INTENSITY ENA STKING ER
189  001546  032030                     DH2                 ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
190  001550  032324                     DT2                 ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
191  001552  000000                     0
192
193                             ;ERROR 30 - L.P. INTR ENABLED FAILED ON STACKING
194
195  001554  030541                     EM30                ;L.P. INTR ENA STKING ER
196  001556  032030                     DH2                 ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
197  001560  032324                     DT2                 ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
198  001562  000000                     0
199
200                             ;ERROR 31 - L.P. SWITCH INTR ENABLED FAILED ON STACKING
201
202  001564  030571                     EM31                ;L.P. SW INTR ENA STKING ER
203  001566  032030                     DH2                 ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
204  001570  032324                     DT2                 ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
205  001572  000000                     0
206
207
208                             ;ERROR 32 - SHIFT OUT FAILED ON STACKING
209
210  001574  030624                     EM32                ;SHIFT OUT STKING ER
211  001576  032030                     DH2                 ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD    STK SEL
212  001600  032324                     DT2                 ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT  $TMP0
213  001602  000000                     0
```

```
                                ;ERROR 33 - POP INSTR FAILED TO INCREMENT THE STACK POINTER

001604  030650                          EM33                    ;STK PTR INCREMENT ER
001606  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001610  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001612  000000                          0

                                ;ERROR 34 - STACK UNDERFLOW FAILED TO SET OR RESET

001614  030667                          EM34                    ;STACK UNDERFLOW ER
001616  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001620  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001622  000000                          0

                                ;ERROR 35 - DPC FAILED TO RESTORE ON POP INSTR

001624  030710                          EM35                    ;DPC POP ER
001626  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001630  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001632  000000                          0

                                ;ERROR 36 - NAME BITS FAILED TO RESTORE ON POP INSTR

001634  030723                          EM36                    ;NAME POP ER
001636  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001640  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001642  000000                          0

                                ;ERROR 37 - VECTOR SCALE FAILED TO RESTORE ON POP INSTR

001644  030737                          EM37                    ;VECTOR SCALE POP ER
001646  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001650  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001652  000000                          0

                                ;ERROR 40 - CHAR SCALE FAILED TO RESTORE ON POP INSTR

001654  030763                          EM40                    ;CHAR SCALE POP ER
001656  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001660  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001662  000000                          0

                                ;ERROR 41 - CHAR ROTATE FAILED TO RESTORE ON POP INSTR

001664  031005                          EM41                    ;CHAR ROTATE POP ER
001666  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001670  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001672  000000                          0
```

```
                                ;ERROR 42 - COLOR LEVEL FAILED TO RESTORE ON POP INSTR

001674  031030                        EM42                    ;COLOR LEVEL POP ER
001676  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001700  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001702  000000                        0

                                ;ERROR 43 - ITALICS FAILED TO RESTORE ON POP INSTR

001704  031053                        EM43                    ;ITALICS POP ER
001706  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001710  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001712  000000                        0

                                ;ERROR 44 - MENU FAILED TO RESTORE ON POP INSTR

001714  031072                        EM44                    ;MENU POP ER
001716  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001720  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001722  000000                        0

                                ;ERROR 45 - INTENSITY ENABLED FAILED TO RESTORE ON POP INSTR

001724  031106                        EM45                    ;INTENSITY ENA POP ER
001726  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001730  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001732  000000                        0

                                ;ERROR 46 - L.P. INTR ENABLED FAILED TO RESTORE ON POP INSTR

001734  031133                        EM46                    ;L.P. INTR ENA POP ER
001736  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001740  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001742  000000                        0

                                ;ERROR 47 - L.P. SWITCH INTR ENABLED FAILED TO RESTORE ON POP INSTR

001744  031160                        EM47                    ;L.P. SW INTR ENA POP ER
001746  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001750  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001752  000000                        0

                                ;ERROR 50 - LINE TYPE FAILED TO RESTORE ON POP INSTR

001754  031210                        EM50                    ;LINE TYPE POP ER
001756  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001760  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001762  000000                        0
```

```
                                ;ERROR 51 - INTENSITY LEVEL FAILED TO RESTORE ON POP INSTR

001764  031231                        EM51                    ;INTENSITY LEVEL POP ER
001766  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
001770  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
001772  000000                        0

                                ;ERROR 52 - BLINK FAILED TO RESTORE ON POP INSTR

001774  031260                        EM52                    ;BLINK POP ER
001776  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002000  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002002  000000                        0

                                ;ERROR 53 - MODE FAILED TO RESTORE ON POP INSTR

002004  031275                        EM53                    ;MODE POP ER
002006  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002010  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002012  000000                        0

                                ;ERROR 54 - STOP INTR ENABLED FAILED TO RESTORE ON POP INSTR

002014  031311                        EM54                    ;STOP INTR ENA POP ER
002016  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002020  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002022  000000                        0

                                ;ERROR 55 - DEPTH QUEING OR DEPTH QUE CONTROL FAILED TO RESTORE ON POP INSTR

002024  031336                        EM55                    ;DEPTH QUE POP ER
002026  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002030  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002032  000000                        0

                                ;ERROR 56 - EDGE INTR ENABLED FAILED TO RESTORE ON POP INSTR

002034  031357                        EM56                    ;EDGE INTR ENA POP ER
002036  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002040  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002042  000000                        0

                                ;ERROR 57 - CHAR STRING ESCAPE FAILED TO RESTORE ON POP INSTR

002044  031404                        EM57                    ;CHAR STRING ESC POP ER
002046  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002050  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002052  000000                        0
```

```
                                ;ERROR 60 - L.P. HIT DISABLE FAILED TO RESTORE ON POP INSTR

002054  031433                        EM60                    ;L.P. HIT DISABLE POP ER
002056  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002060  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002062  000000                        0

                                ;ERROR 61 - SHIFT OUT FAILED TO RESTORED ON POP INSTR

002064  031463                        EM61                    ;SHIFT OUT ER
002066  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002070  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002072  000000                        0

                                ;ERROR 62 - TERMINATE CHARACTER FAILED TO POP THE STACK

002074  031504                        EM62                    ;TERMINATE CHAR POP ER
002076  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002100  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002102  000000                        0

                                ;ERROR 63 - STACK OVERFLOW FAILED ON INTERRUPT

002104  031532                        EM63                    ;STK OVFLO INT ER
002106  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002110  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002112  000000                        0

                                ;ERROR 64 - STACK UNDERFLOW FAILED ON INTERRUPT

002114  031553                        EM64                    ;STK UNFLO INT ER
002116  031763                        DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002120  032310                        DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002122  000000                        0

                                ;ERROR 65 - INCORRECT DELTA LENGTH FOR X OR Y VECTOR INDICATED

002124  031574                        EM65                    ;DELTA LENGTH ER
002126  032165                        DH4                     ;ERRPC  TSTNUM  VECTOR  EXPCT   RCVD
002130  032360                        DT4                     ;$ERRPC TSTNUM  $TMP0   $GDDAT  $BDDAT
002132  000000                        0

                                ;ERROR 66 - TANGENT ERROR FOR X OR Y VECTOR INDICATED

002134  031614                        EM66                    ;TANGENT ER
002136  032165                        DH4                     ;ERRPC  TSTNUM  VECTOR  EXPCT   RCVD
002140  032360                        DT4                     ;$ERRPC TSTNUM  $TMP0   $GDDAT  $BDDAT
002142  000000                        0
```

```
                                ;ERROR 67 - DPC HIGH ORDER BIT 16 & 17 FAILED TO SET OR RESET

002144  031627                          EM67                    ;DPC 16-17 ER
002146  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002150  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002152  000000                          0

                                ;ERROR 70 - VS60 ADDRESS FAILURE ABOVE 28K

002154  031644                          EM70                    ;ADRS ER
002156  032110                          DH3                     ;ERRPC  TSTNUM  GD-1716 GD-DPC  BD-1716 BD-DPC
002160  032342                          DT3                     ;$ERRPC TSTNUM  G1716   $GDDAT  B1716   $BDDAT
002162  000000                          0

                                ;ERROR 71 - STOP FLAG FAILED TO SET WHEN VS60 OPERATING AT FULL SPEED

002164  031672                          EM71                    ;STOP FLAG FAILED TO SET
002166  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002170  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002172  000000                          0

                                ;ERROR 72 - START FAILED TO CLR STACK BITS & SET 'TOP OF STACK'

002174  031722                          EM72                    ;START FAILED TO SET 'TOP OF STK'
002176  031763                          DH1                     ;ERRPC  TSTNUM  BUSADR  EXPCT   RCVD
002200  032310                          DT1                     ;$ERRPC TSTNUM  $BDADR  $GDDAT  $BDDAT
002202  000000                          0
```

```
                                        ;MORE PROGRAM DEFINITIONS

            000000              OPEN=   0                       ;LOCATION WILL LATER BE REWRITTEN
            010000              MAINT1= 10000                   ;SINGLE STEP MAINTENANCE BIT - NO NPR ALLOWED
            020000              MAINT2= 20000                   ;SINGLE CYCLE MAINTENANCE BIT - ONE NPR ALLOWED

            000250              KTERRV= 250                     ;KT TRAP ADRS
            177572              KTSR0=  177572                  ;KT STATUS REG

                                        ;MORE PROGRAM TAGS

002204      000000              TSTNUM: OPEN                    ;CONTAINS TEST NO OF TEST WITH ERROR
002206      000000              DLYCNT: OPEN                    ;COUNTER USED BY DELAY ROUTINE
002210      000000              LDRSV1: OPEN                    ;LOCATION SAVES DATA OF LOC USED IN MEM TEST
002212      000000              LDRSV2: OPEN                    ;DITTO
002214      000000              MEMMAX: OPEN                    ;CONTAINS STARTING ADRS OF HIGHEST 4K
002216      000000              CENAB:  OPEN                    ;NON ZERO ENABLES THE DPC MEM TEST
002220      000000              KTENAB: OPEN                    ;NON ZERO ENABLES THE ABOVE 28K DPC MEM TEST
002222      000000              KTMAX:  OPEN                    ;CONTAINS LAST 4K PAGE ADRS AVAILABLE ABOVE 28K
002224      000000              G1716:  OPEN                    ;CONTAINS EXPECTED ADRS BITS 17 & 16
002226      000000              B1716:  OPEN                    ;CONTAINS ACTUAL ADRS BITS 17 & 16

                                        ;KT11 PAGE ADDRESS REGS & PAGE DESCRIPTOR REGS ADDRESSES

002230      172340              KIPAR0: 172340                  ;PAR 0
002232      172300              KIPDR0: 172300                  ;PDR 0
002234      172344              KIPAR2: 172344                  ;PAR 2
002236      172304              KIPDR2: 172304                  ;PDR 2
002240      172356              KIPAR7: 172356                  ;PAR 7
002242      172316              KIPDR7: 172316                  ;PDR 7

                                        ;VS60 REGISTER ADDRESS POINTERS

002244      000000              DPC:    OPEN            ;DISPLAY PROCESSOR PC
002246      000000              SREG0:  OPEN            ;STATUS REG 1
002250      000000              XPOS:   OPEN            ;X POSITION & GRAPHPLOT INC REG
002252      000000              YPOS:   OPEN            ;Y POSITION & CHAR REG
002254      000000              RLO:    OPEN            ;RELOCATE REG
002256      000000              SREG1:  OPEN            ;STATUS REG 2
002260      000000              XDOFF:  OPEN            ;X DYNAMIC OFFSET REG
002262      000000              YDOFF:  OPEN            ;Y DYNAMIC OFFSET REG
002264      000000              ANAME:  OPEN            ;ASSOCIATE NAME REG
002266      000000              CONS:   OPEN            ;CONSOLE INDICATORS & COLOR LEVEL
002270      000000              DNAME:  OPEN            ;DPU NAME REG
002272      000000              STKVAL: OPEN            ;STACK READ REG
002274      000000              TERMCH: OPEN            ;TERMINATE CHAR REG
002276      000000              STKPT:  OPEN            ;MAINTENANCE & STK PTR
002300      000000              ZPOS:   OPEN            ;Z POSOITION REG
002302      000000              ZDOFF:  OPEN            ;Z DYNAMIC OFFSET REG
```

```
  489                                   ;SOFTWARE INITIALIZATION ROUTINE
  490
  492  002304                           START:
  (1)                                   .SBTTL  INITIALIZE THE COMMON TAGS
  (1)                                   ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
  (1)  002304  012706  001100                   MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
  (1)  002310  005026                           CLR     (R6)+           ;;CLEAR MEMORY LOCATION
  (1)  002312  022706  001140                   CMP     #SWR,R6 ;;DONE?
  (1)  002316  001374                           BNE     .-6             ;;LOOP BACK IF NO
  (1)  002320  012706  001100                   MOV     #STACK,SP       ;;SETUP THE STACK POINTER
  (1)                                   ;;INITIALIZE A FEW VECTORS
  (1)  002324  012737  025116  000020           MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
  (1)  002332  012737  000340  000022           MOV     #340,@#IOTVEC+2 ;;LEVEL 7
  (1)  002340  012737  025642  000030           MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
  (1)  002346  012737  000340  000032           MOV     #340,@#EMTVEC+2 ;;LEVEL 7
  (1)  002354  012737  027134  000034           MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
  (1)  002362  012737  000340  000036           MOV     #340,@#TRAPVEC+2;LEVEL 7
  (1)  002370  012737  027204  000024           MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
  (1)  002376  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;LEVEL 7
  (1)  002404  013737  024360  024352           MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
  (1)  002412  005037  001166                   CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
  (1)  002416  005037  001170                   CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
  (1)  002422  112737  000001  001115           MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
  (1)  002430  012737  002430  001106           MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
  (1)  002436  012737  002436  001110           MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
  (2)                                   ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
  (2)                                   ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
  (2)  002444  013746  000004                   MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
  (2)  002450  012737  002504  000004           MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
  (2)  002456  012737  177570  001140           MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
  (2)  002464  012737  177570  001142           MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
  (2)  002472  022777  177777  176440           CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
  (2)  002500  001012                           BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
  (2)                                                                   ;;AND  THE HARDWARE SWR IS NOT = -1
  (2)  002502  000403                           BR      65$             ;;BRANCH IF NO TIMEOUT
  (2)  002504  012716  002512          64$:     MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
  (2)  002510  000002                           RTI
  (2)  002512  012737  000176  001140  65$:     MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
  (2)  002520  012737  000174  001142           MOV     #DISPREG,DISPLAY
  (2)  002526  012637  000004          66$:     MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
  (1)
  (2)  002532  005037  001210                   CLR     $PASS           ;;CLEAR PASS COUNT
  (2)  002536  132737  000200  001223           BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
  (2)  002544  001403                           BEQ     67$             ;;YES,USE NON-APT SWITCH
  (2)  002546  012737  001224  001140           MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
  (2)  002554                           67$:
  493  002554  005737  000042                   TST     @#42            ;TEST IF RUNNING CHAIN MODE UNDER XXDP
  494  002560  001002                           BNE     START1          ;BR IF SO
  495  002562  104401  027423                   TYPE    ,MSG1           ;GO IDENTIFY TEST
  496  002566  013700  001256          START1:  MOV     $BASE,R0        ;GET 1ST VS60 BUS ADRS
  497  002572  012701  002244                   MOV     #DPC,R1         ;INITIALIZE VS60 REG POINTER
  498  002576  010011                SETPTR:  MOV     R0,(R1)         ;SET UP THIS ADRS POINTER
  499  002600  062700  000002                   ADD     #2,R0           ;ADVANCE TO NEXT REG ADRS
  500  002604  062701  000002                   ADD     #2,R1           ;MOVE POINTER TO NEXT REG ADRS
  501  002610  020127  002304                   CMP     R1,#DPC+40      ;HAVE WE DONE ALL REG POINTERS?
  502  002614  001370                           BNE     SETPTR          ;BR IF NOT
```

```
                                CSIZR:  CLR     CENAB           ;TURN OFF DPC MEM TEST
                                        CLR     KTENAB          ;TURN OFF KT UPPER MEM TEST
                                        MOV     #2$,@#ERRVEC    ;SET UP MEM TIMEOUT SERVICE ADRS
                                        MOV     #40000,R0       ;START MEM SIZER GREATER THAN 8K
                                1$:     MOV     R0,R1           ;DO ACTUAL TESTING IN R1
                                        TST     (R1)+           ;CHECK EVEN ADRS
                                        TST     (R1)            ;CHECK ODD ADRS
                                        ADD     #20000,R0       ;BUMP IT BY 4K
                                        CMP     #160000,R0      ;ARE WE TO I/O PAGE?
                                        BNE     1$              ;BR IF NOT
                                        INC     KTENAB          ;'KTENAB' NON-ZERO MEANS KT11 SIZER REQ
                                        BR      3$              ;GO SAVE 28K MEM LIMIT
                                2$:     CMP     (R6)+,(R6)+     ;FIX STACK SINCE NO RTI ON MEM TIMEOUT
                                3$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE LOC 4 & 6
                                        SUB     #20000,R0       ;BACK UP TO LAST 4K MEM BLK
                                        MOV     R0,MEMMAX       ;SAVE HIGHEST NON-KT MEM ADRS
                                        CMP     R0,#20000       ;IS THERE MORE THAN 8K?
                                        BHI     4$              ;BR IF SO
                                        JMP     TYPCOR          ;GO TYPE CORE AMT - NOTHING OVER 8K
                                4$:     INC     CENAB           ;NON-ZERO REQ DPC MEM TEST
                                        TST     KTENAB          ;DO WE NEED TO DO KT MEM SIZER?
                                        BNE     KTSIZR          ;BR IF SO
                                        JMP     TYPCOR          ;GO TYPE CORE AMT - LESS THAN 28K
                                KTSIZR: MOV     #KTSER,@#KTERRV ;GET UP KT11 TRAP SERVICE ADRS
                                        MOV     #2$,@#ERRVEC            ;RETURN TO 2$ ON KT TIMOUT
                                        TST     @#KTSR0         ;LOOK FOR STATUS REG
                                1$:     CLR     @KIPAR0         ;SET UP FOR TRAPS TO PAGE 0
                                        MOV     #77406,@KIPDR0  ;4K PAGE LENGTH, EXPAND UP, R/W ACCESS
                                        MOV     #1600,@KIPAR2   ;START AT 28K
                                        MOV     #77406,@KIPDR2  ;4K PAGE LENGTH, EXPAND UP, R/W ACCESS
                                        MOV     #7600,@KIPAR7   ;SET UP I/O 4K PAGE ADRS
                                        MOV     #77406,@KIPDR7  ;4K PAGE LENGTH, EXPAND UP, R/W ACCESS
                                        BR      3$              ;SKIP OVER TIMEOUT STUFF
                                2$:     CMP     (R6)+,(R6)+     ;FIX STACK SINCE NO RTI
                                        CLR     KTENAB          ;THERE IS NO KT11
                                        MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE 4 & 6
                                        JMP     TYPCOR          ;GO TYPE CORE AMT - KT NOT THERE
                                3$:     MOV     #5$,@#ERRVEC    ;SET UP NON-EXISTENT MEM TIMEOUT ADRS
                                4$:     MOV     #40000,R0       ;START WITH ADRS 0 AT PAGE ADRS IN KIPAR2
                                        BIS     #1400,@#KTSR0   ;ENAB KT (MAINT)
                                        TST     (R0)+           ;IS THIS BLK THERE?
                                        TST     (R0)+           ;CK INTERLEAVING ALSO
                                        BIC     #1400,@#KTSR0   ;DISABLE KT
                                        ADD     #200,@KIPAR2    ;ADVANCE TO NEXT 4K BLK
                                        CMP     @KIPAR2,@KIPAR7 ;ARE WE OUT TO 128K?
                                        BNE     4$              ;BR IF NOT
                                        BR      6$              ;GO SET UP KTMAX
                                5$:     CLR     @#KTSR0         ;DISABLE KT
                                        CMP     (R6)+,(R6)+     ;FIX STACK SINCE NO RTI
                                6$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE ERRVEC
                                        SUB     #200,@KIPAR2    ;BACK UP TO LAST AVAILABLE BLK
                                        MOV     @KIPAR2,KTMAX   ;SAVE IT
                                        CMP     KTMAX,#1600     ;ARE WE OVER 28K?
                                        BHIS    7$              ;BR IF SO
                                        CLR     KTENAB          ;DON'T USE KT
                                        JMP     TYPCOR          ;KT THERE BUT NOTHING OVER 28K?
```

```
559  003164  013700  002222          7$:     MOV     KTMAX,R0        ;GET MAX PAGE ADRS VALUE
560  003170  006300                          ASL     R0              ;JUSTIFY RIGHT FOR TYPE
561  003172  000300                          SWAB    R0              ;
562  003174  006300                          ASL     R0              ;
563  003176  006300                          ASL     R0              ;
564  003200  000406                          BR      TYPMEM          ;GO TYPE IT
565  003202  013700  002214          TYPCOR: MOV     MEMMAX,R0       ;GET TOP 4K BLK NO BELOW 28K
566  003206  000300                          SWAB    R0              ;SET IT UP FOR TYPE
567  003210  006200                          ASR     R0              ;ADJUST FOR TYPE
568  003212  006200                          ASR     R0              ;
569  003214  006200                          ASR     R0              ;
570  003216  062700  000004          TYPMEM: ADD     #4,R0           ;POINT TO LAST ADRS
571  003222  005737  000042                  TST     @#42            ;TEST IF RUNNING CHAIN MODE UNDER XXDP
572  003226  001006                          BNE     RSTART          ;BR IF SO
573  003230  104401  027530                  TYPE    ,MSG3           ;GO TYPE 'MEMORY SIZE='
574  003234  010046                          MOV     R0,-(SP)        ;SAVE IT ON STK FOR DECIMAL TYPE
575  003236  104405                          TYPDS                   ;GO TYPE IT
576  003240  104401  027550                  TYPE    ,MSG4           ;TYPE 'K'
577  003244  000005          RSTART: RESET                           ;INITIALIZE THE VS60
578  003246  012706  001100                  MOV     #STACK,SP       ;SET UP STACK POINTER
579  003252  012737  000340  177776          MOV     #340,PSW        ;SET UP PSW TO HIGHEST LEVEL
580  003260  105037  001102                  CLRB    $TSTNM          ;CLR TEST NO. ON RESTARTS
```

```
                                        .SBTTL
                                        .SBTTL  THESE TESTS USE MAINT. SW 2
                                        .SBTTL

                                ;;**********************************************************************
                                ;*TEST 1         TEST THAT THE JSR REL INSTR WILL DECREMENT THE STACK POINTER
                                ;;**********************************************************************
003264  000004                  TST1:   SCOPE
003266  012737  000001  001206          MOV     #1,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
003274  012737  003330  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
003302  013737  002276  001122          MOV     STKPT,$BDADR    ;SET UP REG 32 ADRS
003310  012700  020040                  MOV     #20040,R0       ;SET TOP OF STACK & MAINT SW2 IN R0
003314  012737  000034  001124          MOV     #34,$GDDAT      ;WILL EXPECT PTR TO BE 34 INITIALLY
003322  012737  163000  032374          MOV     #163000,BUFFER  ;SET UP JSR INSTR
003330  010077  176742          1$:     MOV     R0,@STKPT       ;SET UP STACK PTR
003334  012777  032374  176702          MOV     #BUFFER,@DPC    ;START
003342  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
003346  000001                          BITO                    ;COUNT TO 1
003350  017737  176722  001126          MOV     @STKPT,$BDDAT   ;READ STACK POINTER
003356  042737  177703  001126          BIC     #177703,$BDDAT  ;ONLY WANT STACK POINTER - TOP OF STK SB 0
003364  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
003372  001401                          BEQ     2$              ;BR IF OK
003374  104003                          ERROR   3               ;JSR REL INSTR FAILED TO DECREMENT STACK PTR
003376  162700  000004          2$:     SUB     #4,R0           ;SET UP NEXT STK PTR VALUE IN R0
003402  162737  000004  001124          SUB     #4,$GDDAT       ;ADVANCE TO NEXT EXPECTED PTR VALUE
003410  001347                          BNE     1$              ;BR IF THIS VALUE NOT TESTED YET

                                ;;**********************************************************************
                                ;*TEST 2         TEST THAT THE STACK OVERFLOW BIT CAN SET
                                ;;**********************************************************************
003412  000004                  TST2:   SCOPE
003414  012737  000002  001206          MOV     #2,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
003422  012737  000340  177776          MOV     #340,PSW        ;SET PRIORITY TO HIGHEST LEVEL
003430  013737  002256  001122          MOV     SREG1,$BDADR    ;SET UP REG 12 ADRS
003436  012777  020040  176632          MOV     #20040,@STKPT   ;RESET THE STACK PTR
003444  005037  001124                  CLR     $GDDAT          ;EXPECT NO OVERFLOW FOR 8 JSR'S
003450  012700  000011                  MOV     #11,R0          ;SET UP COUNT FOR 9 JSR'S
003454  012737  163000  032374          MOV     #163000,BUFFER  ;SET UP JSR INSTR
003462  012777  032374  176554  1$:     MOV     #BUFFER,@DPC    ;START
003470  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
003474  000001                          BITO                    ;COUNT TO 1
003476  005300                          DEC     R0              ;COUNT STACKING OPERATION
003500  001410                          BEQ     2$              ;BR IF THIS IS OVERFLOW
003502  032777  020000  176546          BIT     #20000,@SREG1   ;SEE THAT OVERFLOW IS NOT SET
003510  001764                          BEQ     1$              ;BR IF OVERFLOW NOT SET
003512  012737  020000  001126          MOV     #20000,$BDDAT   ;INDICATE STACK OVERFLOW
003520  104004                          ERROR   4               ;STACK OVERFLOW SET PREMATURELY
003522  032777  020000  176526  2$:     BIT     #20000,@SREG1   ;IS STACK OVERFLOW SET?
003530  001006                          BNE     TST3            ;;ADVANCE TO NEXT TEST IF OK
003532  012737  020000  001124          MOV     #20000,$GDDAT   ;EXPECTED STACK OVERFLOW
003540  005037  001126                  CLR     $BDDAT          ;INDICATE IT WAS NOT SET
003544  104004                          ERROR   4               ;STACK OVERFLOW FAILED TO SET

                                ;;**********************************************************************
                                ;*TEST 3         TEST THAT START WILL SET 'TOP OF STACK'
                                ;;**********************************************************************
```

```
(2)  003546  000004                        TST3:   SCOPE
(2)  003550  012737  000003  001206                MOV     #3,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
628  003556  013737  002276  001122                MOV     STKPT,$BDADR    ;SET UP REG 32 ADRS
629  003564  012737  000040  001124                MOV     #40,$GDDAT      ;EXPECT TOP OF STACK
630  003572  012777  010037  176476                MOV     #10037,@STKPT   ;SET ALL STACK BITS & MAINT 1
631  003600  005077  176440                        CLR     @DPC            ;START - NO NPR
632  003604  017737  176466  001126                MOV     @STKPT,$BDDAT   ;READ STK PTR REG
633  003612  042737  177701  001126                BIC     #177701,$BDDAT  ;SAVE ONLY STK PTR BITS
634  003620  023737  001124  001126                CMP     $GDDAT,$BDDAT   ;DID START SET 'TOP OF STACK'?
635  003626  001401                                BEQ     TST4            ;;NEXT TEST IF SO
636  003630  104072                                ERROR   72              ;START FAILED TO SET 'TOP OF STACK'
637
638                                         ;;*****************************************************************
(3)                                         ;*TEST 4         TEST THAT THE JSR RELATIVE INSTR WILL PUSH DPC ONTO STACK
(3)                                         ;;*****************************************************************
(2)  003632  000004                        TST4:   SCOPE
(2)  003634  012737  000004  001206                MOV     #4,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
639  003642  013737  002272  001122                MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
640  003650  012737  163000  032374                MOV     #163000,BUFFER  ;SET UP JSR RELATIVE INSTR
641  003656  012700  000034                        MOV     #34,R0          ;SET UP STK SEL AFTER JSR IN R0
642  003662  012701  020040                        MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
643  003666  012737  032376  001124                MOV     #BUFFER+2,$GDDAT        ;EXPECT ADRS BUFFER+2 ON STACK
644  003674  012737  003702  001110                MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
645  003702  010177  176370                1$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
646  003706  012777  032374  176330                MOV     #BUFFER,@DPC    ;START
647  003714  004537  024446                        JSR     R5,DELAY        ;STALL FOR STACKING
648  003720  000001                                BIT0                    ;COUNT TO 1
649  003722  010077  176350                        MOV     R0,@STKPT       ;SELECT STK WORD & BYTE
650  003726  017737  176340  001126                MOV     @STKVAL,$BDDAT  ;READ STACK BYTE
651  003734  042737  000001  001126                BIC     #BIT0,$BDDAT    ;DON'T WANT LSB
652  003742  023737  001124  001126                CMP     $GDDAT,$BDDAT   ;DID THE DPC GET STORED OK?
653  003750  001403                                BEQ     2$              ;BR IF OK
654  003752  010037  001164                        MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
655  003756  104005                                ERROR   5               ;DPC FAILED TO STACK AT LEVEL INDICATED
656  003760  162701  000004                2$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
657  003764  162700  000004                        SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
658  003770  100344                                BPL     1$              ;BR IF NOT
659
660                                         ;;*****************************************************************
(3)                                         ;*TEST 5         TEST THAT THE JSR ABS INSTR WILL PUSH DPC ONTO STACK
(3)                                         ;;*****************************************************************
(2)  003772  000004                        TST5:   SCOPE
(2)  003774  012737  000005  001206                MOV     #5,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
661  004002  013737  002272  001122                MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
662  004010  012737  162000  032374                MOV     #162000,BUFFER  ;SET UP JSR ABS INSTR
663  004016  012737  032400  001124                MOV     #BUFFER+4,$GDDAT        ;EXPCT ADRS BUFFER+4 ON STACK
664  004024  012701  020040                        MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
665  004030  012700  000034                        MOV     #34,R0          ;SET UP STACK SEL AFTER JSR IN R0
666  004034  012737  004042  001110                MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
667  004042  010177  176230                1$:     MOV     R1,@STKPT       ;SET UP STACK PTR BEFORE JSR
668  004046  012777  032374  176170                MOV     #BUFFER,@DPC    ;START
669  004054  004537  024446                        JSR     R5,DELAY        ;STALL FOR STACKING
670  004060  000001                                BIT0                    ;COUNT TO 1
671  004062  005277  176156                        INC     @DPC            ;ADVANCE TO ABS ADRS
672  004066  004537  024446                        JSR     R5,DELAY        ;STALL FOR STACKING
673  004072  000001                                BIT0                    ;COUNT TO 1
```

```
 674  004074  010077  176176                    MOV     R0,@STKPT       ;SELECT STK WORD & BYTE
 675  004100  017737  176166  001126            MOV     @STKVAL,$BDDAT  ;READ STACK BYTE
 676  004106  042737  000001  001126            BIC     #BIT0,$BDDAT    ;DON'T WANT LSB
 677  004114  023737  001124  001126            CMP     $GDDAT,$BDDAT   ;DID DPC GET STORED OK?
 678  004122  001403                            BEQ     2$              ;BR IF OK
 679  004124  010037  001164                    MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
 680  004130  104005                            ERROR   5               ;DPC FAILED TO STACK AT LEVEL INDICATED
 681  004132  162701  000004          2$:       SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
 682  004136  162700  000004                    SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
 683  004142  100337                            BPL     1$              ;BR IF NOT
 684
 685
 686                                    ;;*******************************************************************
 (3)                                    ;*TEST 6         TEST THAT THE JSR RELATIVE INSTR WILL PUSH DPC NAME REG ONTO STACK
 (3)                                    ;;*******************************************************************
 (2)  004144  000004                    TST6:   SCOPE
 (1)  004146  012737  000004  001166            MOV     #4,$TIMES       ;;DO 4 ITERATIONS
 (2)  004154  012737  000006  001206            MOV     #6,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
 687  004162  012737  153777  032374            MOV     #153777,BUFFER  ;SET UP NAME INSTR WITH 3777
 688  004170  012737  163000  032376            MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
 689  004176  012737  077760  001124            MOV     #77760,$GDDAT   ;EXPECT 7776 INITIALLY
 690  004204  012737  004222  001110            MOV     #2$,$LPERR      ;SET SCOPE LOOP ADRS
 691  004212  012701  020040          1$:       MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 692  004216  012700  000035                    MOV     #35,R0          ;SET STACK SELECTION AFTER JSR IN R0
 693  004222  010177  176050          2$:       MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
 694  004226  012777  032374  176010            MOV     #BUFFER,@DPC    ;START
 695  004234  013737  002272  001122            MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
 696  004242  005277  175776                    INC     @DPC            ;DO NEXT INSTR
 697  004246  004537  024446                    JSR     R5,DELAY        ;ALLOW TIME FOR STACKING
 698  004252  000001                            BIT0                    ;COUNT TO 1
 699  004254  010077  176016                    MOV     R0,@STKPT       ;SELECT STK WORD & BYTE
 700  004260  017737  176006  001126            MOV     @STKVAL,$BDDAT  ;READ STACK BYTE
 701  004266  042737  100017  001126            BIC     #100017,$BDDAT  ;ONLY WANT NAME BITS
 702  004274  023737  001124  001126            CMP     $GDDAT,$BDDAT   ;DID THE NAME BITS GET STORED OK?
 703  004302  001403                            BEQ     3$              ;BR IF OK
 704  004304  010037  001164                    MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
 705  004310  104006                            ERROR   6               ;DPC NAME FAILED TO STACK AT LEVEL INDICATED
 706  004312  162701  000004          3$:       SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
 707  004316  162700  000004                    SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
 708  004322  100337                            BPL     2$              ;BR IF NOT
 709  004324  162737  000020  001124            SUB     #20,$GDDAT      ;ADVANCE TO NEXT PATTERN
 710  004332  100403                            BMI     TST7            ;;ADVANCE TO NEXT TEST IF ALL PATTERNS TESTED
 711  004334  005337  032374                    DEC     BUFFER          ;SET UP NAME INSTR WITH NEXT PATTERN
 712  004340  000724                            BR      1$              ;GO TRY NEXT PATTERN
 713
 714                                    ;;*******************************************************************
 (3)                                    ;*TEST 7         TEST THAT THE MODE BITS GET PUSHED ON THE STACK
 (3)                                    ;;*******************************************************************
 (2)  004342  000004                    TST7:   SCOPE
 (1)  004344  012737  000010  001166            MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 (2)  004352  012737  000007  001206            MOV     #7,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
 715  004360  005037  001124                    CLR     $GDDAT          ;INITIALLY EXPECT ZERO FOR MODE
 716  004364  005002                            CLR     R2              ;SET INSTR TO CHAR MODE INITIALLY
 717  004366  012737  163000  032376            MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
 718  004374  012737  004412  001110            MOV     #2$,$LPERR      ;SET UP LOOP ADRS
 719  004402  012701  020040          1$:       MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
```

```
   720  004406  012700  000035            MOV     #35,R0          ;SET UP STK LEVEL AFTER JSR IN R0
   721  004412  010177  175660      2$:   MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
   722  004416  012737  100000  032374    MOV     #100000,BUFFER  ;SET UP INSTR
   723  004424  050237  032374            BIS     R2,BUFFER       ;ADD IN CURRENT INSTR MODE
   724  004430  012777  032374  175606    MOV     #BUFFER,@DPC    ;START
   725  004436  013737  002272  001122    MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
   726  004444  005277  175574            INC     @DPC            ;RESUME
   727  004450  004537  024446            JSR     R5,DELAY        ;STALL FOR STACKING
   728  004454  000001                    BIT0                    ;COUNT TO 1
   729  004456  010077  175614            MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
   730  004462  017737  175604  001126    MOV     @STKVAL,$BDDAT  ;READ MODE BITS
   731  004470  042737  177760  001126    BIC     #177760,$BDDAT  ;DON'T WANT NAME BITS
   732  004476  023737  001124  001126    CMP     $GDDAT,$BDDAT   ;ARE THE MODE BITS CORRECT?
   733  004504  001403                    BEQ     3$              ;BR IF OK
   734  004506  010037  001164            MOV     R0,$TMP0        ;SET UP STK LEVEL BEFORE ER TYPE
   735  004512  104007                    ERROR   7               ;MODE FAILED TM STACK AT LEVEL INDICATED
   736  004514  162701  000004      3$:   SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
   737  004520  162700  000004            SUB     #4,R0           ;HAS THIS INSTR MODE BEEN WRITTEN THRU STACK?
   738  004524  100332                    BPL     2$              ;BR IF NOT
   739  004526  022702  044000            CMP     #44000,R2       ;IS THIS THE ABS VECTOR INSTR?
   740  004532  001405                    BEQ     TST10           ;;ADVANCE TO NEXT TEST IF SPARE HAS BEEN TESTED
   741  004534  062702  004000            ADD     #4000,R2        ;ADVANCE TO NEXT INSTR MODE
   742  004540  005237  001124            INC     $GDDAT          ;ADVANCE TO NEXT MODE EXPECTED
   743  004544  000716                    BR      1$              ;TRY NEXT INSTR
   744
   745                              ;;*****************************************************************
   (3)                              ;*TEST 10       TEST THAT THE VECTOR SCALE BITS GET PUSHED ON THE STACK
   (3)                              ;;*****************************************************************
   (2)  004546  000004              TST10:  SCOPE
   (1)  004550  012737  000010  001166    MOV     #10,$TIMES      ;;DO 10 ITERATIONS
   (2)  004556  012737  000010  001206    MOV     #10,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
   746  004564  012737  074000  001124    MOV     #74000,$GDDAT   ;EXPECT ALL ONES INITIALLY
   747  004572  012737  154037  032374    MOV     #154037,BUFFER  ;SET UP LOAD STATUS C INSTR
   748  004600  012737  163000  032376    MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
   749  004606  012737  004624  001110    MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
   750  004614  012701  020040      1$:   MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
   751  004620  012700  000036            MOV     #36,R0          ;SET UP STK LEVEL AFTER JSR IN R0
   752  004624  010177  175446      2$:   MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
   753  004630  012777  032374  175406    MOV     #BUFFER,@DPC    ;START
   754  004636  013737  002272  001122    MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
   755  004644  005277  175374            INC     @DPC            ;ADVANCE TO JSR INSTR
   756  004650  004537  024446            JSR     R5,DELAY        ;STALL FOR STACKING
   757  004654  000001                    BIT0                    ;COUNT TO 1
   758  004656  010077  175414            MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
   759  004662  017737  175404  001126    MOV     @STKVAL,$BDDAT  ;READ VECTOR SCALE FROM STACK
   760  004670  042737  103777  001126    BIC     #103777,$BDDAT  ;ONLY WANT VECTOR SCALE BITS
   761  004676  023737  001124  001126    CMP     $GDDAT,$BDDAT   ;ARE THEY CORRECT?
   762  004704  001403                    BEQ     3$              ;BR IF OK
   763  004706  010037  001164            MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
   764  004712  104010                    ERROR   10              ;VECTOR SCALE FAILED TO STACK AT LEVEL INDICATED
   765  004714  162701  000004      3$:   SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
   766  004720  162700  000004            SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
   767  004724  100337                    BPL     2$              ;BR IF NOT
   768  004726  162737  004000  001124    SUB     #4000,$GDDAT    ;ADVANCE TO NEXT VECTOR SCALE PATTERN
   769  004734  100403                    BMI     TST11           ;;ADVANCE TO NEXT TEST IF ALL PATTERNS TESTED
   770  004736  005337  032374            DEC     BUFFER          ;UPDATE INSTRUCTION
```

```
   771  004742  000724                          BR      1$              ;TRY NEXT VECTOR SCALE VALUE
   772
   773                                  ;;*****************************************************************
   (3)                                  ;*TEST 11       TEST THAT THE CHARACTER SCALE BITS GET PUSHED ON THE STACK
   (3)                                  ;;*****************************************************************
   (2)  004744  000004                  TST11:  SCOPE
   (1)  004746  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
   (2)  004754  012737  000011  001206          MOV     #11,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
   774  004762  012737  003000  001124          MOV     #3000,$GDDAT    ;EXPECT ALL ONES INITIALLY
   775  004770  012737  154340  032374          MOV     #154340,BUFFER  ;SET UP LOAD STATUS C INSTR
   776  004776  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
   777  005004  012737  005022  001110          MOV     #2$,$LPERR      ;SET UP LOOP ADRS
   778  005012  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
   779  005016  012700  000036                  MOV     #36,R0          ;SET UP STK LEVEL AFTER JSR IN R0
   780  005022  010177  175250          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
   781  005026  012777  032374  175210          MOV     #BUFFER,@DPC    ;START
   782  005034  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
   783  005042  005277  175176                  INC     @DPC            ;ADVANCE TO JSR INSTR
   784  005046  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
   785  005052  000001                          BIT0                    ;COUNT TO 1
   786  005054  010077  175216                  MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
   787  005060  017737  175206  001126          MOV     @STKVAL,$BDDAT  ;READ CHARACTER SCALE
   788  005066  042737  174777  001126          BIC     #174777,$BDDAT  ;ONLY WANT CHAR SCALE
   789  005074  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;ARE THEY CORRECT?
   790  005102  001403                          BEQ     3$              ;BR IF OK
   791  005104  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
   792  005110  104011                          ERROR   11              ;CHARACTER SCALE FAILED TO STACK AT LEVEL INDICATED
   793  005112  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
   794  005116  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
   795  005122  100337                          BPL     2$              ;BR IF NOT
   796  005124  162737  001000  001124          SUB     #1000,$GDDAT    ;ADVANCE CHARACTER SCALE PATTERN
   797  005132  100404                          BMI     TST12           ;;ADVANCE TO NEXT TEST IF ALL PATTERNS TESTED
   798  005134  162737  000040  032374          SUB     #40,BUFFER      ;UPDATE INSTRUCTION
   799  005142  000723                          BR      1$              ;TRY NEXT CHARACTER SCALE VALUE
   800
   801                                  ;;*****************************************************************
   (3)                                  ;*TEST 12       TEST THAT THE CHARACTER ROTATE BIT GETS PUSHED ON THE STACK
   (3)                                  ;;*****************************************************************
   (2)  005144  000004                  TST12:  SCOPE
   (2)  005146  012737  000012  001206          MOV     #12,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
   802  005154  012737  000200  001124          MOV     #200,$GDDAT     ;EXPECT IT TO BE SET INITIALLY
   803  005162  012737  155400  032374          MOV     #155400,BUFFER  ;SET UP LOAD STATUS C INSTR
   804  005170  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
   805  005176  012737  005214  001110          MOV     #2$,$LPERR      ;SET UP LOOP ADRS
   806  005204  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
   807  005210  012700  000036                  MOV     #36,R0          ;SET UP STK LEVEL AFTER JSR IN R0
   808  005214  010177  175056          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
   809  005220  012777  032374  175016          MOV     #BUFFER,@DPC    ;START
   810  005226  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
   811  005234  005277  175004                  INC     @DPC            ;ADVANCE TO JSR INSTR
   812  005240  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
   813  005244  000001                          BIT0                    ;COUNT TO 1
   814  005246  010077  175024                  MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
   815  005252  017737  175014  001126          MOV     @STKVAL,$BDDAT  ;READ CHARACTER ROTATE BIT
   816  005260  042737  177577  001126          BIC     #177577,$BDDAT  ;ONLY WANT CHARACTER ROTATE BIT
   817  005266  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
```

```
                                        BEQ     3$              ;BR IF OK
                                        MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
                                        ERROR   12              ;CHARACTER ROTATE FEILED TO STACK AT LEVEL INDICATED
                                3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
                                        SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
                                        BPL     2$              ;BR IF NOT
                                        SUB     #400,BUFFER     ;UPDATE INSTR
                                        SUB     #200,$GDDAT     ;EXPECT ZERO NEXT
                                        BPL     1$              ;BR IF RESET STATE NOT TESTED

                                ;;*******************************************************************
                                ;*TEST 13        TEST THAT THE INTENSITY LEVEL BITS GET PUSHED ON THE STACK
                                ;;*******************************************************************
                                TST13:  SCOPE
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                        MOV     #13,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                                        MOV     #160,$GDDAT     ;EXPECT ALL BITS INITIALLY
                                        MOV     #103600,BUFFER  ;SET UP ALL INTENSITY LEVEL BITS INITIALLY
                                        MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
                                        MOV     #2$,$LPERR      ;SET UP LOOP ADRS
                                1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
                                        MOV     #36,R0          ;SET UP STK LEVEL AFTER JSR IN R0
                                2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
                                        MOV     #BUFFER,@DPC    ;START
                                        MOV     STKVAL,$BDADR   ;SET UP REG ADRS26
                                        MOV     #BUFFER+2,@DPC  ;START
                                        JSR     R5,DELAY        ;STALL FOR STACKING
                                        BIT0                    ;COUNT TO 1
                                        MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
                                        MOV     @STKVAL,$BDDAT  ;READ INTENSITY LEVEL
                                        BIC     #177617,$BDDAT  ;ONLY WANT INTENISTY LEVEL BITS
                                        CMP     $GDDAT,$BDDAT   ;ARE THEY CORRECT?
                                        BEQ     3$              ;BR IF OK
                                        MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
                                        ERROR   13              ;INTENSITY LEVEL FAILED TO STACK AT LEVEL INDICATED
                                3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
                                        SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
                                        BPL     2$              ;BR IF NOT
                                        SUB     #200,BUFFER     ;SET UP NEXT INTENSITY LEVEL
                                        SUB     #20,$GDDAT      ;ADVANCE TO NEXT INTENSITY LEVEL
                                        BPL     1$              ;BR IF RESET STATE NOT TESTED

                                ;;*******************************************************************
                                ;*TEST 14        TEST THAT THE COLOR LEVEL BITS GET PUSHED ON THE STACK
                                ;;*******************************************************************
                                TST14:  SCOPE
                                        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
                                        MOV     #14,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                                        MOV     #14,$GDDAT      ;EXPECT ALL BIT INITIALLY
                                        MOV     #175600,BUFFER  ;SET UP ALL COLOR LEVEL BITS INITIALLY
                                        MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
                                        MOV     #1$,$LPERR      ;SET UP LOOP ADRS
                                1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
                                        MOV     #36,R0          ;SET UP STK LEVEL AFTER JSR IN R0
                                2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
                                        MOV     #BUFFER,@DPC    ;START
```

```
  864  005624  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
  865  005632  005277  174406                  INC     @DPC            ;ADVANCE TO JSR INSTR
  866  005636  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
  867  005642  000001                          BIT0                    ;COUNT TO 1
  868  005644  010077  174426                  MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
  869  005650  017737  174416  001126          MOV     @STKVAL,$BDDAT  ;READ COLOR LEVEL FROM STACK
  870  005656  042737  177763  001126          BIC     #177763,$BDDAT  ;SAVE ONLY COLOR LEVEL
  871  005664  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;ARE THEY CORRECT?
  872  005672  001403                          BEQ     3$              ;BR IF SO
  873  005674  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
  874  005700  104014                          ERROR   14              ;COLOR LEVEL FAILED TO STACK AT LEVEL INDICATED
  875  005702  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
  876  005706  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
  877  005712  100337                          BPL     2$              ;BR IF NOT
  878  005714  162737  000200  032374          SUB     #200,BUFFER     ;ADVANCE COLOR LEVEL AT INSTR LOC
  879  005722  162737  000004  001124          SUB     #4,$GDDAT       ;ADVANCE COLOR LEVEL AT EXPECTED LOC
  880  005730  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
  881
  882                                  ;;*******************************************************************
  (3)                                  ;*TEST 15       TEST THAT ITALICS GETS PUSHED ON THE STACK
  (3)                                  ;;*******************************************************************
  (2)  005732  000004                  TST15:  SCOPE
  (2)  005734  012737  000015  001206          MOV     #15,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
  883  005742  012737  100000  001124          MOV     #100000,$GDDAT  ;EXPECT IT TO BE SET INITIALLY
  884  005750  012737  170060  032374          MOV     #170060,BUFFER  ;SET UP ITALICS AT INSTR LOC
  885  005756  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
  886  005764  012737  006002  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
  887  005772  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
  888  005776  012700  000036                  MOV     #36,R0          ;SET UP STK LEVEL AFTER JSR IN R0
  889  006002  010177  174270          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
  890  006006  012777  032374  174230          MOV     #BUFFER,@DPC    ;START
  891  006014  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
  892  006022  005277  174216                  INC     @DPC            ;ADVANCE TO JSR INSTR
  893  006026  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
  894  006032  000001                          BIT0                    ;COUNT TO 1
  895  006034  010077  174236                  MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
  896  006040  017737  174226  001126          MOV     @STKVAL,$BDDAT  ;READ ITALICS FROM STACK
  897  006046  042737  077777  001126          BIC     #77777,$BDDAT   ;SAVE AONLY ITALICS BIT
  898  006054  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
  899  006062  001403                          BEQ     3$              ;BR IF SO
  900  006064  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
  901  006070  104015                          ERROR   15              ;ITALICS FAILED TO STACK AT LEVEL INDICATED
  902  006072  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
  903  006076  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
  904  006102  100337                          BPL     2$              ;BR IF NOT
  905  006104  162737  000020  032374          SUB     #20,BUFFER      ;RESET ITALICS BIT AT INSTR LOC
  906  006112  162737  100000  001124          SUB     #100000,$GDDAT  ;RESET ITALICS AT EXPECTED LOCATION
  907  006120  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
  908
  909                                  ;;*******************************************************************
  (3)                                  ;*TEST 16       TEST THAT MENU GETS PUSHED ON THE STACK
  (3)                                  ;;*******************************************************************
  (2)  006122  000004                  TST16:  SCOPE
  (2)  006124  012737  000016  001206          MOV     #16,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
  910  006132  012737  020000  001124          MOV     #20000,$GDDAT   ;EXPECT IT TO BE SET INITIALLY
  911  006140  012737  170003  032374          MOV     #170003,BUFFER  ;SET UP MENU AT INSTR LOC
```

```
   912  006146  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
   913  006154  012737  006172  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
   914  006162  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
   915  006166  012700  000037                  MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
   916  006172  010177  174100          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
   917  006176  012777  032374  174040          MOV     #BUFFER,@DPC    ;START
   918  006204  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
   919  006212  005277  174026                  INC     @DPC            ;ADVANCE TO JSR INSTR
   920  006216  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
   921  006222  000001                          BIT0                    ;COUNT TO 1
   922  006224  010077  174046                  MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
   923  006230  017737  174036  001126          MOV     @STKVAL,$BDDAT  ;READ MENU BIT
   924  006236  042737  157777  001126          BIC     #157777,$BDDAT  ;SAVE ONLY MENU BIT
   925  006244  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
   926  006252  001403                          BEQ     3$              ;BR IF SO
   927  006254  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
   928  006260  104016                          ERROR   16              ;MENU FAILED TO STACK AT LEVEL INDICATED
   929  006262  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
   930  006266  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
   931  006272  100337                          BPL     2$              ;BR IF NOT
   932  006274  005337  032374                  DEC     BUFFER          ;RESET MENU BIT AT INSTR LOC
   933  006300  162737  020000  001124          SUB     #20000,$GDDAT   ;RESET MENU BIT AT EXPECTED LOC
   934  006306  100325                          BPL     1$              ;BR IF RESET STATE NOT TESTED
   935
   936                                  ;;*******************************************************************
   (3)                                  ;*TEST 17       TEST THAT BLINK ENA GETS PUSHED ON THE STACK
   (3)                                  ;;*******************************************************************
   (2)  006310  000004                  TST17:  SCOPE
   (2)  006312  012737  000017  001206          MOV     #17,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
   937  006320  012737  100000  001124          MOV     #100000,$GDDAT  ;EXPECT IT TO BE SET INITIALLY
   938  006326  012737  100030  032374          MOV     #100030,BUFFER  ;SET UP BLINK ENA AT INSTR LOC
   939  006334  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
   940  006342  012737  006360  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
   941  006350  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
   942  006354  012700  000037                  MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
   943  006360  010177  173712          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
   944  006364  012777  032374  173652          MOV     #BUFFER,@DPC    ;START
   945  006372  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
   946  006400  012777  032376  173636          MOV     #BUFFER+2,@DPC  ;START
   947  006406  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
   948  006412  000001                          BIT0                    ;COUNT TO 1
   949  006414  010077  173656                  MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
   950  006420  017737  173646  001126          MOV     @STKVAL,$BDDAT  ;READ BLINK ENA BIT
   951  006426  042737  077777  001126          BIC     #77777,$BDDAT   ;SAVE ONLY BLINK ENA
   952  006434  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
   953  006442  001403                          BEQ     3$              ;BR IF SO
   954  006444  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
   955  006450  104017                          ERROR   17              ;BLINK ENA FAILED TO STACK AT LEVEL INDICATED
   956  006452  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
   957  006456  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
   958  006462  100336                          BPL     2$              ;BR IF NOT
   959  006464  162737  000010  032374          SUB     #10,BUFFER      ;RESET BLINK ENA BIT INSTR LOC
   960  006472  162737  100000  001124          SUB     #100000,$GDDAT  ;RESET BLINK ENA AT EXPECTED LOC
   961  006500  100323                          BPL     1$              ;BR IF RESET STATE NOT TESTED
   962
   963                                  ;;*******************************************************************
```

```
(3)                                     ;*TEST 20       TEST THAT STOP INTR ENA GETS PUSHED ON THE STACK
(3)                                     ;;*****************************************************************
(2)  006502  000004                     TST20:  SCOPE
(2)  006504  012737  000020  001206             MOV     #20,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
964  006512  012737  000001  001124             MOV     #1,$GDDAT       ;EXPECT IT TO BE SET INITIALLY
965  006520  012737  171400  032374             MOV     #171400,BUFFER  ;SET UP STOP INTR ENA AT INSTR LOC
966  006526  012737  163000  032376             MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
967  006534  012737  006552  001110             MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
968  006542  012701  020040             1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
969  006546  012700  000037                     MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
970  006552  010177  173520             2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
971  006556  012777  032374  173460             MOV     #BUFFER,@DPC    ;START
972  006564  013737  002272  001122             MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
973  006572  005277  173446                     INC     @DPC            ;ADVANCE TO JSR INSTR
974  006576  004537  024446                     JSR     R5,DELAY        ;STALL FOR STACKING
975  006602  000001                             BIT0                    ;COUNT TO 1
976  006604  010077  173466                     MOV     R0,@STKPT       ;SELECT WORD & BYTE
977  006610  017737  173456  001126             MOV     @STKVAL,$BDDAT  ;READ STOP INTR ENA WORD
978  006616  042737  177776  001126             BIC     #177776,$BDDAT  ;SAVE ONLY STOP INTR ENA BIT
979  006624  023737  001124  001126             CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
980  006632  001403                             BEQ     3$              ;BR IF SO
981  006634  010037  001164                     MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
982  006640  104020                             ERROR   20              ;STOP INTR ENA FAILED TO STACK AT LEVEL INDICATED
983  006642  162701  000004             3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
984  006646  162700  000004                     SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
985  006652  100337                             BPL     2$              ;BR IF NOT
986  006654  162737  000400  032374             SUB     #400,BUFFER     ;RESET STOP INTR ENA AT INSTR LOC
987  006662  005337  001124                     DEC     $GDDAT          ;RESET STOP INTR ENA AT EXPECTED LOC
988  006666  100325                             BPL     1$              ;BR IF RESET STATE NOT TESTED
989
990                                     ;;*****************************************************************
(3)                                     ;*TEST 21       TEST THAT L.P. HIT DISABLE GETS PUSHED ON THE STACK
(3)                                     ;;*****************************************************************
(2)  006670  000004                     TST21:  SCOPE
(2)  006672  012737  000021  001206             MOV     #21,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
991  006700  012737  000002  001124             MOV     #2,$GDDAT       ;EXPECT IT TO BE SET INITIALLY
992  006706  012737  170300  032374             MOV     #170300,BUFFER  ;SET UP HIT DISABLE AT INSTR LOC
993  006714  012737  163000  032376             MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
994  006722  012737  006740  001110             MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
995  006730  012701  020040             1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
996  006734  012700  000037                     MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
997  006740  010177  173332             2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
998  006744  012777  032374  173272             MOV     #BUFFER,@DPC    ;START
999  006752  013737  002272  001122             MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
1000 006760  005277  173260                     INC     @DPC            ;ADVANCE TO JSR INSTR
1001 006764  004537  024446                     JSR     R5,DELAY        ;STALL FOR STACKING
1002 006770  000001                             BIT0                    ;COUNT TO 1
1003 006772  010077  173300                     MOV     R0,@STKPT       ;SELECT WORD & BYTE
1004 006776  017737  173270  001126             MOV     @STKVAL,$BDDAT  ;READ LP HIT DISABLE WORD
1005 007004  042737  177775  001126             BIC     #177775,$BDDAT  ;SAVE ONLY LP HIT DISABLE BIT
1006 007012  023737  001124  001126             CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
1007 007020  001403                             BEQ     3$              ;BR IF SO
1008 007022  010037  001164                     MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
1009 007026  104021                             ERROR   21              ;L.P. HIT DISABLE FAILED TO STACK AT LEVEL INDICATED
1010 007030  162701  000004             3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
1011 007034  162700  000004                     SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
```

```
 1012  007040  100337                          BPL     2$                      ;BR IF NOT
 1013  007042  162737  000100  032374          SUB     #100,BUFFER             ;RESET LP HIT DISABLE BIT AT INSTR LOC
 1014  007050  162737  000002  001124          SUB     #2,$GDDAT               ;RESET LP HIT DISABLE BIT AT EXPECTED LOC
 1015  007056  100324                          BPL     1$                      ;BR IF RESET STATE NOT TESTED
 1016                                  ;;**********************************************************************
  (3)                                  ;*TEST 22       TEST THAT STOP INTR & L.P. HIT ENABLES DON'T CHANGE WHEN CHANGE ENA=0
  (3)                                  ;;**********************************************************************
  (2)  007060  000004                  TST22:  SCOPE
  (2)  007062  012737  000022  001206          MOV     #22,$TESTN              ;;SET TEST NUMBER IN APT MAIL BOX
 1017  007070  012737  000003  001124          MOV     #3,$GDDAT               ;EXPECT STOP INTR ENABLED & L.P. HIT FROM STK
 1018  007076  012737  171700  032374          MOV     #171700,BUFFER          ;SET UP LOAD STATUS A INSTR
 1019  007104  012737  163000  032376          MOV     #163000,BUFFER+2                ;SET UP JSR REL INSTR
 1020  007112  012777  020040  173156  1$:     MOV     #20040,@STKPT           ;SET 'TOP OF STACK'
 1021  007120  012777  032374  173116          MOV     #BUFFER,@DPC            ;START
 1022  007126  013737  002272  001122          MOV     STKVAL,$BDADR           ;SET UP REG ADRS 26
 1023  007134  005277  173104                  INC     @DPC                    ;PICK UP JSR INSTR
 1024  007140  004537  024446                  JSR     R5,DELAY                ;STALL FOR STACKING
 1025  007144  000001                          BIT0                            ;COUNT TO ONE
 1026  007146  012777  020037  173122          MOV     #20037,@STKPT           ;SET STACK PTR
 1027  007154  017737  173112  001126          MOV     @STKVAL,$BDDAT          ;READ STACK
 1028  007162  042737  177774  001126          BIC     #177774,$BDDAT          ;SAVE ONLY STOP INTR & L.P. HIT
 1029  007170  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;SHOULD STAY SET ONCE CHANGE ENABLES ARE OFF
 1030  007176  001404                          BEQ     2$                      ;BR IF CORRECT
 1031  007200  012737  000037  001164          MOV     #37,$TMP0               ;SAVE STACK LEVEL FOR TYPE
 1032  007206  104022                          ERROR   22                      ;LD STATUS A STROBED WHEN CHANGE ENA=0
 1033  007210  022737  171700  032374  2$:     CMP     #171700,BUFFER          ;IS THIS THE 2ND PASS?
 1034  007216  001004                          BNE     TST23                   ;;GO TO NEXT TEST IF TESTED WITH CHANGE ENA=0
 1035  007220  042737  001700  032374          BIC     #1700,BUFFER            ;CLR CHANGE ENABLES FOR 2ND PASS THIS TEST
 1036  007226  000731                          BR      1$                      ;GO REPEAT TEST WITH CHANGE ENABLES OFF
 1037
 1038
 1039                                  ;;**********************************************************************
  (3)                                  ;*TEST 23       TEST THAT EDGE INT ENA GETS PUSHED ON THE STACK
  (3)                                  ;;**********************************************************************
  (2)  007230  000004                  TST23:  SCOPE
  (2)  007232  012737  000023  001206          MOV     #23,$TESTN              ;;SET TEST NUMBER IN APT MAIL BOX
 1040  007240  012737  004000  001124          MOV     #4000,$GDDAT            ;EXPECT IT TO BE SET INITIALLY
 1041  007246  012737  176060  032374          MOV     #176060,BUFFER          ;SET UP EDGE INT ENA AT INSTR LOC
 1042  007254  012737  163000  032376          MOV     #163000,BUFFER+2                ;SET UP JSR INSTR
 1043  007262  012737  007300  001110          MOV     #2$,$LPERR              ;SET UP SCOPE LOOP ADRS
 1044  007270  012701  020040          1$:     MOV     #20040,R1               ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 1045  007274  012700  000037                  MOV     #37,R0                  ;SET UP STK LEVEL AFTER JSR IN R0
 1046  007300  010177  172772          2$:     MOV     R1,@STKPT               ;SET STK PTR BEFORE JSR
 1047  007304  012777  032374  172732          MOV     #BUFFER,@DPC            ;START
 1048  007312  013737  002272  001122          MOV     STKVAL,$BDADR           ;SET UP REG ADRS 26
 1049  007320  005277  172720                  INC     @DPC                    ;ADVANCE TO JSR INSTR
 1050  007324  004537  024446                  JSR     R5,DELAY                ;STALL FOR STACKING
 1051  007330  000001                          BIT0                            ;COUNT TO 1
 1052  007332  010077  172740                  MOV     R0,@STKPT               ;SELECT WORD & BYTE
 1053  007336  017737  172730  001126          MOV     @STKVAL,$BDDAT          ;READ STACK WORD
 1054  007344  042737  173777  001126          BIC     #173777,$BDDAT          ;SAVE ONLY EDGE INT ENA BIT
 1055  007352  023737  001124  001126          CMP     $GDDAT,$BDDAT           ;IS IT CORRECT?
 1056  007360  001403                          BEQ     3$                      ;BR IF SO
 1057  007362  010037  001164                  MOV     R0,$TMP0                ;SET UP STK LEVEL FOR ER TYPE
 1058  007366  104023                          ERROR   23                      ;EDGE INTR ENA FAILED TO STACK AT LEVEL INDICATED
 1059  007370  162701  000004          3$:     SUB     #4,R1                   ;ADVANCE TO NEXT STK LEVEL
```

```
                        SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
                        BPL     2$              ;BR IF NOT
                        SUB     #20,BUFFER      ;RESET EDGE INT ENA BIT AT INSTR LOC
                        SUB     #4000,$GDDAT    ;RESET EDGE INT ENA BIT AT EXPECTED LOC
                        BPL     1$              ;BR IF RESET STATE NOT TESTED

                ;;*****************************************************************
                ;*TEST 24        TEST THAT CHAR STRING ESCAPE BIT GETS PUSHED ON THE STACK
                ;;*****************************************************************
                TST24:  SCOPE
                        MOV     #24,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                        MOV     #10000,$GDDAT   ;EXPECT IT TO BE SET INITIALLY
                        MOV     #176003,BUFFER  ;SET UP CHAR STRING ESCAPE ENA AT INSTR LOC
                        MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
                        MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
                1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
                        MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
                2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
                        MOV     #BUFFER,@DPC    ;START
                        MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
                        INC     @DPC            ;ADVANCE TO JSR INSTR
                        JSR     R5,DELAY        ;STALL FOR STACKING
                        BIT0                    ;COUNT TO 1
                        MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
                        MOV     @STKVAL,$BDDAT  ;READ STACK WORD
                        BIC     #167777,$BDDAT  ;SAVE ONLY CHAR STRING ESCAPE BIT
                        CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
                        BEQ     3$              ;BR IF OK
                        MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
                        ERROR   24              ;CHAR STRING ESCAPE FAILED TO STACK AT LEVEL INDICATED
                3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
                        SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
                        BPL     2$              ;BR IF NOT
                        DEC     BUFFER          ;RESET CHAR STRING ESCAPE AT INSTR LOC
                        SUB     #10000,$GDDAT   ;RESET CHAR STRING ESCAPE AT EXPECTED LOC
                        BPL     1$              ;BR IF RESET STATE NOT TESTED

                ;;*****************************************************************
                ;*TEST 25        TEST THAT DEPTH QUEING BIT GETS PUSHED ON THE STACK
                ;;*****************************************************************
                TST25:  SCOPE
                        MOV     #25,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                        MOV     #2000,$GDDAT    ;EXPECT IT TO BE SET INITIALLY
                        MOV     #176014,BUFFER  ;SET UP DEPTH QUEING AT INSTR LOC
                        MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
                        MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
                1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
                        MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
                2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
                        MOV     #BUFFER,@DPC    ;START
                        MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
                        INC     @DPC            ;ADVANCE TO JSR INSTR
                        JSR     R5,DELAY        ;STALL FOR STACKING
                        BIT0                    ;COUNT TO 1
                        MOV     R0,@STKPT       ;SELECT STACK WORD & BYTE
                        MOV     @STKVAL,$BDDAT  ;READ STACK WORD
```

```
 1108  007722  042737  175777  001126          BIC     #175777,$BDDAT  ;SAVE ONLY DEPTH QUEING BIT
 1109  007730  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1110  007736  001403                          BEQ     3$              ;BR IF SO
 1111  007740  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
 1112  007744  104025                          ERROR   25              ;DEPTH QUEING FAILED TO STACK AT LEVEL INDICATED
 1113  007746  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
 1114  007752  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
 1115  007756  100337                          BPL     2$              ;BR IF NOT
 1116  007760  162737  000004  032374          SUB     #4,BUFFER       ;RESET DEPTH QUEING BIT AT INSTR LOC
 1117  007766  162737  002000  001124          SUB     #2000,$GDDAT    ;RESET DEPTH QUEING BIT AT EXPECTED LOC
 1118  007774  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
 1119
 1120                                  ;;*******************************************************************
  (3)                                  ;*TEST 26        TEST THAT THE DEPTH QUE CONTROL GETS PUSHED ON THE STACK
  (3)                                  ;;*******************************************************************
  (2)  007776  000004           TST26:  SCOPE
  (2)  010000  012737  000026  001206          MOV     #26,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1121  010006  012737  000400  001124          MOV     #400,$GDDAT     ;EXPECT IT TO BE SET INITIALLY
 1122  010014  012737  176300  032374          MOV     #176300,BUFFER  ;SET UP DEPTH QUE CONTROL AT INSTR LOC
 1123  010022  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
 1124  010030  012737  010046  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1125  010036  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 1126  010042  012700  000037                  MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
 1127  010046  010177  172224          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
 1128  010052  012777  032374  172164          MOV     #BUFFER,@DPC    ;START
 1129  010060  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
 1130  010066  005277  172152                  INC     @DPC            ;ADVANCE TO JSR INSTR
 1131  010072  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1132  010076  000001                          BIT0                    ;COUNT TO 1
 1133  010100  010077  172172                  MOV     R0,@STKPT       ;SEL STK WORD & BYTE
 1134  010104  017737  172162  001126          MOV     @STKVAL,$BDDAT  ;READ STACK WORD
 1135  010112  042737  177377  001126          BIC     #177377,$BDDAT  ;SAVE ONLY DEPTH QUE CONTROL
 1136  010120  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1137  010126  001403                          BEQ     3$              ;BR IF SO
 1138  010130  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
 1139  010134  104025                          ERROR   25              ;DEPTH QUE CONTROL FAILED TO STACK
 1140  010136  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
 1141  010142  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
 1142  010146  100337                          BPL     2$              ;BR IF NOT
 1143  010150  162737  000100  032374          SUB     #100,BUFFER     ;RESET DEPTH QUE CONTROL BIT AT INSTR LOC
 1144  010156  162737  000400  001124          SUB     #400,$GDDAT     ;RESET DEPTH QUE CONTROL AT EXPECTED LOC
 1145  010164  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
 1146
 1147                                  ;;*******************************************************************
  (3)                                  ;*TEST 27        TEST THAT BITS SET BY LOAD STATUS B' DON'T CHANGE WHEN CHANGE ENA=0
  (3)                                  ;;*******************************************************************
  (2)  010166  000004           TST27:  SCOPE
  (1)  010170  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
  (2)  010176  012737  000027  001206          MOV     #27,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1148  010204  012737  016400  001124          MOV     #16400,$GDDAT   ;EXPECT ALL LOAD STATUS B' ENABLED BITS FROM STK
 1149  010212  012737  176377  032374          MOV     #176377,BUFFER  ;SET UP LOAD STATUS B' INSTR
 1150  010220  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
 1151  010226  012777  020040  172042  1$:     MOV     #20040,@STKPT   ;SET 'TOP OF STACK'
 1152  010234  012777  032374  172002          MOV     #BUFFER,@DPC    ;START
 1153  010242  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
 1154  010250  000240                          NOP                     ;ALLOW TIME FOR NPR
```

```
1155  010252  005277  171766                    INC     @DPC                    ;PICK UP JSR INSTR
1156  010256  004537  024446                    JSR     R5,DELAY                ;STALL FOR STACKING
1157  010262  000001                            BIT0                            ;COUNT TO 1
1158  010264  012777  020037  172004            MOV     #20037,@STKPT           ;SET STK PTR
1159  010272  017737  171774  001126            MOV     @STKVAL,$BDDAT          ;READ STACK
1160  010300  042737  161377  001126            BIC     #161377,$BDDAT          ;SAVE ONLY DEPTH QUE, EDGE FLG & CHAR STRING
1161  010306  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;THEY SHOULD STAY SET ONCE CHANGE ENABLES ARE OFF
1162  010314  001404                            BEQ     2$                      ;BR IF SO
1163  010316  012737  000037  001164            MOV     #37,$TMP0               ;SAVE STACK LEVEL FOR TYPE
1164  010324  104022                            ERROR   22                      ;LD STATUS B' STROBED WHEN CHANGE ENA=0
1165  010326  105737  032374          2$:       TSTB    BUFFER                  ;IS THIS THE SECOND PASS THIS TEST?
1166  010332  100004                            BPL     3$                      ;GO TO NEXT TEST IF TESTED WITH CHANGE ENA=0
1167  010334  042737  000377  032374            BIC     #377,BUFFER             ;CLR OUT CHANGE ENABLES FOR 2ND PASS THIS TEST
1168  010342  000731                            BR      1$                      ;GO REPEAT TEST WITH ENA=0
1169  010344  000005                  3$:       RESET                           ;CLR ENABLES BEFORE ADVANCING
1170
1171                                            ;;*****************************************************************
 (3)                                            ;*TEST 30       TEST THAT THE LINE TYPE BITS GETS PUSHED ON THE STACK
 (3)                                            ;;*****************************************************************
 (2)  010346  000004                  TST30:    SCOPE
 (2)  010350  012737  000030  001206            MOV     #30,$TESTN              ;;SET TEST NUMBER IN APT MAIL BOX
1172  010356  012737  000003  001124            MOV     #3,$GDDAT               ;EXPECT BOTH BITS INITIALLY
1173  010364  012737  100007  032374            MOV     #100007,BUFFER          ;SET UP BOTH LINE TYP BITS AT INSTR LOC
1174  010372  012737  163000  032376            MOV     #163000,BUFFER+2                ;SET UP JSR INSTR
1175  010400  012737  010416  001110            MOV     #2$,$LPERR              ;SET UP SCOPE LOOP ADRS
1176  010406  012701  020040          1$:       MOV     #20040,R1               ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
1177  010412  012700  000036                    MOV     #36,R0                  ;SET UP STK LEVEL AFTER JSR IN R0
1178  010416  010177  171654          2$:       MOV     R1,@STKPT               ;SET STK PTR BEFORE JSR
1179  010422  012777  032374  171614            MOV     #BUFFER,@DPC            ;START
1180  010430  013737  002272  001122            MOV     STKVAL,$BDADR           ;SET UP REG ADRS 26
1181  010436  012777  032376  171600            MOV     #BUFFER+2,@DPC          ;START
1182  010444  004537  024446                    JSR     R5,DELAY                ;STALL FOR STACKING
1183  010450  000001                            BIT0                            ;COUNT TO 1
1184  010452  010077  171620                    MOV     R0,@STKPT               ;SELECT STACK WORD & BYTE
1185  010456  017737  171610  001126            MOV     @STKVAL,$BDDAT          ;READ STACK
1186  010464  042737  177774  001126            BIC     #177774,$BDDAT          ;SAVE LINE TYPE ONLY
1187  010472  023737  001124  001126            CMP     $GDDAT,$BDDAT           ;ARE THEY CORRECT?
1188  010500  001403                            BEQ     3$                      ;BR IF SO
1189  010502  010037  001164                    MOV     R0,$TMP0                ;SET UP STK LEVEL FOR ER TYPE
1190  010506  104026                            ERROR   26                      ;LINE TYPE FAILED TO STACK AT LEVEL INDICATED
1191  010510  162701  000004          3$:       SUB     #4,R1                   ;ADVANCE TO NEXT STK LEVEL
1192  010514  162700  000004                    SUB     #4,R0                   ;ALL STACK LOCATIONS TESTED?
1193  010520  100336                            BPL     2$                      ;BR IF NOT
1194  010522  005337  032374                    DEC     BUFFER                  ;ADVANCE LINE TYPE AT INSTR LOC
1195  010526  005337  001124                    DEC     $GDDAT                  ;ADVANCE LINE TYPE AT EXPECTED LOC
1196  010532  100325                            BPL     1$                      ;BR IF RESET STATE NOT TESTED
1197
1198                                            ;;*****************************************************************
 (3)                                            ;*TEST 31       TEST THAT INTENSITY ENABLED (CONSOLE 0) GETS PUSHED ON THE STACK
 (3)                                            ;;*****************************************************************
 (2)  010534  000004                  TST31:    SCOPE
 (2)  010536  012737  000031  001206            MOV     #31,$TESTN              ;;SET TEST NUMBER IN APT MAIL BOX
1199  010544  012737  000004  001124            MOV     #4,$GDDAT               ;EXPECT IT TO BE SET INITIALLY
1200  010552  012737  164300  032374            MOV     #164300,BUFFER          ;SET UP INT EN AT INSTR LOC
1201  010560  012737  163000  032376            MOV     #163000,BUFFER+2                ;SET UP JSR REL INSTR
1202  010566  012737  010604  001110            MOV     #2$,$LPERR              ;SET UP SCOPE LOOP ADRS
```

```
 1203  010574  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 1204  010600  012700  000037                  MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
 1205  010604  010177  171466          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
 1206  010610  012777  032374  171426          MOV     #BUFFER,@DPC    ;START
 1207  010616  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
 1208  010624  005277  171414                  INC     @DPC            ;ADVANCE TO JSR INSTR
 1209  010630  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1210  010634  000001                          BIT0                    ;COUNT TO 1
 1211  010636  010077  171434                  MOV     R0,@STKPT       ;SELECT STK WORD & BYTE
 1212  010642  017737  171424  001126          MOV     @STKVAL,$BDDAT  ;READ STK BYTE
 1213  010650  042737  177773  001126          BIC     #177773,$BDDAT  ;SAVE ONLY INTENSITY ENABLED BIT
 1214  010656  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1215  010664  001403                          BEQ     3$              ;BR IF OK
 1216  010666  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
 1217  010672  104027                          ERROR   27              ;INTENSITY ENABLE FAILED TO STACK AT LEVEL INDICATED
 1218  010674  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
 1219  010700  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
 1220  010704  100337                          BPL     2$              ;BR IF NOT
 1221  010706  162737  000100  032374          SUB     #100,BUFFER     ;SET UP RESET STATE AT INSTR LOC
 1222  010714  162737  000004  001124          SUB     #4,$GDDAT       ;SET UP RESET STATE AT EXPECTED LOC
 1223  010722  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
 1224
 1225                                  ;;*******************************************************************
  (3)                                  ;*TEST 32       TEST THAT INTENSITY ENABLED (CONSOLE 1) GETS PUSHED ON THE STACK
  (3)                                  ;;*******************************************************************
  (2)  010724  000004                  TST32:  SCOPE
  (2)  010726  012737  000032  001206          MOV     #32,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1226  010734  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
 1227  010742  012737  000020  001124          MOV     #20,$GDDAT      ;EXPECT IT TO BE SET INITIALLY
 1228  010750  012737  164700  032374          MOV     #164700,BUFFER  ;SET UP INT EN AT INSTR LOC
 1229  010756  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
 1230  010764  012737  011002  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1231  010772  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 1232  010776  012700  000037                  MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
 1233  011002  010177  171270          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
 1234  011006  012777  032374  171230          MOV     #BUFFER,@DPC    ;START
 1235  011014  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
 1236  011022  005277  171216                  INC     @DPC            ;ADVANCE TO JSR INSTR
 1237  011026  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1238  011032  000001                          BIT0                    ;COUNT TO 1
 1239  011034  010077  171236                  MOV     R0,@STKPT       ;SEL STK WORD & BYTE
 1240  011040  017737  171226  001126          MOV     @STKVAL,$BDDAT  ;READ STK BYTE
 1241  011046  042737  177757  001126          BIC     #177757,$BDDAT  ;SAVE ONLY INTENSITY ENABLED BIT
 1242  011054  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1243  011062  001403                          BEQ     3$              ;BR IF OK
 1244  011064  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
 1245  011070  104027                          ERROR   27              ;INTENSITY ENABLE FAILED TO STACK AT LEVEL INDICATED
 1246  011072  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
 1247  011076  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
 1248  011102  100337                          BPL     2$              ;BR IF NOT
 1249  011104  162737  000100  032374          SUB     #100,BUFFER     ;SET UP RESET STATE AT INSTR LOC
 1250  011112  162737  000020  001124          SUB     #20,$GDDAT      ;SET UP RESET STATE AT EXPECTED LOC
 1251  011120  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
 1252
 1253                                  ;;*******************************************************************
  (3)                                  ;*TEST 33       TEST THAT L.P. INTR ENABLED (CONSOLE 0) GETS PUSHED ON THE STACK
```

```
   (3)
   (2)  011122  000004
   (2)  011124  012737  000033  001206  TST33:  SCOPE
                                               MOV     #33,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                                               MOV     #10,$GDDAT      ;EXPECT IT TO BE SET INITIALLY
  1254  011132  012737  000010  001124
  1255  011140  012737  164060  032374         MOV     #164060,BUFFER  ;SET UP L.P. INTR EN AT INSTR LOC
  1256  011146  012737  163000  032376         MOV     #163000,BUFFER+2        ;SET UP JSR INSTR
  1257  011154  012737  011172  001110         MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
  1258  011162  012701  020040          1$:    MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
  1259  011166  012700  000037                 MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
  1260  011172  010177  171100          2$:    MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
  1261  011176  012777  032374  171040         MOV     #BUFFER,@DPC    ;START
  1262  011204  013737  002272  001122         MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
  1263  011212  005277  171026                 INC     @DPC            ;ADVANCE TO JSR INSTR
  1264  011216  004537  024446                 JSR     R5,DELAY        ;STALL FOR STACKING
  1265  011222  000001                         BIT0                    ;COUNT TO 1
  1266  011224  010077  171046                 MOV     R0,@STKPT       ;SEL STK WORD & BYTE
  1267  011230  017737  171036  001126         MOV     @STKVAL,$BDDAT  ;READ STK BYTE
  1268  011236  042737  177767  001126         BIC     #177767,$BDDAT  ;SAVE ONLY L.P. INTR ENABLED BIT
  1269  011244  023737  001124  001126         CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
  1270  011252  001403                         BEQ     3$              ;BR IF SO
  1271  011254  010037  001164                 MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
  1272  011260  104030                         ERROR   30              ;L.P. INTR ENABLE FAILED TO STACK AT LEVEL INDICATED
  1273  011262  162701  000004          3$:    SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
  1274  011266  162700  000004                 SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
  1275  011272  100337                         BPL     2$              ;BR IF NOT
  1276  011274  162737  000020  032374         SUB     #20,BUFFER      ;SET UP RESET STATE AT INSTR LOC
  1277  011302  162737  000010  001124         SUB     #10,$GDDAT      ;SET UP RESET STATE AT EXPECTED LOC
  1278  011310  100324                         BPL     1$              ;BR IF RESET STATE NOT TESTED
  1279
  1280
   (3)                                         ;*TEST 34       TEST THAT L.P. INTR ENABLED (CONSOLE 1) GETS PUSHED ON THE STACK
   (3)
   (2)  011312  000004
   (2)  011314  012737  000034  001206  TST34:  SCOPE
                                               MOV     #34,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
  1281  011322  012737  000040  001124         MOV     #40,$GDDAT      ;EXPECT IT TO BE SET INITIALLY
  1282  011330  012737  164460  032374         MOV     #164460,BUFFER  ;SET UP L.P. INTR EN AT INSTR LOC
  1283  011336  012737  163000  032376         MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
  1284  011344  012737  011362  001110         MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
  1285  011352  012701  020040          1$:    MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
  1286  011356  012700  000037                 MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
  1287  011362  010177  170710          2$:    MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
  1288  011366  012777  032374  170650         MOV     #BUFFER,@DPC    ;START
  1289  011374  013737  002272  001122         MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
  1290  011402  005277  170636                 INC     @DPC            ;ADVANCE TO JSR INSTR
  1291  011406  004537  024446                 JSR     R5,DELAY        ;STALL FOR STACKING
  1292  011412  000001                         BIT0                    ;COUNT TO 1
  1293  011414  010077  170656                 MOV     R0,@STKPT       ;SEL STK WORD & BYTE
  1294  011420  017737  170646  001126         MOV     @STKVAL,$BDDAT  ;READ STK BYTE
  1295  011426  042737  177737  001126         BIC     #177737,$BDDAT  ;SAVE ONLY L.P. INTR ENABLED BIT
  1296  011434  023737  001124  001126         CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
  1297  011442  001403                         BEQ     3$              ;BR IF SO
  1298  011444  010037  001164                 MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
  1299  011450  104030                         ERROR   30              ;L.P. INTR ENABLE FAILED TO STACK AT LEVEL INDICATED
  1300  011452  162701  000004          3$:    SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
  1301  011456  162700  000004                 SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
  1302  011462  100337                         BPL     2$              ;BR IF NOT
```

```
 1303  011464  162737  000020  032374          SUB     #20,BUFFER      ;SET UP RESET STATE AT INSTR LOC
 1304  011472  162737  000040  001124          SUB     #40,$GDDAT      ;SET UP RESET STATE AT EXPECTED LOC
 1305  011500  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
 1306
 1307                                  ;;*******************************************************************
  (3)                                  ;*TEST 35       TEST THAT L.P. SWITCH INTR ENABLED (CONSOLE 0) GETS PUSHED ON THE STACK
  (3)                                  ;;*******************************************************************
  (2)  011502  000004                  TST35:  SCOPE
  (2)  011504  012737  000035  001206          MOV     #35,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1308  011512  012737  000100  001124          MOV     #100,$GDDAT     ;EXPECT IT TO BE SET INITIALLY
 1309  011520  012737  164014  032374          MOV     #164014,BUFFER  ;SET UP L.P. SW INTR EN AT INSTR LOC
 1310  011526  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
 1311  011534  012737  011552  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1312  011542  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 1313  011546  012700  000037                  MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
 1314  011552  010177  170520          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFOE JSR
 1315  011556  012777  032374  170460          MOV     #BUFFER,@DPC    ;START
 1316  011564  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
 1317  011572  005277  170446                  INC     @DPC            ;ADVANCE TO JSR INSTR
 1318  011576  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1319  011602  000001                          BIT0                    ;COUNT TO 1
 1320  011604  010077  170466                  MOV     R0,@STKPT       ;SEL STK WORD & BYTE
 1321  011610  017737  170456  001126          MOV     @STKVAL,$BDDAT  ;READ STK BYTE
 1322  011616  042737  177677  001126          BIC     #177677,$BDDAT  ;SAVE ONLY L.P. SW INTR ENABLED BIT
 1323  011624  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1324  011632  001403                          BEQ     3$              ;BR IF OK
 1325  011634  010037  001164                  MOV     R0,$TMP0        ;SET UP STK LEVEL FOR ER TYPE
 1326  011640  104031                          ERROR   31              ;L.P. SW INTR ENABLE FAILED TO STACK AT LEVEL INDICATED
 1327  011642  162701  000004          3$:     SUB     #4,R1           ;ADVANCE TO NEXT STK LEVEL
 1328  011646  162700  000004                  SUB     #4,R0           ;ALL STACK LOCATIONS TESTED?
 1329  011652  100337                          BPL     2$              ;BR IF NOT
 1330  011654  162737  000004  032374          SUB     #4,BUFFER       ;SET UP RESET STATE AT INSTR. LOC.
 1331  011662  162737  000100  001124          SUB     #100,$GDDAT     ;SET UP RESET STATE AT EXPECTED LOC
 1332  011670  100324                          BPL     1$              ;BR IF RESET STATE NOT TESTED
 1333
 1334                                  ;;*******************************************************************
  (3)                                  ;*TEST 36       TEST THAT L.P. SWITCH INTR ENABLED (CONSOLE 1) GETS PUSHED ON THE STACK
  (3)                                  ;;*******************************************************************
  (2)  011672  000004                  TST36:  SCOPE
  (2)  011674  012737  000036  001206          MOV     #36,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1335  011702  012737  000200  001124          MOV     #200,$GDDAT     ;EXPECT IT TO BE SET INITIALLY
 1336  011710  012737  164414  032374          MOV     #164414,BUFFER  ;SET UP L.P. SW INSTR EN AT INSTR LOC
 1337  011716  012737  163000  032376          MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
 1338  011724  012737  011742  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1339  011732  012701  020040          1$:     MOV     #20040,R1       ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 1340  011736  012700  000037                  MOV     #37,R0          ;SET UP STK LEVEL AFTER JSR IN R0
 1341  011742  010177  170330          2$:     MOV     R1,@STKPT       ;SET STK PTR BEFORE JSR
 1342  011746  012777  032374  170270          MOV     #BUFFER,@DPC    ;START
 1343  011754  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
 1344  011762  005277  170256                  INC     @DPC            ;ADVANCE TO JSR INSTR
 1345  011766  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1346  011772  000001                          BIT0                    ;COUNT TO 1
 1347  011774  010077  170276                  MOV     R0,@STKPT       ;SEL STK WORD & BYTE
 1348  012000  017737  170266  001126          MOV     @STKVAL,$BDDAT  ;READ STK BYTE
 1349  012006  042737  177577  001126          BIC     #177577,$BDDAT  ;SAVE ONLY L.P. SW INTR ENABLED BIT
 1350  012014  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
```

```
 1351  012022  001403                          BEQ     3$                ;BR IF OK
 1352  012024  010037  001164                  MOV     R0,$TMP0          ;SET UP STK LEVEL FOR ER TYPE
 1353  012030  104031                          ERROR   31                ;L.P. SW INTR ENABLE FAILED TO STACK AT LEVEL INDICATED
 1354  012032  162701  000004        3$:     SUB     #4,R1             ;ADVANCE TO NEXT STK LEVEL
 1355  012036  162700  000004                SUB     #4,R0             ;ALL STACK LOCATIONS TESTED?
 1356  012042  100337                          BPL     2$                ;BR IF NOT
 1357  012044  162737  000004  032374        SUB     #4,BUFFER         ;SET UP RESET STATE AT INSTR LOC
 1358  012052  162737  000200  001124        SUB     #200,$GDDAT       ;SET UP RESET STATE AT EXPECTED LOC
 1359  012060  100324                          BPL     1$                ;BR IF RESET STATE NOT TESTED
 1360
 1361                                  ;;*****************************************************************
  (3)                                  ;*TEST 37       TEST THAT THE SHIFT OUT BIT GETS PUSHED ON THE STACK
  (3)                                  ;;*****************************************************************
  (2)  012062  000004                  TST37:  SCOPE
  (2)  012064  012737  000037  001206          MOV     #37,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX
 1362  012072  012737  040000  001124          MOV     #40000,$GDDAT     ;EXPECT IT TO BE SET INITIALLY
 1363  012100  012737  100000  032374          MOV     #100000,BUFFER    ;SET UP CHAR INSTR
 1364  012106  012737  007000  032376          MOV     #7000,BUFFER+2    ;SET CHAR TO SHIFT OUT
 1365  012114  012737  163000  032400          MOV     #163000,BUFFER+4          ;SET UP JSR REL INSTR
 1366  012122  012737  012140  001110          MOV     #2$,$LPERR        ;SET UP SCOPE LOOP ADRS
 1367  012130  012701  020040          1$:     MOV     #20040,R1         ;SET TOP OF STACK & MAINT SW2 INITIALLY IN R1
 1368  012134  012700  000037                  MOV     #37,R0            ;SET UP STK LEVEL AFTER JSR IN R0
 1369  012140  010177  170132          2$:     MOV     R1,@STKPT         ;SET STK PTR BEFORE JSR
 1370  012144  012777  032374  170072          MOV     #BUFFER,@DPC      ;START
 1371  012152  013737  002272  001122          MOV     STKVAL,$BDADR     ;SET UP REG ADRS 26
 1372  012160  005277  170060                  INC     @DPC              ;RESUME
 1373  012164  004537  024446                  JSR     R5,DELAY          ;STALL FOR CHAR DISPLAY
 1374  012170  000010                          BIT3                      ;COUNT TO 10
 1375  012172  005277  170046                  INC     @DPC              ;RESUME
 1376  012176  004537  024446                  JSR     R5,DELAY          ;STALL FOR STKING
 1377  012202  000001                          BIT0                      ;COUNT TO 1
 1378  012204  010077  170066                  MOV     R0,@STKPT         ;SEL STK WD & BYTE
 1379  012210  017737  170056  001126          MOV     @STKVAL,$BDDAT    ;READ STK
 1380  012216  042737  137777  001126          BIC     #137777,$BDDAT    ;SAVE SHIFT OUT BIT ONLY
 1381  012224  023737  001124  001126          CMP     $GDDAT,$BDDAT     ;IS IT CORRECT?
 1382  012232  001403                          BEQ     3$                ;BR IF SO
 1383  012234  010037  001164                  MOV     R0,$TMP0          ;SET UP STK LEVEL FOR ER TYPE
 1384  012240  104032                          ERROR   32                ;SHIFT OUT FAILED TO STACK AT LEVEL INDICATED
 1385  012242  162701  000004        3$:     SUB     #4,R1             ;ADVANCE TO NEXT STK LEVEL
 1386  012246  162700  000004                SUB     #4,R0             ;ALL STACK LOCATIONS TESTED?
 1387  012252  100332                          BPL     2$                ;BR IF MORE STK LOC TO TEST
 1388  012254  012737  000017  032376          MOV     #17,BUFFER+2      ;PROVIDE CHAR THAT WON'T SET SHIFT OUT
 1389  012262  162737  040000  001124          SUB     #40000,$GDDAT     ;EXPECT ZERO STATE NEXT
 1390  012270  100317                          BPL     1$                ;BR IF RESET STATE NOT TESTED
 1391
 1392                                  ;;*****************************************************************
  (3)                                  ;*TEST 40       TEST THAT THE 'POP' INSTR WILL INCREMENT THE STACK POINTER
  (3)                                  ;;*****************************************************************
  (2)  012272  000004                  TST40:  SCOPE
  (2)  012274  012737  000040  001206          MOV     #40,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX
 1393  012302  012777  020000  167766          MOV     #20000,@STKPT     ;SET STACK POINTER TO ZERO
 1394  012310  012737  000004  001124          MOV     #4,$GDDAT         ;EXPECT STK PTR TO BE 1 INITIALLY
 1395  012316  012737  165000  032374          MOV     #165000,BUFFER    ;SET UP POP INSTR
 1396  012324  012777  032374  167712  1$:     MOV     #BUFFER,@DPC      ;START
 1397  012332  013737  002276  001122          MOV     STKPT,$BDADR      ;SET UP REG ADRS 32
 1398  012340  017737  167732  001126          MOV     @STKPT,$BDDAT     ;READ STACK SELECTION
```

```
1399  012346  042737  177740  001126          BIC     #177740,$BDDAT  ;SAVE ONLY STK SEL BITS
1400  012354  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;ARE THEY CORRECT?
1401  012362  001401                          BEQ     2$              ;BR IF SO
1402  012364  104033                          ERROR   33              ;POP INSTR FAILED TO INCREMENT STACK PTR
1403  012366  062737  000004  001124  2$:     ADD     #4,$GDDAT       ;ADVANCE EXPECTED STACK POINTER VALUE
1404  012374  022737  000040  001124          CMP     #40,$GDDAT      ;HAVE DONE ALL STK LOCS?
1405  012402  001350                          BNE     1$              ;BR IF NOT
1406
1407                                  ;;*******************************************************************
 (3)                                  ;*TEST 41        TEST THAT STACK UNDERFLOW WILL SET
 (3)                                  ;;*******************************************************************
 (2)  012404  000004                  TST41:  SCOPE
 (1)  012406  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
 (2)  012414  012737  000041  001206          MOV     #41,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1408  012422  012777  020000  167646          MOV     #20000,@STKPT   ;SET STACK TO ZERO
1409  012430  005037  001124                  CLR     $GDDAT          ;EXPECT NO UNDERFLOW FOR 8 POPS
1410  012434  012700  000011                  MOV     #11,R0          ;SET UP POP COUNT
1411  012440  012737  165000  032374          MOV     #165000,BUFFER  ;SET UP POP INSTR
1412  012446  012777  032374  167570  1$:     MOV     #BUFFER,@DPC    ;START
1413  012454  013737  002256  001122          MOV     SREG1,$BDADR    ;SET UP REG ADRS 12
1414  012462  005300                          DEC     R0              ;COUNT POP OPERATION
1415  012464  001410                          BEQ     2$              ;BR IF UNDERFLOW EXPECTED
1416  012466  032777  010000  167562          BIT     #10000,@SREG1   ;SEE THAT UNDERFLOW  NOT SET
1417  012474  001764                          BEQ     1$              ;BR IF OK
1418  012476  012737  010000  001126          MOV     #10000,$BDDAT   ;INDICATE STACK UNDERFLOW
1419  012504  104034                          ERROR   34              ;STACK UNDERFLOW SET PREMATURELY
1420  012506  032777  010000  167542  2$:     BIT     #10000,@SREG1   ;SEE THAT UNDERFLOW IS SET
1421  012514  001006                          BNE     3$              ;BR IF SO
1422  012516  012737  010000  001124          MOV     #10000,$GDDAT   ;EXPECTED STACK UNDERFLOW
1423  012524  005037  001126                  CLR     $BDDAT          ;INDICATE IT WAS NOT THERE
1424  012530  104034                          ERROR   34              ;STACK UNDERFLOW FAILED TO SET
1425  012532  000005                  3$:     RESET                   ;RESET STACK UNDERFLOW INT REQ BEFORE ADVANCING
1426
1427                                  ;;*******************************************************************
 (3)                                  ;*TEST 42        TEST THAT THE POP INSTR CAN RESTORE THE DPC
 (3)                                  ;;*******************************************************************
 (2)  012534  000004                  TST42:  SCOPE
 (2)  012536  012737  000042  001206          MOV     #42,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1428  012544  013737  002244  001122          MOV     DPC,$BDADR      ;SET UP REG 00 ADRS
1429  012552  012737  032376  001124          MOV     #BUFFER+2,$GDDAT        ;EXPECT BUFFER+2 ADRS AFTER POP
1430  012560  012777  020040  167510          MOV     #20040,@STKPT   ;SET TOP OF STACK
1431  012566  012737  163000  032374          MOV     #163000,BUFFER  ;SET UP JSR REL INSTR
1432  012574  012737  166000  032404          MOV     #166000,BUFFER+10       ;SET UP POP INSTR
1433  012602  012777  032374  167434          MOV     #BUFFER,@DPC    ;START
1434  012610  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
1435  012614  000001                          BIT0                    ;COUNT TO 1
1436  012616  012777  032404  167420          MOV     #BUFFER+10,@DPC ;START
1437  012624  004537  024446                  JSR     R5,DELAY        ;STALL FOR UNSTACKING
1438  012630  000001                          BIT0                    ;COUNT TO 1
1439  012632  017737  167406  001126          MOV     @DPC,$BDDAT     ;READ DPC
1440  012640  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;DID THE DPC GET RESTORED OK
1441  012646  001401                          BEQ     TST43           ;;ADVANCE TO NEXT TEST IF OK
1442  012650  104035                          ERROR   35              ;POP INSTR FAILED TO RESTORE DPC FROM STACK
1443
1444                                  ;;*******************************************************************
 (3)                                  ;*TEST 43        TEST THAT THE NAME BITS WILL GET RESTORED
```

```
  (3)                                   ;;*****************************************************************
  (2)  012652  000004                   TST43:  SCOPE
  (1)  012654  012737  000004  001166            MOV     #4,$TIMES       ;;DO 4 ITERATIONS
  (2)  012662  012737  000043  001206            MOV     #43,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1445  012670  012737  012720  001110            MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1446  012676  013737  002270  001122            MOV     DNAME,$BDADR    ;SET UP REG 24 ADRS
 1447  012704  012737  003777  001124            MOV     #3777,$GDDAT    ;START NAME REG WITH MAXIMUM VALUE
 1448  012712  004437  024462                    JSR     R4,POPSET       ;SET UP JSR, POP & NAME INSTRUCTIONS
 1449  012716  150000                            150000                  ;NAME INSTR
 1450  012720  012737  150000  032374  1$:       MOV     #150000,BUFFER  ;SET UP NAME INSTR
 1451  012726  053737  001124  032374            BIS     $GDDAT,BUFFER   ;SET NAME WORD
 1452  012734  004737  024512                    JSR     PC,DO           ;GO CYCLE THRU INSTRUCTIONS
 1453  012740  017737  167324  001126            MOV     @DNAME,$BDDAT   ;READ NAME REG
 1454  012746  042737  174000  001126            BIC     #174000,$BDDAT  ;SAVE ONLY NAME BITS
 1455  012754  023737  001124  001126            CMP     $GDDAT,$BDDAT   ;DID THEY GET RESTORED?
 1456  012762  001401                            BEQ     2$              ;BR IF SO
 1457  012764  104036                            ERROR   36              ;NAME FAILED TO RESTORE
 1458  012766  005337  001124          2$:       DEC     $GDDAT          ;ADVANCE PATTERN
 1459  012772  100405                            BMI     TST44           ;;ADVANCE TO NEXT TEST IF ALL VALUES TESTED
 1460  012774  001351                            BNE     1$              ;BR IF NAME BITS NOT EQ TO ZERO
 1461  012776  052737  003777  032400            BIS     #3777,BUFFER+4  ;SET NAME REG TO ALL ONES WHEN RESTORING ZERO
 1462  013004  000745                            BR      1$              ;GO RESTORE ALL ZEROS
 1463
 1464                                   ;;*****************************************************************
  (3)                                   ;*TEST 44       TEST THAT THE VECTOR SCALE WILL GET RESTORED
  (3)                                   ;;*****************************************************************
  (2)  013006  000004                   TST44:  SCOPE
  (1)  013010  012737  000100  001166            MOV     #100,$TIMES     ;;DO 100 ITERATIONS
  (2)  013016  012737  000044  001206            MOV     #44,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1465  013024  012737  013054  001110            MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1466  013032  013737  002256  001122            MOV     SREG1,$BDADR    ;SET UP REG 12 ADRS
 1467  013040  012737  000017  001124            MOV     #17,$GDDAT      ;START VECTOR SCALE AT MAXIMUM VALUE
 1468  013046  004437  024462                    JSR     R4,POPSET       ;SET UP JSR,POP & LOAD STATUS C INSTRS
 1469  013052  154020                            154020                  ;LOAD STATUS C INSTR
 1470  013054  012737  154020  032374  1$:       MOV     #154020,BUFFER  ;SET UP LOAD STATUS C INSTR
 1471  013062  053737  001124  032374            BIS     $GDDAT,BUFFER   ;SET VECTOR SCALE
 1472  013070  004737  024512                    JSR     PC,DO           ;GO CYCLE THRU INSTRUCTIONS
 1473  013074  017737  167156  001126            MOV     @SREG1,$BDDAT   ;READ VECTOR SCALE REG
 1474  013102  042737  177760  001126            BIC     #177760,$BDDAT  ;SAVE ONLY THE VECTOR SCALE BITS
 1475  013110  023737  001124  001126            CMP     $GDDAT,$BDDAT   ;DID THEY GET RESTORED?
 1476  013116  001401                            BEQ     2$              ;BR IF SO
 1477  013120  104037                            ERROR   37              ;VECTOR SCALE FAILED TO RESTORE
 1478  013122  005337  001124          2$:       DEC     $GDDAT          ;ADVANCE PATTERN
 1479  013126  100405                            BMI     TST45           ;;ADVANCE TO NEXT TEST IF ALL VALUES TESTED
 1480  013130  001351                            BNE     1$              ;BR IF VECTOR SCALE NOT EQ TO ZERO
 1481  013132  052737  000017  032400            BIS     #17,BUFFER+4    ;SET VECTOR SCALE TO ALL ONES WHEN RESTORING ZEROS
 1482  013140  000745                            BR      1$              ;GO RESTORE ALL ZEROS
 1483
 1484                                   ;;*****************************************************************
  (3)                                   ;*TEST 45       TEST THAT THE CHAR SCALE WILL GET RESTORED
  (3)                                   ;;*****************************************************************
  (2)  013142  000004                   TST45:  SCOPE
  (2)  013144  012737  000045  001206            MOV     #45,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1485  013152  012737  013206  001110            MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1486  013160  013737  002256  001122            MOV     SREG1,$BDADR    ;SET UP REG 12 ADRS
 1487  013166  012737  001400  001124            MOV     #1400,$GDDAT    ;EXPECT CHAR SCALE AT MAXIMUM VALUE
```

```
 1488  013174  012700  000140                    MOV     #140,R0          ;START CHAR SCALE AT MAXIMUM VALUE
 1489  013200  004437  024462                    JSR     R4,POPSET        ;SET UP JSR,POP & LOAD STATUS C INSTR
 1490  013204  154200                            154200                   ;LOAD STATUS C INSTR
 1491  013206  012737  154200  032374  1$:       MOV     #154200,BUFFER   ;SET UP LOAD STATUS C INSTR
 1492  013214  050037  032374                    BIS     R0,BUFFER        ;SET CHAR SCALE
 1493  013220  004737  024512                    JSR     PC,DO            ;GO CYCLE THRU INSTRUCTIONS
 1494  013224  017737  167026  001126            MOV     @SREG1,$BDDAT    ;READ CHAR SCALE REG
 1495  013232  042737  176377  001126            BIC     #176377,$BDDAT   ;SAVE ONLY THE CHAR SCALE BITS
 1496  013240  023737  001124  001126            CMP     $GDDAT,$BDDAT    ;DID THE CHAR SCALE GET RESTORED OK?
 1497  013246  001401                            BEQ     2$               ;BR IF SO
 1498  013250  104040                            ERROR   40               ;CHARACTER SCALE FAILED TO RESTORE
 1499  013252  162737  000400  001124  2$:       SUB     #400,$GDDAT      ;ADVANCE CHAR SCALE VALUE
 1500  013260  162700  000040                    SUB     #40,R0           ;SET UP R0 WITH NEXT VALUE
 1501  013264  100405                            BMI     TST46            ;;ADVANCE TO NEXT TEST IF ALL VALUES TESTED
 1502  013266  001347                            BNE     1$               ;GO TRY NEXT VALUE IF CHAR SCALE NOT EQ ZERO
 1503  013270  052737  000140  032400            BIS     #140,BUFFER+4    ;SET CHAR SCALE TO ALL ONES WHEN RETORING ZEROS
 1504  013276  000743                            BR      1$               ;GO RESTORE ALL ZEROS
 1505
 1506                                    ;;*******************************************************************
   (3)                                   ;*TEST 46       TEST THAT THE CHAR ROTATE BIT GETS RESTORED
   (3)                                   ;;*******************************************************************
   (2)  013300  000004                   TST46:  SCOPE
   (2)  013302  012737  000046  001206            MOV     #46,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
 1507  013310  012737  013346  001110            MOV     #1$,$LPERR       ;SET UP SCOPE LOOP ADRS
 1508  013316  013737  002256  001122            MOV     SREG1,$BDADR     ;SET UP REG 12 ADRS
 1509  013324  012737  002000  001124            MOV     #2000,$GDDAT     ;EXPECT CHAR ROTATE TO BE SET INITIALLY
 1510  013332  004437  024462                    JSR     R4,POPSET        ;SET UP JSR,POP & LOAD STATUS C INSTR
 1511  013336  155000                            155000                   ;LOAD STATUS C INSTR
 1512  013340  012737  155400  032374            MOV     #155400,BUFFER   ;SET UP LOAD STATUS C INSTR
 1513  013346  004737  024512          1$:       JSR     PC,DO            ;GO CYCLE THRU INSTRUCTIONS
 1514  013352  017737  166700  001126            MOV     @SREG1,$BDDAT    ;READ CHAR ROTATE REG
 1515  013360  042737  175777  001126            BIC     #175777,$BDDAT   ;SAVE ONLY CHAR ROTATE BIT
 1516  013366  023737  001124  001126            CMP     $GDDAT,$BDDAT    ;IS IT CORRECT?
 1517  013374  001401                            BEQ     2$               ;BR IF SO
 1518  013376  104041                            ERROR   41               ;CHARACTER ROTATE FAILED TO RESTORE
 1519  013400  162737  002000  001124  2$:       SUB     #2000,$GDDAT     ;CLR CHAR ROTATE
 1520  013406  100407                            BMI     TST47            ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1521  013410  042737  000400  032374            BIC     #400,BUFFER      ;SET UP FOR RESET STATE
 1522  013416  052737  000400  032400            BIS     #400,BUFFER+4    ;SET CHAR ROTATE BIT
 1523  013424  000750                            BR      1$               ;GO TRY RESET STATE
 1524
 1525                                    ;;*******************************************************************
   (3)                                   ;*TEST 47       TEST THAT THE COLOR LEVEL WILL GET RESTORED
   (3)                                   ;;*******************************************************************
   (2)  013426  000004                   TST47:  SCOPE
   (2)  013430  012737  000047  001206            MOV     #47,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
 1526  013436  012737  013472  001110            MOV     #1$,$LPERR       ;SET UP SCOPE LOOP ADRS
 1527  013444  013737  002266  001122            MOV     CONS,$BDADR      ;SET UP REG 22 ADRS
 1528  013452  012737  000014  001124            MOV     #14,$GDDAT       ;EXPECT MAX VALUE INITIALLY
 1529  013460  012700  000600                    MOV     #600,R0          ;SET UP COLOR LEVEL FOR INSTR USE
 1530  013464  004437  024462                    JSR     R4,POPSET        ;SET UP JSR,POP & LOAD STATUS B INSTR
 1531  013470  175000                            175000                   ;LOAD STATUS B INSTR
 1532  013472  012737  175000  032374  1$:       MOV     #175000,BUFFER   ;SET UP LOAD STATUS B INSTR
 1533  013500  050037  032374                    BIS     R0,BUFFER        ;SET UP COLOR LEVEL
 1534  013504  004737  024512                    JSR     PC,DO            ;GO CYCLE THRU INSTRUCTIONS
 1535  013510  017737  166552  001126            MOV     @CONS,$BDDAT     ;READ COLOR LEVEL REG
```

```
 1536  013516  042737  177763  001126          BIC     #177763,$BDDAT  ;SAVE COLOR LEVEL ONLY
 1537  013524  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1538  013532  001401                          BEQ     2$              ;BR IF SO
 1539  013534  104042                          ERROR   42              ;COLOR LEVEL FAILED TO RESTORE
 1540  013536  162737  000004  001124  2$:     SUB     #4,$GDDAT       ;ADVANCE TO NEXT VALUE
 1541  013544  162700  000200                  SUB     #200,R0         ;SET UP R4 WITH NEXT VALUE
 1542  013550  100405                          BMI     TST50           ;;ADVANCE TO NEXT TEST IF ALL COLOR LEVELS TESTED
 1543  013552  001347                          BNE     1$              ;GO TEST NEXT COLOR LEVEL IF NOT ZERO
 1544  013554  052737  000600  032400          BIS     #600,BUFFER+4   ;SET COLOR LEVEL TO ALL ONES WHEN RESTORING ZEROS
 1545  013562  000743                          BR      1$              ;GO RESTORE ALL ZEROS
 1546
 1547                                  ;;*******************************************************************
  (3)                                  ;*TEST 50       TEST THAT THE ITALICS BIT GETS RETORED
  (3)                                  ;;*******************************************************************
  (2)  013564  000004                  TST50:  SCOPE
  (2)  013566  012737  000050  001206          MOV     #50,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1548  013574  012737  013624  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1549  013602  013737  002246  001122          MOV     SREG0,$BDADR    ;SET UP REG 02 ADRS
 1550  013610  012737  000020  001124          MOV     #20,$GDDAT      ;EXPECT ITALICS INITIALLY
 1551  013616  004437  024462                  JSR     R4,POPSET       ;SET UP JSR,POP & LOAD STATUS A
 1552  013622  170040                          170040                  ;LOAD STATUS A INSTR
 1553  013624  012737  170040  032374  1$:     MOV     #170040,BUFFER  ;SET UP LOAD STATUS A INSTR
 1554  013632  053737  001124  032374          BIS     $GDDAT,BUFFER   ;SET UP ITALICS
 1555  013640  004737  024512                  JSR     PC,DO           ;GO CYCLE THRU INSTRUCTIONS
 1556  013644  017737  166376  001126          MOV     @SREG0,$BDDAT   ;READ ITALICS REG
 1557  013652  042737  177757  001126          BIC     #177757,$BDDAT  ;SAVE ONLY ITALICS
 1558  013660  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1559  013666  001401                          BEQ     2$              ;BR IF SO
 1560  013670  104043                          ERROR   43              ;ITALICS FAILED TO RESTORE
 1561  013672  162737  000020  001124  2$:     SUB     #20,$GDDAT      ;GO TO RESET STATE
 1562  013700  100404                          BMI     TST51           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1563  013702  052737  000020  032400          BIS     #20,BUFFER+4    ;SET ITALICS BIT TO ONE WHEN RESTORING ZERO
 1564  013710  000745                          BR      1$              ;GO RESTORE RESET STATE
 1565
 1566                                  ;;*******************************************************************
  (3)                                  ;*TEST 51       TEST THAT THE MENU BIT GETS RESTORED
  (3)                                  ;;*******************************************************************
  (2)  013712  000004                  TST51:  SCOPE
  (2)  013714  012737  000051  001206          MOV     #51,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1567  013722  012737  013760  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1568  013730  013737  002256  001122          MOV     SREG1,$BDADR    ;SET UP REG 12 ADRS
 1569  013736  012737  000100  001124          MOV     #100,$GDDAT     ;EXPECT MENU INITIALLY
 1570  013744  004437  024462                  JSR     R4,POPSET       ;SET UP JSR,POP & LOAD STATUS A INSTR
 1571  013750  170002                          170002                  ;LOAD STATUS A INSTR
 1572  013752  012737  170003  032374          MOV     #170003,BUFFER  ;SET UP LOAD STATUS A INSTR
 1573  013760  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRUCTIONS
 1574  013764  017737  166266  001126          MOV     @SREG1,$BDDAT   ;READ MENU REG
 1575  013772  042737  177677  001126          BIC     #177677,$BDDAT  ;SAVE ONLY THE MENU BIT
 1576  014000  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1577  014006  001401                          BEQ     2$              ;BR IF SO
 1578  014010  104044                          ERROR   44              ;MENU FAILED TO RESTORE
 1579  014012  162737  000100  001124  2$:     SUB     #100,$GDDAT     ;RESET MENU BIT
 1580  014020  100407                          BMI     TST52           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1581  014022  042737  000001  032374          BIC     #1,BUFFER       ;RESET MENU FOR INSTR USE
 1582  014030  052737  000001  032400          BIS     #1,BUFFER+4     ;SET MENU WHEN RESTORING ZERO
 1583  014036  000750                          BR      1$              ;GO RESTORE ZERO
```

```
                                        ;;*****************************************************************
                                        ;*TEST 52        TEST THAT THE INTENSITY ENA GETS RESTORED ON CONSOLE 0
                                        ;;*****************************************************************
014040  000004                  TST52:  SCOPE
014042  012737  000052  001206          MOV     #52,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
014050  012737  014106  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
014056  013737  002266  001122          MOV     CONS,$BDADR     ;SET UP REG 22 ADRS
014064  012737  100000  001124          MOV     #100000,$GDDAT  ;EXPECT INTENSITY ENABLED TO BE SET INITIALLY
014072  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & NOP INSTRS
014076  164200                          164200                  ;NOP INSTR
014100  012737  164300  032374          MOV     #164300,BUFFER  ;SET UP NOP INSTR
014106  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
014112  017737  166150  001126          MOV     @CONS,$BDDAT    ;READ REG 22
014120  042737  077777  001126          BIC     #77777,$BDDAT   ;SAVE ONLY INTENSITY ENA BIT
014126  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
014134  001401                          BEQ     2$              ;BR IF SO
014136  104045                          ERROR   45              ;INTENSITY ENA FAILED TO RESTORE
014140  162737  100000  001124  2$:     SUB     #100000,$GDDAT  ;CLR INTENSITY ENA BIT AT EXPECTED LOC
014146  100407                          BMI     TST53           ;;ADVANCE TO NEXT STATE IF BOTH STATES TESTED
014150  042737  000100  032374          BIC     #100,BUFFER     ;SET UP FOR RESET STATE
014156  052737  000100  032400          BIS     #100,BUFFER+4   ;SET INTENSITY ENA AT INSTR LOC
014164  000750                          BR      1$              ;GO TRY RESET STATE

                                        ;;*****************************************************************
                                        ;*TEST 53        TEST THAT THE INTENSITY ENA GETS RESTORED ON CONSOLE 1
                                        ;;*****************************************************************
014166  000004                  TST53:  SCOPE
014170  012737  000053  001206          MOV     #53,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
014176  012737  014234  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
014204  013737  002266  001122          MOV     CONS,$BDADR     ;SET UP REG 22 ADRS
014212  012737  001000  001124          MOV     #1000,$GDDAT    ;EXPECT INTENSITY ENABLED TO BE SET INITIALLY
014220  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & NOP INSTRS
014224  164600                          164600                  ;NOP INSTR
014226  012737  164700  032374          MOV     #164700,BUFFER  ;SET UP NOP INSTR
014234  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
014240  017737  166022  001126          MOV     @CONS,$BDDAT    ;READ REG 22
014246  042737  176777  001126          BIC     #176777,$BDDAT  ;SAVE ONLY INTENSITY ENA BIT
014254  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
014262  001401                          BEQ     2$              ;BR IS SO
014264  104045                          ERROR   45              ;INTENSITY ENA FAILED TO RESTORE
014266  162737  001000  001124  2$:     SUB     #1000,$GDDAT    ;CLR INTENSITY ENA BIT AT EXPECTED LOC
014274  100407                          BMI     TST54           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
014276  042737  000100  032374          BIC     #100,BUFFER     ;SET UP FOR RESET STATE
014304  052737  000100  032400          BIS     #100,BUFFER+4   ;SET INTENSITY ENA AT INSTR LOC
014312  000750                          BR      1$              ;GO TRY RESET STATE

                                        ;;*****************************************************************
                                        ;*TEST 54        TEST THAT L.P. INTR ENA GETS RESTORED ON CONSOLE 0
                                        ;;*****************************************************************
014314  000004                  TST54:  SCOPE
014316  012737  000054  001206          MOV     #54,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
014324  012737  014362  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
014332  013737  002266  001122          MOV     CONS,$BDADR     ;SET UP REG 22 ADRS
014340  012737  004000  001124          MOV     #4000,$GDDAT    ;EXPECT L.P. INTR ENA TO BE SET INITIALLY
014346  012737  164060  032374          MOV     #164060,BUFFER  ;SET UP NOP INSTR
```

```
 1628  014354  004437  024462                      JSR     R4,POPSET       ;SET UP JSR, POP & NOP INSTRS
 1629  014360  164040                              164040                  ;NOP INSTR
 1630  014362  004737  024512              1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1631  014366  017737  165674  001126              MOV     @CONS,$BDDAT    ;READ REG 22
 1632  014374  042737  173777  001126              BIC     #173777,$BDDAT  ;SAVE ONLY L.P. INTR ENA
 1633  014402  023737  001124  001126              CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1634  014410  001401                              BEQ     2$              ;BR IF SO
 1635  014412  104046                              ERROR   46              ;L.P. INTR ENA FAILED TO RESTORE
 1636  014414  162737  004000  001124  2$:         SUB     #4000,$GDDAT    ;SET UP FOR RESET STATE
 1637  014422  100407                              BMI     TST55           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1638  014424  042737  000020  032374              BIC     #20,BUFFER      ;SET UP FOR RESET STATE
 1639  014432  052737  000020  032400              BIS     #20,BUFFER+4    ;SET IT BEFORE POP
 1640  014440  000750                              BR      1$              ;GO TRY RESET STATE
 1641
 1642                                      ;;*****************************************************************
  (3)                                      ;*TEST 55       TEST THAT L.P. INTR ENA GETS RESTORED ON CONSOLE 1
  (3)                                      ;;*****************************************************************
  (2)  014442  000004                      TST55:  SCOPE
  (2)  014444  012737  000055  001206              MOV     #55,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1643  014452  012737  014510  001110              MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1644  014460  013737  002266  001122              MOV     CONS,$BDADR     ;SET UP REG 22 ADRS
 1645  014466  012737  000040  001124              MOV     #40,$GDDAT      ;EXPECT L.P. INTR ENABLED TO BE SET INITIALLY
 1646  014474  012737  164460  032374              MOV     #164460,BUFFER  ;SET UP NOP INSTR
 1647  014502  004437  024462                      JSR     R4,POPSET       ;SET UP JSR, POP & NOP INSTRS
 1648  014506  164440                              164440                  ;NOP INSTR
 1649  014510  004737  024512              1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1650  014514  017737  165546  001126              MOV     @CONS,$BDDAT    ;READ REG 22
 1651  014522  042737  177737  001126              BIC     #177737,$BDDAT  ;SAVE ONLY L.P. INTR ENA
 1652  014530  023737  001124  001126              CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1653  014536  001401                              BEQ     2$              ;BR IF SO
 1654  014540  104046                              ERROR   46              ;L.P. INTR ENA FAILED TO RESTORE
 1655  014542  162737  000040  001124  2$:         SUB     #40,$GDDAT      ;SET UP FOR RESET STATE
 1656  014550  100407                              BMI     TST56           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1657  014552  042737  000020  032374              BIC     #20,BUFFER      ;SET UP FOR RESET STATE
 1658  014560  052737  000020  032400              BIS     #20,BUFFER+4    ;SET IF BEFORE POP
 1659  014566  000750                              BR      1$              ;GO TRY RESET STATE
 1660
 1661                                      ;;*****************************************************************
  (3)                                      ;*TEST 56       TEST THAT L.P. SW INTR ENA GETS RESTORED ON CONSOLE 0
  (3)                                      ;;*****************************************************************
  (2)  014570  000004                      TST56:  SCOPE
  (2)  014572  012737  000056  001206              MOV     #56,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1662  014600  012737  014636  001110              MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1663  014606  013737  002266  001122              MOV     CONS,$BDADR     ;SET UP REG 22 ADRS
 1664  014614  012737  002000  001124              MOV     #2000,$GDDAT    ;EXPECT L.P. SW INTR ENA TO BE SET INITIALLY
 1665  014622  012737  164014  032374              MOV     #164014,BUFFER  ;SET UP NOP INSTR
 1666  014630  004437  024462                      JSR     R4,POPSET       ;SET UP JSR, POP & NOP INSTRS
 1667  014634  164010                              164010                  ;NOP INSTR
 1668  014636  004737  024512              1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRUCTIONS
 1669  014642  017737  165420  001126              MOV     @CONS,$BDDAT    ;READ REG 22
 1670  014650  042737  175777  001126              BIC     #175777,$BDDAT  ;SAVE ONLY L.P. SW INTR ENA
 1671  014656  023737  001124  001126              CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1672  014664  001401                              BEQ     2$              ;BR IF NOT
 1673  014666  104047                              ERROR   47              ;L.P. SW INTR ENA FAILED TO RESTORE
 1674  014670  162737  002000  001124  2$:         SUB     #2000,$GDDAT    ;SET UP FOR RESET STATE
 1675  014676  100407                              BMI     TST57           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
```

```
 1676  014700  042737  000004  032374          BIC     #4,BUFFER       ;SET UP FOR RESET STATE
 1677  014706  052737  000004  032400          BIS     #4,BUFFER+4     ;SET IT BEFORE POP
 1678  014714  000750                          BR      1$              ;GO TRY RESET STATE
 1679
 1680                                  ;;*******************************************************************
  (3)                                  ;*TEST 57       TEST THAT L.P. SW INTR ENA GETS RESTORED ON CONSOLE 1
  (3)                                  ;;*******************************************************************
  (2)  014716  000004                  TST57:  SCOPE
  (2)  014720  012737  000057  001206          MOV     #57,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1681  014726  012737  014764  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1682  014734  013737  002266  001122          MOV     CONS,$BDADR     ;SET UP REG 22 ADRS
 1683  014742  012737  000020  001124          MOV     #20,$GDDAT      ;EXPECT L.P. SW INTR ENA TO BE SET INITIALLY
 1684  014750  012737  164414  032374          MOV     #164414,BUFFER  ;SET UP NOP INSTR
 1685  014756  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & NOP INSTRS
 1686  014762  164410                          164410                  ;NOP INSTR
 1687  014764  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1688  014770  017737  165272  001126          MOV     @CONS,$BDDAT    ;READ REG 22
 1689  014776  042737  177757  001126          BIC     #177757,$BDDAT  ;SAVE ONLY L.P. SW INTR ENA
 1690  015004  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1691  015012  001401                          BEQ     2$              ;BR IF NOT
 1692  015014  104047                          ERROR   47              ;L.P. SW INTR ENA FAILED TO RESTORE
 1693  015016  162737  000020  001124  2$:     SUB     #20,$GDDAT      ;SET UP FOR RESET STATE
 1694  015024  100407                          BMI     TST60           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1695  015026  042737  000004  032374          BIC     #4,BUFFER       ;SET UP RESET STATE
 1696  015034  052737  000004  032400          BIS     #4,BUFFER+4     ;SET IT BEFORE POPS
 1697  015042  000750                          BR      1$              ;GO TRY RESET STATE
 1698
 1699                                  ;;*******************************************************************
  (3)                                  ;*TEST 60       TEST THAT THE LINE TYPE WILL GET RESTORED
  (3)                                  ;;*******************************************************************
  (2)  015044  000004                  TST60:  SCOPE
  (2)  015046  012737  000060  001206          MOV     #60,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1700  015054  012737  015104  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1701  015062  013737  002246  001122          MOV     SREG0,$BDADR    ;SET UP REG 02 ADRS
 1702  015070  012737  000003  001124          MOV     #3,$GDDAT       ;EXPECT MAX VALUE INITIALLY
 1703  015076  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & CHAR INSTRS
 1704  015102  100004                          100004                  ;CHAR INSTR
 1705  015104  012737  100004  032374  1$:     MOV     #100004,BUFFER  ;SET UP CHAR INSTR
 1706  015112  053737  001124  032374          BIS     $GDDAT,BUFFER   ;SET UP LINE TYPE
 1707  015120  004737  024512                  JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1708  015124  017737  165116  001126          MOV     @SREG0,$BDDAT   ;READ LINE TYPE
 1709  015132  042737  077774  001126          BIC     #77774,$BDDAT   ;SAVE ONLY LINE TYPE
 1710  015140  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1711  015146  001401                          BEQ     2$              ;BR IF SO
 1712  015150  104050                          ERROR   50              ;LINE TYPE FAILED TO RESTORE
 1713  015152  005337  001124          2$:     DEC     $GDDAT          ;ADVANCE TO NEXT VALUE
 1714  015156  100405                          BMI     TST61           ;;ADVANCE TO NEXT TEST IF ALL VALUES TESTED
 1715  015160  001351                          BNE     1$              ;TRY NEXT LINE TYPE IF NOT ZERO
 1716  015162  052737  000003  032400          BIS     #3,BUFFER+4     ;SET LINE TYPE TO 3 WHEN RESTORING ZEROS
 1717  015170  000745                          BR      1$              ;GO RESTORE ALL ZEROS
 1718
 1719                                  ;;*******************************************************************
  (3)                                  ;*TEST 61       TEST THAT THE INTENSITY LEVEL WILL GET RESTORED
  (3)                                  ;;*******************************************************************
  (2)  015172  000004                  TST61:  SCOPE
  (2)  015174  012737  000061  001206          MOV     #61,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
```

```
 1720  015202  012737  015236  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1721  015210  013737  002246  001122          MOV     SREG0,$BDADR    ;SET UP REG 02 ADRS
 1722  015216  012737  003400  001124          MOV     #3400,$GDDAT    ;EXPECT MAX VALUE INITIALLY
 1723  015224  012700  001600                  MOV     #1600,R0        ;SET UP INTENSITY LEVEL FOR INSTR USE
 1724  015230  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & CHAR INSTRS
 1725  015234  102000                          102000                  ;CHAR INSTR
 1726  015236  012737  102000  032374  1$:     MOV     #102000,BUFFER  ;SET UP CHAR INSTR
 1727  015244  050037  032374                  BIS     R0,BUFFER       ;SET UP INTENSITY LEVEL
 1728  015250  004737  024512                  JSR     PC,DO           ;GO CYCLE THRU INSTR
 1729  015254  017737  164766  001126          MOV     @SREG0,$BDDAT   ;READ INTENSITY LEVEL
 1730  015262  042737  174377  001126          BIC     #174377,$BDDAT  ;SAVE ONLY INTENSITY LEVEL
 1731  015270  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1732  015276  001401                          BEQ     2$              ;BR IF SO
 1733  015300  104051                          ERROR   51              ;INTENSITY LEVEL FAILED TO RESTORE
 1734  015302  162700  000200          2$:     SUB     #200,R0         ;SET UP R0 WITH NEXT VALUE
 1735  015306  162737  000400  001124          SUB     #400,$GDDAT     ;ADVANCE TO NEXT VALUE
 1736  015314  100405                          BMI     TST62           ;;ADVANCE TO NEXT TEST IF ALL VALUES TESTED
 1737  015316  001347                          BNE     1$              ;BR IF NEXT INTENSITY LEVEL NOT ZERO
 1738  015320  052737  001600  032400          BIS     #1600,BUFFER+4  ;SET INTENSITY LEVEL TO ALL ONES WHEN RESTORING ZEROS
 1739  015326  000743                          BR      1$              ;GO RESTORE ALL ZEROS
 1740
 1741                                  ;;*****************************************************************
  (3)                                  ;*TEST 62       TEST THAT BLINK WILL GET RESTORED
  (3)                                  ;;*****************************************************************
  (2)  015330  000004                  TST62:  SCOPE
  (2)  015332  012737  000062  001206          MOV     #62,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1742  015340  012737  015376  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1743  015346  013737  002246  001122          MOV     SREG0,$BDADR    ;SET UP REG 02 ADRS
 1744  015354  012737  000010  001124          MOV     #10,$GDDAT      ;EXPECT BLINK INITIALLY
 1745  015362  012737  100030  032374          MOV     #100030,BUFFER  ;SET UP CHAR INSTR
 1746  015370  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & CHAR INSTRS
 1747  015374  100020                          100020                  ;CHAR INSTR
 1748  015376  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1749  015402  017737  164640  001126          MOV     @SREG0,$BDDAT   ;READ REG 02
 1750  015410  042737  177767  001126          BIC     #177767,$BDDAT  ;SAVE ONLY THE BLINK BIT
 1751  015416  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1752  015424  001401                          BEQ     2$              ;BR IF SO
 1753  015426  104052                          ERROR   52              ;BLINK FAILED TO RESTORE
 1754  015430  162737  000010  032374  2$:     SUB     #10,BUFFER      ;SET UP RESET STATE
 1755  015436  162737  000010  001124          SUB     #10,$GDDAT      ;GO TO RESET STATE
 1756  015444  100404                          BMI     TST63           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1757  015446  052737  000010  032400          BIS     #10,BUFFER+4    ;SET BLINK WHEN RESTORING ZERO
 1758  015454  000750                          BR      1$              ;GO TRY RESET STATE
 1759
 1760                                  ;;*****************************************************************
  (3)                                  ;*TEST 63       TEST THAT THE MODE WILL GET RESTORED
  (3)                                  ;;*****************************************************************
  (2)  015456  000004                  TST63:  SCOPE
  (1)  015460  012737  000100  001166          MOV     #100,$TIMES     ;;DO 100 ITERATIONS
  (2)  015466  012737  000063  001206          MOV     #63,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1761  015474  012737  015524  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1762  015502  013737  002246  001122          MOV     SREG0,$BDADR    ;SET UP REG 02 ADRS
 1763  015510  012737  044000  001124          MOV     #44000,$GDDAT   ;EXPECT ABS VEC MODE INITIALLY
 1764  015516  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & MODE INSTR
 1765  015522  100000                          100000                  ;CHAR INSTR
 1766  015524  052737  100000  032400  1$:     BIS     #100000,BUFFER+4        ;RESTORE 'INSTR' BIT AT LOC BUFFER+4
```

```
 1767  015532  012737  100000  032374          MOV     #100000,BUFFER  ;SET UP CHAR INSTR
 1768  015540  053737  001124  032374          BIS     $GDDAT,BUFFER   ;SET UP CURRENT DISPLAY INSTR
 1769  015546  004737  024512                  JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1770  015552  042737  100000  032400          BIC     #100000,BUFFER+4        ;USE LOC BUFFER+4 NOW AS DISPLAY DATA
 1771  015560  005277  164460                  INC     @DPC            ;PICK UP DATA FOR DISPLAY INSTR ISSUED
 1772  015564  000240                          NOP                     ; BEFORE JSR - THIS WILL LD MODE BITS
 1773  015566  017737  164454  001126          MOV     @SREG0,$BDDAT   ;READ REG 02
 1774  015574  042737  103777  001126          BIC     #103777,$BDDAT  ;SAVE ONLY MODE BITS
 1775  015602  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1776  015610  001401                          BEQ     2$              ;BR IF OK
 1777  015612  104053                          ERROR   53              ;MODE FAILED TO RESTORE
 1778  015614  162737  004000  001124  2$:     SUB     #4000,$GDDAT    ;GO TO NEXT DISPLAY MODE
 1779  015622  100340                          BPL     1$              ;BR IF MORE MODES TO TEST
 1780
 1781                                  ;;*******************************************************************
  (3)                                  ;*TEST 64        TEST THAT STOP INTR ENABLED GETS RESTORED-CK IT ON STK
  (3)                                  ;;*******************************************************************
  (2)  015624  000004                  TST64:  SCOPE
  (2)  015626  012737  000064  001206          MOV     #64,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1782  015634  012737  015672  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1783  015642  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
 1784  015650  012737  000001  001124          MOV     #1,$GDDAT       ;EXPECT IT TO BE SET INITIALLY
 1785  015656  012737  171400  032374          MOV     #171400,BUFFER  ;SET UP LOAD STATUS A INSTR
 1786  015664  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & LOAD STATUS A INSTR
 1787  015670  171000                          171000                  ;LOAD STATUS A INSTR
 1788  015672  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1789  015676  012777  032376  164340          MOV     #BUFFER+2,@DPC  ;DO A JSR
 1790  015704  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1791  015710  000001                          BIT0                    ;COUNT TO 1
 1792  015712  012777  020037  164356          MOV     #20037,@STKPT   ;SET UP STK SEL
 1793  015720  017737  164346  001126          MOV     @STKVAL,$BDDAT  ;READ STOP INTR ENA BYTE FROM STACK
 1794  015726  042737  177776  001126          BIC     #177776,$BDDAT  ;SAVE ONLY STOP INTR ENABLED BIT
 1795  015734  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;WAS IT RESTORED ON THE POP INSTR?
 1796  015742  001401                          BEQ     2$              ;BR IF SO
 1797  015744  104054                          ERROR   54              ;STOP INTR ENABLE FAILED TO RESTORE
 1798  015746  005337  001124          2$:     DEC     $GDDAT          ;SET UP FOR RESET STATE
 1799  015752  100407                          BMI     TST65           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1800  015754  042737  000400  032374          BIC     #400,BUFFER     ;SET UP FOR STKING RESET STATE
 1801  015762  052737  000400  032400          BIS     #400,BUFFER+4   ;SET BIT BEFORE POP INSTR
 1802  015770  000740                          BR      1$              ;GO TRY RESET STATE
 1803
 1804                                  ;;*******************************************************************
  (3)                                  ;*TEST 65        TEST THAT DEPTH QUEING BIT GETS RESTORED-CK IT ON STK
  (3)                                  ;;*******************************************************************
  (2)  015772  000004                  TST65:  SCOPE
  (2)  015774  012737  000065  001206          MOV     #65,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1805  016002  012737  016040  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1806  016010  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
 1807  016016  012737  002000  001124          MOV     #2000,$GDDAT    ;EXPECT IT TO BE SET INITIALLY
 1808  016024  012737  176014  032374          MOV     #176014,BUFFER  ;SET UP LOAD STATUS B' INSTR
 1809  016032  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & LOAD STATUS B' INSTRS
 1810  016036  176010                          176010                  ;LOAD STATUS B' INSTR
 1811  016040  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1812  016044  012777  032376  164172          MOV     #BUFFER+2,@DPC  ;DO A JSR
 1813  016052  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1814  016056  000001                          BIT0                    ;COUNT TO 1
```

```
 1815  016060  012777  020037  164210          MOV     #20037,@STKPT   ;SET UP STK SEL
 1816  016066  017737  164200  001126          MOV     @STKVAL,$BDDAT  ;READ DEPTH QUEING BYTE FROM STK
 1817  016074  042737  175777  001126          BIC     #175777,$BDDAT  ;SAVE ONLY DEPTH QUEING BIT
 1818  016102  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;WAS IT RESTORED ON THE POP INSTR?
 1819  016110  001401                          BEQ     2$              ;BR IF SO
 1820  016112  104055                          ERROR   55              ;DEPTH QUE FAILED TO RESTORE
 1821  016114  162737  002000  001124  2$:     SUB     #2000,$GDDAT    ;SET UP FOR RESET STATE
 1822  016122  100407                          BMI     TST66           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1823  016124  042737  000004  032374          BIC     #4,BUFFER       ;SET UP FOR STKING RESET STATE
 1824  016132  052737  000004  032400          BIS     #4,BUFFER+4     ;SET BIT BEFORE POP INSTR
 1825  016140  000737                          BR      1$              ;GO TRY RESET STATE
 1826
 1827                                  ;;*******************************************************************
  (3)                                  ;*TEST 66       TEST THAT DEPTH QUE CONTROL BIT GETS RESTROED-CK IT ON STK
  (3)                                  ;;*******************************************************************
  (2)  016142  000004                  TST66:  SCOPE
  (2)  016144  012737  000066  001206          MOV     #66,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1828  016152  012737  016210  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1829  016160  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
 1830  016166  012737  000400  001124          MOV     #400,$GDDAT     ;EXPECT IT TO BE SET INITIALLY
 1831  016174  012737  176300  032374          MOV     #176300,BUFFER  ;SET UP LOAD STATUS B' INSTR
 1832  016202  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & LOAD STATUS B' INSTRS
 1833  016206  176200                          176200                  ;LOAD STATUS B' INSTR
 1834  016210  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1835  016214  012777  032374  164022          MOV     #BUFFER,@DPC    ;DO A JSR REL
 1836  016222  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1837  016226  000001                          BIT0                    ;COUNT TO 1
 1838  016230  012777  020037  164040          MOV     #20037,@STKPT   ;SET UP STACK SEL
 1839  016236  017737  164030  001126          MOV     @STKVAL,$BDDAT  ;READ STACK
 1840  016244  042737  177377  001126          BIC     #177377,$BDDAT  ;SAVE ONLY DEPTH QUE CONTROL BIT
 1841  016252  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;WAS IT RESTORED ON THE POP INSTR?
 1842  016260  001401                          BEQ     2$              ;BR IF SO
 1843  016262  104055                          ERROR   55              ;DEPTH QUE CONTROL FAILED TO RESTORE
 1844  016264  162737  000400  001124  2$:     SUB     #400,$GDDAT     ;SET UP FOR RESET SATATE
 1845  016272  100407                          BMI     TST67           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1846  016274  042737  000100  032374          BIC     #100,BUFFER     ;SET UP FOR STACKING RESET STATE
 1847  016302  052737  000100  032400          BIS     #100,BUFFER+4   ;SET BIT BEFORE POP INSTR
 1848  016310  000737                          BR      1$              ;GO TRY RESET STATE
 1849
 1850                                  ;;*******************************************************************
  (3)                                  ;*TEST 67       TEST THAT EDGE INTR ENABLED GETS RESTORED-CK IT ON STK
  (3)                                  ;;*******************************************************************
  (2)  016312  000004                  TST67:  SCOPE
  (2)  016314  012737  000067  001206          MOV     #67,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1851  016322  012737  016360  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1852  016330  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
 1853  016336  012737  004000  001124          MOV     #4000,$GDDAT    ;EXPECT IT TO BE SET INITIALLY
 1854  016344  012737  176060  032374          MOV     #176060,BUFFER  ;SET UP LOAD STATUS B' INSTR
 1855  016352  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & LOAD STATUS B' INSTR
 1856  016356  176040                          176040                  ;LOAD STATUS B' INSTR
 1857  016360  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1858  016364  012777  032376  163652          MOV     #BUFFER+2,@DPC  ;DO A JSR
 1859  016372  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1860  016376  000001                          BIT0                    ;COUNT TO 1
 1861  016400  012777  020037  163670          MOV     #20037,@STKPT   ;SET UP STK SEL
 1862  016406  017737  163660  001126          MOV     @STKVAL,$BDDAT  ;READ EDGE INTR ENA BYTE FROM STK
```

```
 1863  016414  042737  173777  001126          BIC     #173777,$BDDAT  ;SAVE ONLY EDGD INTR ENA BIT
 1864  016422  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;WAS IT RESTORED ON THE POP INSTR?
 1865  016430  001401                          BEQ     2$              ;BR IF SO
 1866  016432  104056                          ERROR   56              ;EDGE INTR ENABLE FAILED TO RESTORE
 1867  016434  162737  004000  001124  2$:     SUB     #4000,$GDDAT    ;SET UP FOR RESET STATE
 1868  016442  100407                          BMI     TST70           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1869  016444  042737  000020  032374          BIC     #20,BUFFER      ;SET UP FOR STACKING RESET STATE
 1870  016452  052737  000020  032400          BIS     #20,BUFFER+4    ;SET BIT BEFORE POP INSTR
 1871  016460  000737                          BR      1$              ;GO TRY RESET STATE
 1872
 1873                                  ;;*****************************************************************
  (3)                                  ;*TEST 70        TEST THAT CHAR STRING ESCAPE GETS RESTORED-CK IT ON STK
  (3)                                  ;;*****************************************************************
  (2)  016462  000004                  TST70:  SCOPE
  (2)  016464  012737  000070  001206          MOV     #70,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1874  016472  012737  016530  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1875  016500  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
 1876  016506  012737  010000  001124          MOV     #10000,$GDDAT   ;EXPECT IT TO BE SET INITIALLY
 1877  016514  012737  176003  032374          MOV     #176003,BUFFER  ;SET UP LOAD STATUS B' INSTR
 1878  016522  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & LOAD STATUS B' INSTR
 1879  016526  176002                          176002                  ;LOAD STATUS B' INSTR
 1880  016530  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1881  016534  012777  032376  163502          MOV     #BUFFER+2,@DPC  ;DO A JSR
 1882  016542  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1883  016546  000001                          BIT0                    ;COUNT TO 1
 1884  016550  012777  020037  163520          MOV     #20037,@STKPT   ;SET UP STK SEL
 1885  016556  017737  163510  001126          MOV     @STKVAL,$BDDAT  ;READ CHAR STRING ESCAPE BYTE FROM STK
 1886  016564  042737  167777  001126          BIC     #167777,$BDDAT  ;SAVE ONLY CHAR STRING ESCAPE BIT
 1887  016572  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;WAS IT RESTORED ON THE POP INSTR?
 1888  016600  001401                          BEQ     2$              ;BR IF SO
 1889  016602  104057                          ERROR   57              ;CHARACTER STRING ESCAPE FAILED TO RESTORE
 1890  016604  162737  010000  001124  2$:     SUB     #10000,$GDDAT   ;SET UP FOR RESET STATE
 1891  016612  100407                          BMI     TST71           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
 1892  016614  042737  000001  032374          BIC     #1,BUFFER       ;SET UP FOR STACKING RESET STATE
 1893  016622  052737  000001  032400          BIS     #1,BUFFER+4     ;SET BIT BEFORE POP INSTR
 1894  016630  000737                          BR      1$              ;GO TRY RESET STATE
 1895
 1896                                  ;;*****************************************************************
  (3)                                  ;*TEST 71        TEST THAT L.P. HIT DISABLE GETS RESTORED-CK IT ON STK
  (3)                                  ;;*****************************************************************
  (2)  016632  000004                  TST71:  SCOPE
  (2)  016634  012737  000071  001206          MOV     #71,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1897  016642  012737  016700  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1898  016650  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
 1899  016656  012737  000002  001124          MOV     #2,$GDDAT       ;EXPECT IT TO BE SET INITIALLY
 1900  016664  012737  170300  032374          MOV     #170300,BUFFER  ;SET UP LOAD STATUS A INSTR
 1901  016672  004437  024462                  JSR     R4,POPSET       ;SET UP JSR, POP & LOAD STATUS A INSTR
 1902  016676  170200                          170200                  ;LOAD STATUS A INSTR
 1903  016700  004737  024512          1$:     JSR     PC,DO           ;GO CYCLE THRU INSTRS
 1904  016704  012777  032376  163332          MOV     #BUFFER+2,@DPC  ;DO A JSR
 1905  016712  004537  024446                  JSR     R5,DELAY        ;STALL FOR STACKING
 1906  016716  000001                          BIT0                    ;COUNT TO 1
 1907  016720  012777  020037  163350          MOV     #20037,@STKPT   ;SET UP STK SEL
 1908  016726  017737  163340  001126          MOV     @STKVAL,$BDDAT  ;READ L.P. HIS DISABLE BYTE FROM STK
 1909  016734  042737  177775  001126          BIC     #177775,$BDDAT  ;SAVE ONLY L.P. HIS DISABLE BIT
 1910  016742  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;WAS IT RESTORED ON THE POP INSTR?
```

```
1911  016750  001401                          BEQ     2$              ;BR IF SO
1912  016752  104060                          ERROR   60              ;L.P. HIT DISABLE FAILED TO RESTORE
1913  016754  162737  000002  001124  2$:     SUB     #2,$GDDAT       ;SET UP FOR RESET STATE
1914  016762  100407                          BMI     TST72           ;;ADVANCE TO NEXT TEST IF BOTH STATES TESTED
1915  016764  042737  000100  032374          BIC     #100,BUFFER     ;SET UP FOR STKING RESET STATE
1916  016772  052737  000100  032400          BIS     #100,BUFFER+4   ;SET BIT BEFORE POP INSTR
1917  017000  000737                          BR      1$              ;GO TRY RESET STATE
1918
1919                                  ;;**********************************************************************
 (3)                                  ;*TEST 72        TEST THAT SHIFT OUT WILL GET RESTORED - CK IT ON STK
 (3)                                  ;;**********************************************************************
 (2)  017002  000004                  TST72:  SCOPE
 (2)  017004  012737  000072  001206          MOV     #72,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1920  017012  012737  000340  177776          MOV     #340,PSW        ;SET PRIORITY TO HIGEST LEVEL
1921  017020  012737  017102  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
1922  017026  013737  002272  001122          MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
1923  017034  012737  040000  001124          MOV     #40000,$GDDAT   ;EXPECT SHIFT OUT INITIALLY
1924  017042  012704  032374                  MOV     #BUFFER,R4      ;SET UP START ADRS IN R4
1925  017046  012724  100000                  MOV     #100000,(R4)+   ;SET UP CHAR INSTR
1926  017052  012724  007000                  MOV     #7000,(R4)+     ;SET UP SHIFT OUT CHAR
1927  017056  012724  163001                  MOV     #163001,(R4)+   ;SET UP JSR REL + 4
1928  017062  012724  163000                  MOV     #163000,(R4)+   ;SET UP JSR REL - THIS WILL FORCE SHIFT OUT BACK ON STK
1929  017066  012724  100000                  MOV     #100000,(R4)+   ;SET UP CHAR INSTR TO COM SHIFT OUT
1930  017072  012724  000017                  MOV     #17,(R4)+       ;SET UP CHAR LOC - SHIFT IN
1931  017076  012714  166000                  MOV     #166000,(R4)    ;SET UP POP & RESTORE INSTR
1932  017102  012777  020040  163166  1$:     MOV     #20040,@STKPT   ;SET TOP OF STK
1933  017110  012777  032374  163126          MOV     #BUFFER,@DPC    ;START
1934  017116  012704  000006                  MOV     #6,R4           ;SET UP NPR CNTR
1935  017122  005277  163116          2$:     INC     @DPC            ;ADVANCE TO NEXT INSTR OR DATA
1936  017126  004537  024446                  JSR     R5,DELAY        ;ALLOW TIME FOR EACH OPERATION TO COMPLETE
1937  017132  000004                          BIT2                    ;COUNT TO 4
1938  017134  005304                          DEC     R4              ;COUNT THIS NPR
1939  017136  001371                          BNE     2$              ;BR IF MORE NPRS
1940  017140  012777  020037  163130          MOV     #20037,@STKPT   ;SET UP STK SEL
1941  017146  017737  163120  001126          MOV     @STKVAL,$BDDAT  ;READ REG 26
1942  017154  042737  137777  001126          BIC     #137777,$BDDAT  ;SAVE ONLY SHIFT OUT
1943  017162  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
1944  017170  001401                          BEQ     3$              ;BR IF SO
1945  017172  104061                          ERROR   61              ;SHIFT OUT FAILED TO RESTORE
1946  017174  012737  000017  032376  3$:     MOV     #17,BUFFER+2    ;SET UP FOR NO SHIFT OUT
1947  017202  012737  000016  032406          MOV     #16,BUFFER+12   ;COMPLEMENT IT
1948  017210  162737  040000  001124          SUB     #40000,$GDDAT   ;SET UP FOR RESET STATE
1949  017216  100331                          BPL     1$              ;BR IF RESET STATE NOT TESTED
1950
1951                                  ;;**********************************************************************
 (3)                                  ;*TEST 73        TEST THAT CHAR STRING ESCAPE ON TERMINATE CHAR WILL POP THE STACK
 (3)                                  ;;**********************************************************************
 (2)  017220  000004                  TST73:  SCOPE
 (1)  017222  012737  000010  001166          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 (2)  017230  012737  000073  001206          MOV     #73,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1952  017236  012737  032400  001124          MOV     #BUFFER+4,$GDDAT        ;EXPECT ADRS BUFFER+4 AFTER TERM CHAR
1953  017244  012737  176003  032374          MOV     #176003,BUFFER  ;SET UP CHAR STRING ESCAPE INSTR
1954  017252  012737  163002  032376          MOV     #163002,BUFFER+2        ;SET UP JSR REL INSTR
1955  017260  012737  100000  032404          MOV     #100000,BUFFER+10       ;SET CHAR INSTR
1956  017266  012700  000377                  MOV     #377,R0         ;START WITH MAX TERM CHAR VALUE
1957  017272  012777  020040  162776  1$:     MOV     #20040,@STKPT   ;SET TOP OF STACK
```

```
 1958  017300  010077  162770                MOV     R0,@TERMCH      ;SET UP TERM CHAR
 1959  017304  010037  032406                MOV     R0,BUFFER+12            ;SET UP VALUE FOR INSTR
 1960  017310  012777  032374  162726        MOV     #BUFFER,@DPC    ;START
 1961  017316  012737  017272  001110        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 1962  017324  005277  162714                INC     @DPC            ;ADVANCE TO JSR REL INSTR
 1963  017330  004537  024446                JSR     R5,DELAY        ;STALL FOR STACKING
 1964  017334  000001                        BIT0                    ;COUNT TO 1
 1965  017336  005277  162702                INC     @DPC            ;RESUME - SHOULD PICK UP CHAR INSTR
 1966  017342  013737  002244  001122        MOV     DPC,$BDADR      ;SET UP REG ADRS 00
 1967  017350  005277  162670                INC     @DPC            ;RESUME - PICK UP TERM CHAR
 1968  017354  004537  024446                JSR     R5,DELAY        ;STALL FOR POP
 1969  017360  000020                        BIT4                    ;COUNT TO 20
 1970  017362  017737  162656  001126        MOV     @DPC,$BDDAT     ;READ DPC
 1971  017370  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;DID IT DO THE POP?
 1972  017376  001401                        BEQ     2$              ;BR IF SO
 1973  017400  104062                        ERROR   62              ;TERMINATE CHARACTER FAILED TO POP STACK
 1974  017402  105300              2$:       DECB    R0              ;ADVANCE TO NEXT TERMINATE VALUE
 1975  017404  100732                        BMI     1$              ;BR IF MORE TO TEST
 1976
 1977                                ;;*******************************************************************
  (3)                                ;*TEST 74        TEST THAT CHAR STRING ESCAPE WHEN DISABLED WILL NOT POP THE STACK
  (3)                                ;;*******************************************************************
  (2)  017406  000004               TST74:  SCOPE
  (2)  017410  012737  000074  001206        MOV     #74,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 1978  017416  012737  032410  001124        MOV     #BUFFER+14,$GDDAT       ;EXPECT ADRS DPT3 AFTER TERM CHAR
 1979  017424  012737  176002  032374        MOV     #176002,BUFFER  ;SET UP CHAR STRING ESCAPE INSTR-OFF
 1980  017432  012737  100000  032404        MOV     #100000,BUFFER+10       ;SET UP CHAR INSTR
 1981  017440  012737  000177  032406        MOV     #177,BUFFER+12  ;SET UP ASCII TERM CHAR
 1982  017446  012777  000377  162620        MOV     #377,@TERMCH    ;SET UP TERM CHAR
 1983  017454  012777  020040  162614        MOV     #20040,@STKPT   ;SET TOP OF STACK
 1984  017462  012777  032374  162554        MOV     #BUFFER,@DPC    ;START
 1985  017470  012737  163002  032376        MOV     #163002,BUFFER+2        ;SET UP JSR REL
 1986  017476  005277  162542                INC     @DPC            ;ADVANCE TO JSR INSTR
 1987  017502  004537  024446                JSR     R5,DELAY        ;STALL FOR STACKING
 1988  017506  000001                        BIT0                    ;COUNT TO 1
 1989  017510  005277  162530                INC     @DPC            ;ADVANCE TO CHAR INSTR
 1990  017514  013737  002244  001122        MOV     DPC,$BDADR      ;SET UP REG ADRS 00
 1991  017522  005277  162516                INC     @DPC            ;ADVANCE TO TERM CHAR
 1992  017526  004537  024446                JSR     R5,DELAY        ;STALL FOR POTENTIAL POP
 1993  017532  000020                        BIT4                    ;COUNT TO 20
 1994  017534  017737  162504  001126        MOV     @DPC,$BDDAT     ;READ THE DPC
 1995  017542  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 1996  017550  001401                        BEQ     TST75           ;;GO TO NEXT TEST IF OK
 1997  017552  104062                        ERROR   62              ;TERM CHAR POPPED WHEN CHAR STRING TERM DISABLED
 1998
 1999                                ;;*******************************************************************
  (3)                                ;*TEST 75        TEST THAT STACK OVERFLOW WILL CAUSE AN INTERRUPT
  (3)                                ;;*******************************************************************
  (2)  017554  000004               TST75:  SCOPE
  (1)  017556  012737  000040  001166        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
  (2)  017564  012737  000075  001206        MOV     #75,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 2000  017572  013701  001252                MOV     $VECT1,R1       ;GET BASIC VECTOR ADRS
 2001  017576  042701  160000                BIC     #160000,R1      ; CLR PSW BITS          ** C **
 2002  017602  012761  017724  000010        MOV     #2$,10(R1)      ;SET UP STK OVFL INT ADRS
 2003  017610  012761  000340  000012        MOV     #340,12(R1)     ;SET UP PSW TO 7 ON INT
 2004  017616  013737  002256  001122        MOV     SREG1,$BDADR    ;SET UP REG 12 ADRS
```

```
 2005  017624  012777  020040  162444          MOV     #20040,@STKPT   ;SET TOP OF STK
 2006  017632  012700  000011                  MOV     #11,R0          ;WILL COUNT TO OVFL IN R0
 2007  017636  012737  020000  001124          MOV     #20000,$GDDAT   ;WILL EXPECT STK OVFL
 2008  017644  012737  163000  032374          MOV     #163000,BUFFER  ;SET UP JSR INSTR
 2009  017652  012737  000340  177776  1$:     MOV     #340,PSW        ;SET PSW TO 7
 2010  017660  012777  032374  162356          MOV     #BUFFER,@DPC    ;START
 2011  017666  004537  024446                  JSR     R5,DELAY        ;STALL FOR STKING
 2012  017672  000001                          BIT0                    ;COUNT TO 1
 2013  017674  005037  177776                  CLR     PSW             ;ALLOW INT TO OCCUR
 2014  017700  005300                          DEC     R0              ;COUNT THIS STK OPERATION
 2015  017702  001363                          BNE     1$              ;BR IF NOT TO STK OVFL LEVEL
 2016  017704  017737  162346  001126          MOV     @SREG1,$BDDAT   ;READ REG 12
 2017  017712  042737  157777  001126          BIC     #157777,$BDDAT  ;SAVE ONLY STK OVFL
 2018  017720  104063                          ERROR   63              ;STACK OVERFLOW FAILED TO INTERRUPT
 2019  017722  000424                          BR      4$              ;GO LOOK FOR LOOP ON TEST SWITCH
 2020  017724  022626                  2$:     CMP     (R6)+,(R6)+     ;FIX STK SINCE NO RTI
 2021  017726  032777  020000  162322          BIT     #20000,@SREG1   ;SEE IF STK OVFL BIT SET
 2022  017734  001004                          BNE     3$              ;BR IF SO
 2023  017736  005037  001126                  CLR     $BDDAT          ;INDICATE OVFLO BIT NOT SET
 2024  017742  104063                          ERROR   63              ;STACK OVERFLOW INTERRUPTED BUT NO FLAG
 2025  017744  000413                          BR      4$              ;LOOK FOR LOOP ON TEST SW
 2026  017746  005037  001124          3$:     CLR     $GDDAT          ;EXPECT ZERO
 2027  017752  000005                          RESET                   ;CLR FLAG
 2028  017754  032777  020000  162264          BIT     #20000,@SREG0   ;STILL SET?
 2029  017762  001404                          BEQ     4$              ;BR IF NOT
 2030  017764  012737  020000  001126          MOV     #20000,$BDDAT   ;INDICATE IT IS STILL SET
 2031  017772  104063                          ERROR   63              ;RESET FAILED TO CLR STACK OVERFLOW FLAG
 2032  017774  004737  024646          4$:     JSR     PC,RSTVEC       ;RESTORE STK OVERFLOW VECTOR HALT
 2033
 2034                                  ;;*****************************************************************
  (3)                                  ;*TEST 76       TEST THAT STACK UNDERFLOW WILL CAUSE AN INTERRUPT
  (3)                                  ;;*****************************************************************
  (2)  020000  000004                  TST76:  SCOPE
  (1)  020002  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
  (2)  020010  012737  000076  001206          MOV     #76,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 2035  020016  013701  001252                  MOV     $VECT1,R1       ;GET BASIC VECTOR ADRS
 2036  020022  042701  160000                  BIC     #160000,R1      ; CLR PSW BITS          ** C **
 2037  020026  012761  020150  000010          MOV     #2$,10(R1)      ;SET UP STK UNDERFLOW INT ADRS
 2038  020034  012761  000340  000012          MOV     #340,12(R1)     ;SET UP PSW TO 7 ON INT
 2039  020042  013737  002256  001122          MOV     SREG1,$BDADR    ;SET UP REG 12 ADRS
 2040  020050  012700  000011                  MOV     #11,R0          ;WILL COUNT TO UNDERFLOW IN R0
 2041  020054  012777  020000  162214          MOV     #20000,@STKPT   ;SET STACK PTR TO ZERO
 2042  020062  012737  010000  001124          MOV     #10000,$GDDAT   ;WILL EXPECT STK UNDERFLOW
 2043  020070  012737  165000  032374          MOV     #165000,BUFFER  ;SET UP POP INSTR
 2044  020076  012737  000340  177776  1$:     MOV     #340,PSW        ;SET PSW TO 7
 2045  020104  012777  032374  162132          MOV     #BUFFER,@DPC    ;START
 2046  020112  004537  024446                  JSR     R5,DELAY        ;STALL FOR POPING
 2047  020116  000001                          BIT0                    ;COUNT TO 1
 2048  020120  005037  177776                  CLR     PSW             ;ALLOW INT TO OCCUR
 2049  020124  005300                          DEC     R0              ;COUNT THIS POP OPERATION
 2050  020126  001363                          BNE     1$              ;BR IF NOT ST STK UNDERFLO LEVEL
 2051  020130  017737  162122  001126          MOV     @SREG1,$BDDAT   ;READ REG 12
 2052  020136  043737  167777  001126          BIC     167777,$BDDAT   ;SAVE ONLY STK UNDERFLOW
 2053  020144  104064                          ERROR   64              ;STACK UNDERFLOW FAILED TO INTERRUPT
 2054  020146  000424                          BR      4$              ;GO LOOK FOR LOOP ON TEST SWITCH
 2055  020150  022626                  2$:     CMP     (R6)+,(R6)+     ;FIX STACK SINCE NO RTI
```

```
 2056  020152  032777  010000  162076          BIT     #10000,@SREG1   ;SEE IF STK UNDERFLOW BIT SET
 2057  020160  001004                          BNE     3$              ;BR IF SO
 2058  020162  005037  001126                  CLR     $BDDAT          ;INDICATE UNDERFLOW BIT NOT SET
 2059  020166  104064                          ERROR   64              ;STACK UNDERFLOW INTERRUPTED BUT NO FLAG
 2060  020170  000413                          BR      4$              ;GO LOOK FOR LOOP ON TEST SW
 2061  020172  005037  001124          3$:     CLR     $GDDAT          ;EXPECT ZERO
 2062  020176  000005                          RESET                   ;CLR FLAG
 2063  020200  032777  010000  162040          BIT     #10000,@SREG0   ;STILL SET?
 2064  020206  001404                          BEQ     4$              ;BR IF NOT
 2065  020210  012737  010000  001126          MOV     #10000,$BDDAT   ;INDICATE IT IS STILL SET
 2066  020216  104064                          ERROR   64              ;RESET FAILED TO CLR STACK UNDERFLOW FLAG
 2067  020220  004737  024646          4$:     JSR     PC,RSTVEC       ;RESTORE STK UNDERFLOW VECTOR WITH HALT
 2068
 2069                                  ;;*****************************************************************
  (3)                                  ;*TEST 77        TEST THAT X DELTA LENGTH COMPUTES WITH MAINT SW3 SET
  (3)                                  ;;*****************************************************************
  (2)  020224  000004                  TST77:  SCOPE
  (1)  020226  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
  (2)  020234  012737  000077  001206          MOV     #77,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 2070  020242  000005                          RESET                   ;MAKE SURE ALL FLAGS ARE CLEARED
 2071  020244  012737  020326  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 2072  020252  013737  002250  001122          MOV     XPOS,$BDADR     ;SET UP REG ADRS 04
 2073  020260  012700  032374                  MOV     #BUFFER,R0      ;GET INSTR ADRS POINTER
 2074  020264  012720  154024                  MOV     #154024,(R0)+   ;SET UP UNITY VECTOR SCALE
 2075  020270  012720  114000                  MOV     #114000,(R0)+   ;STORE POINT INSTR
 2076  020274  005020                          CLR     (R0)+           ;X=0
 2077  020276  005020                          CLR     (R0)+           ;Y=0
 2078  020300  012720  110000                  MOV     #110000,(R0)+   ;STORE LONG VECTOR INSTR
 2079  020304  005020                          CLR     (R0)+           ;X TO BE UPDATED
 2080  020306  005020                          CLR     (R0)+           ;Y=0
 2081  020310  012710  172000                  MOV     #172000,(R0)    ;STORE STOP INSTR
 2082  020314  012700  000001                  MOV     #1,R0           ;R0 CONTAINS X VECTOR LENGTH
 2083  020320  012737  000001  001124          MOV     #1,$GDDAT       ;CONTAINS EXPECTED DELTA LENGTH
 2084  020326  010037  032406          1$:     MOV     R0,BUFFER+12    ;SET UP NEXT VECTOR LENGTH
 2085  020332  004537  025044                  JSR     R5,EXECUTE      ;GO START DISPLAY
 2086  020336  000007                          7                       ;RESUME COUNT
 2087
 2088  020340  017737  161704  001126          MOV     @XPOS,$BDDAT    ;DELTA LENGTH READ BACK FROM X POSITION REG
 2089  020346  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 2090  020354  001403                          BEQ     2$              ;BR IF SO
 2091  020356  010037  001164                  MOV     R0,$TMP0        ;SET UP X VECTOR LENGTH FOR ER TYPE
 2092  020362  104065                          ERROR   65              ;X DELTA LENGTH INCORRECT
 2093  020364  005237  001124          2$:     INC     $GDDAT          ;BUMP EXPECTED DELTA LENGTH
 2094  020370  005200                          INC     R0              ;ADVANCE TO NEXT X VECTOR VALUE
 2095  020372  032700  002000                  BIT     #2000,R0        ;HAVE WE DONE 10 BITS?
 2096  020376  001753                          BEQ     1$              ;BR IF NOT
 2097
 2098                                  ;;*****************************************************************
  (3)                                  ;*TEST 100       TEST THAT Y DELTA LENGTH COMPUTES WITH MAINT SW3 SET
  (3)                                  ;;*****************************************************************
  (2)  020400  000004                  TST100: SCOPE
  (1)  020402  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
  (2)  020410  012737  000100  001206          MOV     #100,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
 2099  020416  012737  020502  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 2100  020424  013737  002252  001122          MOV     YPOS,$BDADR     ;SET UP REG ADRS 06
 2101  020432  012700  032374                  MOV     #BUFFER,R0      ;GET INSTR ADRS POINTER
```

```
 2102  020436  012720  154024                    MOV     #154024,(R0)+   ;SET UP UNITY VECTOR SCALE
 2103  020442  012720  114000                    MOV     #114000,(R0)+   ;STORE POINT INSTR
 2104  020446  005020                            CLR     (R0)+           ;X=0
 2105  020450  012720  001777                    MOV     #1777,(R0)+     ;Y=1777
 2106  020454  012720  110000                    MOV     #110000,(R0)+   ;STORE LONG VECTOR INSTR
 2107  020460  005020                            CLR     (R0)+           ;X=0
 2108  020462  005020                            CLR     (R0)+           ;Y TO BE UPDATED
 2109  020464  012710  172000                    MOV     #172000,(R0)    ;STORE STOP INSTR
 2110  020470  012700  020001                    MOV     #20001,R0       ;R0 CONTAINS Y VECTOR LENGTH (DRAW TOP TO BOTTOM)
 2111  020474  012737  000001  001124            MOV     #1,$GDDAT       ;CONTAINS EXPECTED Y DELTA LENGTH
 2112  020502  010037  032410           1$:     MOV     R0,BUFFER+14    ;SET UP NEXT VECTOR LENGTH
 2113  020506  004537  025044                    JSR     R5,EXECUTE      ;GO START DISPLAY
 2114  020512  000007                            7                       ;RESUME COUNT
 2115
 2116  020514  017737  161532  001126            MOV     @YPOS,$BDDAT    ;DELTA LENGTH READ BACK FROM Y POSITION REG
 2117  020522  023737  001124  001126            CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 2118  020530  001403                            BEQ     2$              ;BR IF SO
 2119  020532  010037  001164                    MOV     R0,$TMP0        ;SET UP Y VECTOR LENGTH FOR ER TYPE
 2120  020536  104065                            ERROR   65              ;Y DELTA LENGTH INCORRECT
 2121  020540  005237  001124           2$:     INC     $GDDAT          ;BUMP EXPECTED DELTA LENGTH
 2122  020544  005200                            INC     R0              ;ADVANCE TO NEXT Y VECTOR VALUE
 2123  020546  032700  002000                    BIT     #2000,R0        ;HAVE WE DONE 10 BITS?
 2124  020552  001753                            BEQ     1$              ;BR IF NOT
 2125
 2126                                    ;;*******************************************************************
   (3)                                   ;*TEST 101      TEST THAT DELTA LENGTH FOLLOWS THE LONGER OF X OR Y WITH MAINT SW3 SET
   (3)                                   ;;*******************************************************************
   (2)  020554  000004           TST101: SCOPE
   (1)  020556  012737  000040  001166            MOV     #40,$TIMES      ;;DO 40 ITERATIONS
   (2)  020564  012737  000101  001206            MOV     #101,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
 2127  020572  012737  020652  001110            MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
 2128  020600  013702  002252                    MOV     YPOS,R2         ;R2 CONTAINS REG ADRS OF DELTA (X OR Y POS REG)
 2129  020604  012700  032374                    MOV     #BUFFER,R0      ;GET INSTR ADRS POINTER
 2130  020610  012720  154024                    MOV     #154024,(R0)+   ;SET UP UNITY VECTOR SCALE
 2131  020614  012720  114000                    MOV     #114000,(R0)+   ;STORE POINT INSTR
 2132  020620  005020                            CLR     (R0)+           ;X=0
 2133  020622  005020                            CLR     (R0)+           ;Y=0
 2134  020624  012720  110000                    MOV     #110000,(R0)+   ;STORE LONG VECTOR INSTR
 2135  020630  005020                            CLR     (R0)+           ;X TO BE UPDATED
 2136  020632  005020                            CLR     (R0)+           ;Y TO BE UPDATED
 2137  020634  012710  172000                    MOV     #172000,(R0)    ;STORE STOP INSTR
 2138  020640  012700  001777                    MOV     #1777,R0        ;R0 CONTAINS Y VECTOR LENGTH
 2139  020644  005001                            CLR     R1              ;R1 CONTAINS X VECTOR LENGTH
 2140  020646  010037  001124                    MOV     R0,$GDDAT       ;CONTAINS EXPECTED DELTA LENGTH
 2141  020652  010037  032410           1$:     MOV     R0,BUFFER+14    ;SET UP Y VECTOR LENGTH (1777 TO 0)
 2142  020656  010137  032406                    MOV     R1,BUFFER+12    ;SET UP X VECTOR LENGTH (0 TO 1777)
 2143  020662  004537  025044                    JSR     R5,EXECUTE      ;GO START DISPLAY
 2144  020666  000007                            7                       ;RESUME COUNT
 2145
 2146  020670  010237  001122                    MOV     R2,$BDADR       ;SET UP REG ADRS WHERE DELTA IS
 2147  020674  011237  001126                    MOV     (R2),$BDDAT     ;GREATER OF X OF Y DELTA READ FROM X OR Y POS REG
 2148  020700  023737  001124  001126            CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 2149  020706  001404                            BEQ     2$              ;BR IF SO
 2150  020710  013737  001124  001164            MOV     $GDDAT,$TMP0    ;SET UP LONGEST X OR Y VECTOR LENGTH FOR ER TYPE
 2151  020716  104065                            ERROR   65              ;DELTA LENGTH DOES NO. EQUAL LONGER OF X OR Y
 2152  020720  005300           2$:     DEC     R0              ;DECREASE Y VECTOR VALUE BY 1
```

```
 2153  020722  100413                          BMI     TST102          ;;TO NEXT TEST IF 10 BITS OF X & Y HAVE BEEN TESTED
 2154  020724  005201                          INC     R1              ;INCREASE X VECTOR VALUE BY 1
 2155  020726  020001                          CMP     R0,R1           ;LOOK FOR THE LARGEST OF X OR Y
 2156  020730  101005                          BHI     3$              ;BR IF Y VECTOR STILL GREATER THAN X
 2157  020732  013702  002250                  MOV     XPOS,R2         ;NOW DELTA WILL BE IN XPOS
 2158  020736  010137  001124                  MOV     R1,$GDDAT       ;DELTA LENGTH WILL BE THAT OF X VECTOR
 2159  020742  000743                          BR      1$              ;TRY NEXT VECTOR COMBINATION
 2160  020744  010037  001124          3$:     MOV     R0,$GDDAT       ;DELTA LENGTH WILL BE THAT OF Y VECTOR
 2161  020750  000740                          BR      1$              ;TRY NEXT VECTOR COMBINATION
 2162
 2163                                  ;;*****************************************************************
   (3)                                 ;*TEST 102      CHECK TANGENT WHEN INCREMENTING Y AT 1777 X WITH MAINT SW3 SET
   (3)                                 ;;*****************************************************************
   (2)  020752  000004                  TST102: SCOPE
   (1)  020754  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
   (2)  020762  012737  000102  001206          MOV     #102,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
 2164  020770  012737  021056  001110          MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
 2165  020776  013702  002252                  MOV     YPOS,R2         ;R2 CONTAINS REG ADRS OF TANGENT (X OR Y POS REG)
 2166  021002  013701  002262                  MOV     YDOFF,R1        ;R1 CONTAINS REG ADRS OF HI ORDER TANGENT
 2167  021006  012700  032374                  MOV     #BUFFER,R0      ;GET INSTR ADRS POINTER
 2168  021012  012720  154024                  MOV     #154024,(R0)+   ;SET UP UNITY VECTOR SCALE
 2169  021016  012720  114000                  MOV     #114000,(R0)+   ;STORE POINT INSTR
 2170  021022  005020                          CLR     (R0)+           ;X=0
 2171  021024  005020                          CLR     (R0)+           ;Y=0
 2172  021026  012720  110000                  MOV     #110000,(R0)+   ;STORE LONG VECTOR INSTR
 2173  021032  012720  001777                  MOV     #1777,(R0)+     ;X=1777
 2174  021036  005020                          CLR     (R0)+           ;Y TO BE UPDATED
 2175  021040  012710  172000                  MOV     #172000,(R0)    ;STORE STOP INSTR
 2176  021044  005000                          CLR     R0              ;R0 CONTAINS INCREMENTING Y VECTOR LENGTH (0-1777)
 2177  021046  010037  032410          1$:     MOV     R0,BUFFER+14    ;SET UP Y VECTOR LENGTH (0-1777)
 2178  021052  004737  024732                  JSR     PC,CALTAN       ;GO FIGURE OUT WHAT TAN SHOULD BE
 2179  021056  004537  025044          2$:     JSR     R5,EXECUTE      ;GO START DISPLAY
 2180  021062  000007                          7                       ;RESUME COUNT
 2181
 2182  021064  010237  001122                  MOV     R2,$BDADR       ;SET UP REG ADRS WHERE TANGENT IS
 2183  021070  011237  001126                  MOV     (R2),$BDDAT     ;TANGENT IS READ FROM X OR Y POSITION REG
 2184  021074  011105                          MOV     (R1),R5         ;READ HI ORDER TANGENT BITS
 2185  021076  004737  024706                  JSR     PC,TANCON       ;GO MAKE UP 12 BIT TANGENT
 2186  021102  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
 2187  021110  001403                          BEQ     3$              ;BR IF SO
 2188  021112  010037  001164                  MOV     R0,$TMP0        ;SET UP Y VECTOR LENGTH FOR ER TYPE
 2189  021116  104066                          ERROR   66              ;INCORRECT TANGENT FOR X & Y VALUE INDICATED
 2190  021120  005200                  3$:     INC     R0              ;INCREASE X VECTOR VALUE BY 1
 2191  021122  022700  001777                  CMP     #1777,R0        ;R0 UP TO 1777 YET?
 2192  021126  101347                          BHI     1$              ;BR IF NOT
 2193  021130  103405                          BCS     TST103          ;;NEXT TEST IF R0 = 2000
 2194  021132  013702  002250                  MOV     XPOS,R2         ;TANGENT WILL NOW BE IN XPOS
 2195  021136  013701  002260                  MOV     XDOFF,R1        ;HI ORDER TANGENT BITS NOW IN XDOFF
 2196  021142  000741                          BR      1$              ;ONE MORE TIME
 2197
 2198                                  ;;*****************************************************************
   (3)                                 ;*TEST 103      CHECK TANGENT WHEN INCREMENTING X AT 1777 Y WITH MAINT SW3 SET
   (3)                                 ;;*****************************************************************
   (2)  021144  000004                  TST103: SCOPE
   (1)  021146  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
   (2)  021154  012737  000103  001206          MOV     #103,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
```

```
                                        MOV     #2$,$LPERR      ;SET UP SCOPE LOOP ADRS
                                        MOV     XPOS,R2         ;R2 CONTAINS REG ADRS OF TANGENT (X OR Y POS REG)
                                        MOV     XDOFF,R1        ;R1 CONTAINS REG ADRS OF HI ORDER TANGENT
                                        MOV     #BUFFER,R0      ;GET INSTR ADRS POINTER
                                        MOV     #154024,(R0)+   ;SET UP UNITY VECTOR SCALE
                                        MOV     #114000,(R0)+   ;STORE POINT INSTR
                                        CLR     (R0)+           ;X=0
                                        CLR     (R0)+           ;Y=0
                                        MOV     #110000,(R0)+   ;STORE LONG VECTOR INSTR
                                        CLR     (R0)+           ;X TO BE UPDATED
                                        MOV     #1777,(R0)+     ;Y=1777
                                        MOV     #172000,(R0)    ;STORE STOP INSTR
                                        CLR     R0              ;R0 CONTAINS INCREMENTING X VECTOR LENGTH (0-1777)
                                1$:     MOV     R0,BUFFER+12    ;SET UP X VECTOR LENGTH (0-1777)
                                        JSR     PC,CALTAN       ;GO FIGURE OUT WHAT TAN SHOULD BE
                                2$:     JSR     R5,EXECUTE      ;GO START DISPLAY
                                        7                       ;RESUME COUNT

                                        MOV     R2,$BDADR       ;SET UP REG ADRS WHERE TANGENT IS
                                        MOV     (R2),$BDDAT     ;TANGENT IS READ FROM X OR Y POSITION REG
                                        MOV     (R1),R5         ;READ HI ORDER TANGENT BITS
                                        JSR     PC,TANCON       ;GO MAKE UP 12 BIT TANGENT
                                        CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
                                        BEQ     3$              ;BR IF SO
                                        MOV     R0,$TMP0        ;SET UP X VECTOR LENGTH FOR ER TYPE
                                        ERROR   66              ;INCORRECT TANGENT FOR X & Y VALUE INDICATED
                                3$:     INC     R0              ;INCREASE Y VECTOR VALUE BY 1
                                        CMP     #2000,R0        ;R0 UP TO 2000 YET?
                                        BNE     1$              ;BR IF NOT

                                ;;**********************************************************************
                                ;*TEST 104       TEST THAT DPC BITS 16 & 17 WILL SET & RESET
                                ;;**********************************************************************
                                TST104: SCOPE
                                        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
                                        MOV     #104,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
                                        MOV     #60,$GDDAT      ;EXPECT BOTH BITS INITIALLY
                                        MOV     #6000,R0        ;PUT RELOCATE VALUE IN R0
                                        MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
                                1$:     MOV     #MAINT1,@STKPT  ;SET MAINT SW 1
                                        MOV     R0,@RLO         ;SET UP RELOCATE VALUE
                                        CLR     @DPC            ;SET UP DPC ADRS & START DISPLAY
                                        MOV     SREG1,$BDADR    ;SET UP REG ADRS 12
                                        MOV     @SREG1,$BDDAT   ;READ REG 12
                                        BIC     #177717,$BDDAT  ;SAVE ONLY DPC BITS 17 & 16
                                        CMP     $GDDAT,$BDDAT   ;ARE THEY CORRECT?
                                        BEQ     2$              ;BR IF OK
                                        ERROR   67              ;RELOCATE VALUE FAILED TO SET UP DPC 16 OR 17
                                2$:     SUB     #20,$GDDAT      ;SET UP NEXT ADRS VALUE AT EXPECTED LOC
                                        SUB     #2000,R0        ;SET UP NEXT ADRS VALUE AT R0
                                        BPL     1$              ;BR IF MORE ADRS TO TEST
                                        CLR     $LPERR          ;NO RESET SCOPE LOOP
                                        CLR     $GDDAT          ;EXPECT ZERO
                                        MOV     #6000,@RLO      ;SET BOTH BITS
                                        CLR     @DPC            ;CLR DPC & START - NO NPR
                                        RESET                   ;CLR DPC 16 & 17
```

```
                                                ;;*****************************************************************
                                                ;*TEST 105       TEST THAT DPC GETS PUSHED ONTO STACK AT HIGHEST BOUNDRY BELOW 28K
                                                ;;*****************************************************************
                                        TST105: SCOPE
                                                MOV     #105,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
                                                TST     CENAB           ;DO WE HAVE MORE THAN 8K?
                                                BEQ     TST106          ;;IF NOT - NEXT TEST
                                        1$:     MOV     #3$,$LPERR      ;SET UP SCOPE LOOP ADRS
                                                MOV     STKVAL,$BDADR   ;SET UP REG 26 ADRS
                                                CLR     R2              ;CLR ADRS ROTATIONAL VALUE
                                                MOV     MEMMAX,R0       ;GET LAST 4K MEM BLK
                                                MOV     R0,R1           ;PUT IT INTO R1
                                                MOV     R0,R3           ;PUT IT INTO R3 ALSO
                                                ADD     #20000,R1       ;POINT TO 1ST NON-EXISTENT MEM ADRS IN R1
                                        2$:     MOV     #20040,R4       ;SET UP TOP OF STACK INITIALLY IN R4
                                                MOV     #34,$TMP0       ;SET UP STACK LEVEL WHERE JSR'ED DPC IS STORED
                                                MOV     R3,$GDDAT       ;SAVE EXPECTED JSR RETURN ADRS OF STK
                                                SUB     #2,R3           ;REDUCE ADRS BY 2 TO ALLOW FOR JSR REL INSTR
                                                MOV     (R3),LDRSV1     ;SAVE LOCATION
                                                MOV     #163400,(R3)    ;LOAD UP JSR REL INSTR
                                        3$:     MOV     R4,@STKPT       ;SET STK PTR
                                                MOV     R3,@DPC         ;START AT ADRS IN R3
                                                JSR     R5,DELAY        ;STALL FOR STACKING
                                                BIT0                    ;COUNT TO ONE
                                                MOV     $TMP0,@STKPT    ;SEL STK LEVEL OF STORED DPC
                                                MOV     @STKVAL,$BDDAT  ;READ STORED JSR RETURN ADRS
                                                CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
                                                BEQ     4$              ;BR IF SO
                                                ERROR   5               ;JSR REL FAILED TO PUSH ON STK THE RETURN ADRS
                                        4$:     SUB     #4,R4           ;ADVANCE TO NEXT STK LEVEL
                                                SUB     #4,$TMP0        ;HAVE ALL 8 LEVELS BEEN TESTED?
                                                BPL     3$              ;BR IF NOT
                                                MOV     LDRSV1,(R3)     ;RESTORE MEM LOC
                                                TST     R2              ;IS THIS THE 1ST ADRS PASS?
                                                BNE     5$              ;BR IF NOT
                                                MOV     #2,R2           ;SET UP ADRS ROTATIONAL FACTOR IN R2
                                                BR      6$              ;THIS TIME DON'T ROTATE
                                        5$:     ASL     R2              ;SELECT NEXT ADRS BIT TO LEFT
                                        6$:     MOV     R0,R3           ;GET HIGHEST ADRS BOUNDRY
                                                ADD     R2,R3           ;ADD IN NEXT ADRS BIT FOR TEST
                                                BCS     TST106          ;;TO NEXT TEST IF ADRS GREATER THAN 16 BITS
                                                CMP     R1,R3           ;IS THERE ROOM FOR THIS ADRS BIT?
                                                BCC     2$              ;BR IF SO

                                                ;;*****************************************************************
                                                ;*TEST 106       TEST THAT DPC GETS RESTORED FROM STACK AT HIGHEST BOUNDRY BELOW 28K
                                                ;;*****************************************************************
                                        TST106: SCOPE
                                                MOV     #106,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
                                                TST     CENAB           ;DO WE HAVE MORE THAN 8K?
```

```
2298  021746  001516                          BEQ     TST107          ;;IF NOT - NEXT TEST
2299  021750  012737  022050  001110  1$:     MOV     #3$,$LPERR      ;SET UP SCOPE LOOP ADRS
2300  021756  005002                          CLR     R2              ;CLR ADRS ROTATIONAL VALUE
2301  021760  013700  002214                  MOV     MEMMAX,R0       ;GET LAST 4K MEM BLK
2302  021764  010001                          MOV     R0,R1           ;PUT IT INTO R1
2303  021766  010003                          MOV     R0,R3           ;PUT IT INTO R3 ALSO
2304  021770  062701  020000                  ADD     #20000,R1       ;POINT TO 1ST NON-EXISTENT MEM ADRS IN R1
2305  021774  012737  166000  032374          MOV     #166000,BUFFER  ;SET UP POP INSTR IN LOWEST 8K
2306  022002  012704  020040          2$:     MOV     #20040,R4       ;SET TOP OF STACK INITIALLY IN R4
2307  022006  012737  000034  001164          MOV     #34,$TMP0       ;SET UP STK LEVEL WHERE JSR'ED DPC IS STORED
2308  022014  010337  001124                  MOV     R3,$GDDAT       ;SAVE EXPECTED JSR RETURN ADRS
2309  022020  162703  000004                  SUB     #4,R3           ;REDUCE ADRS BY 4 TO ALLOW FOR JSR ABS INSTR
2310  022024  011337  002210                  MOV     (R3),LDRSV1     ;SAVE LOCATION
2311  022030  016337  000002  002212          MOV     2(R3),LDRSV2    ;SAVE LOCATION
2312  022036  012713  162000                  MOV     #162000,(R3)    ;LOAD UP JSR ABS INSTR
2313  022042  012763  032374  000002          MOV     #BUFFER,2(R3)   ;LOAD UP ABS ADRS TO POP INSTR IN LOWEST 8K
2314  022050  010477  160222          3$:     MOV     R4,@STKPT       ;SET UP CURRENT STACK LEVEL
2315  022054  010377  160164                  MOV     R3,@DPC         ;START AT ADRS IN R3
2316  022060  013737  002244  001122          MOV     DPC,$BDADR      ;SET UP REG ADRS 00
2317  022066  005277  160152                  INC     @DPC            ;PICK UP ABS ADRS (POP INSTR ADRS)
2318  022072  004537  024446                  JSR     R5,DELAY        ;DELAY FOR STACKING
2319  022076  000001                          BIT0                    ;COUNT TO ONE
2320  022100  005277  160140                  INC     @DPC            ;PICK UP POP INSTR
2321  022104  004537  024446                  JSR     R5,DELAY        ;STALL FOR POPPING
2322  022110  000001                          BIT0                    ;COUNT TO ONE
2323  022112  017737  160126  001126          MOV     @DPC,$BDDAT     ;GET RETURN ADRS FOR DPC
2324  022120  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;IS IT CORRECT?
2325  022126  001401                          BEQ     4$              ;BR IF SO
2326  022130  104035                          ERROR   35              ;DPC FAILED TO RESTORE AT ADRS & STK LEVEL INDICATED
2327  022132  162704  000004          4$:     SUB     #4,R4           ;SET UP NEXT STK SELECTION LEVEL
2328  022136  162737  000004  001164          SUB     #4,$TMP0        ;HAVE WE GONE THRU 8 LEVELS THIS ADRS
2329  022144  100341                          BPL     3$              ;BR IF NOT
2330  022146  013723  002210                  MOV     LDRSV1,(R3)+    ;RESTORE MEM LOC
2331  022152  013713  002212                  MOV     LDRSV2,(R3)     ;RESTORE MEM LOC
2332  022156  005702                          TST     R2              ;IS THIS THE 1ST ADRS PASS?
2333  022160  001003                          BNE     5$              ;BR IF NOT
2334  022162  012702  000002                  MOV     #2,R2           ;SET UP ADRS ROTATIONAL FACTOR IN R2
2335  022166  000401                          BR      6$              ;THIS TIME DON'T ROTATE
2336  022170  006302                  5$:     ASL     R2              ;SELECT NEXT ADRS BIT TO LEFT
2337  022172  010003                  6$:     MOV     R0,R3           ;GET HIGHEST 4K BOUNDRY ADRS
2338  022174  060203                          ADD     R2,R3           ;ADD IN NEXT ADRS BIT FOR TEST
2339  022176  103402                          BCS     TST107          ;;TO NEXT TEST IF ADRS GREATER THAN 16 BITS
2340  022200  020103                          CMP     R1,R3           ;IS THERE ROOM FOR THIS ADRS BIT?
2341  022202  103277                          BCC     2$              ;BR AND EXERCISE THIS BIT IF SO
2342
2343                                  ;;*****************************************************************
 (3)                                  ;*TEST 107       TEST THAT THE VS60 CAN OPERATE ABOVE 28K
 (3)                                  ;;*****************************************************************
 (2)  022204  000004                  TST107: SCOPE
 (1)  022206  012737  000040  001166          MOV     #40,$TIMES      ;;DO 40 ITERATIONS
 (2)  022214  012737  000107  001206          MOV     #107,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
2344
2345                                  ;THIS TEST USES THE GRAPHPLOT INCREMENT PART OF THE 'LOAD STATUS B'
2346                                  ;INSTR TO HOLD THE CURRENT 4K MEMORY BLOCK ADRS BEING ADDRESSED
2347                                  ;ABOVE 28K.  THE 6 BITS OF THE GRAPHPLOT HAVE THE FOLLOWING RELATIONSHIP:
2348                                  ;16=32K, 20=36K, 22=40K, 24=44K, ETC. UP TO 76=128K.  THE VS60 PREFORMS
```

```
                                        ;THE FOLLOWING INSTRS IN EA 4K BLK ABOVE 28K - JSR,POP & 'LOAD STATUS
                                        ;B' (LD GRAPHPLOT INC).  AN ERROR IS REPORTED IF THE GRAPHPLOT VALUE
                                        ;RECEIVED IS DIFFERENT THAN THE VALUE WRITTEN TO THE 4K BLK UNDER TEST.

022222  005737  002220                        TST     KTENAB          ;DO WE HAVE UPPER CORE?
022226  001547                                BEQ     TST110          ;;IF NOT GO TO NEXT TEST
022230  100437                        1$:     BMI     3$              ;BR IF MEM ALREADY LOADED WITH DISPLAY CODE
022232  052737  100000  002220                BIS     #100000,KTENAB  ;DO IT 1ST TIME ONLY
022240  012777  001600  157766                MOV     #1600,@KIPAR2   ;START AT 28K
022246  012701  174116                        MOV     #174116,R1      ;GRAPHPLOT VALUE WILL CONTAIN 4K MEM BLK NO
022252  012700  040000                2$:     MOV     #40000,R0       ;USE ADRS ZERO & KIPAR2
022256  052737  001400  177572                BIS     #1400,@#KTSR0   ;ENABLE KT MAINT MODE
022264  012720  163001                        MOV     #163001,(R0)+   ;SET UP JSR REL INSTR
022270  010120                                MOV     R1,(R0)+        ;SET UP LOAD STATUS B INSTR
022272  012710  166000                        MOV     #166000,(R0)    ;SET UP POP INSTR
022276  042737  001400  177572                BIC     #1400,@#KTSR0   ;DISABLE KT
022304  027737  157724  002222                CMP     @KIPAR2,KTMAX   ;HAS INSTR CODE BEEN LOADED THRU CODE
022312  001406                                BEQ     3$              ;BR IF SO
022314  062777  000200  157712                ADD     #200,@KIPAR2    ;ADVANCE TO NEXT 4K BLK
022322  062701  000002                        ADD     #2,R1           ;RECORED MEM BLK IN GRAPHPLOT INC INSTR
022326  000751                                BR      2$              ;GO SET UP INSTRS IN NEXT 4K BLK OF CORE
022330  012777  001600  157676        3$:     MOV     #1600,@KIPAR2   ;SET UP 28K ADRS
022336  005077  157712                4$:     CLR     @RLO            ;CLR RELOCATE REG
022342  017700  157666                        MOV     @KIPAR2,R0      ;GET CURRENT 4K BLK ADRS
022346  004737  024576                        JSR     PC,DPCONV       ;GO CONVERT 4K BLK ADRS INTO 18 BIT ADRS
022352  010037  001124                        MOV     R0,$GDDAT       ;SAVE 16 BIT ADRS
022356  010137  002224                        MOV     R1,G1716        ;SAVE ADRS BITS 17 & 16
022362  006201                                ASR     R1              ;NOW SET UP RELOCATION REG
022364  103003                                BCC     5$              ;BR IF ADRS BIT 16 IS NOT SET
022366  052777  002000  157660                BIS     #2000,@RLO      ;SET UP RELOCATION REG BIT 10
022374  006201                        5$:     ASR     R1              ;LOOK AT ADRS BIT 17
022376  103003                                BCC     6$              ;BR IF ADRS BIT 17 IS NOT SET
022400  052777  004000  157646                BIS     #4000,@RLO      ;SET UP RELOCATION REG BIT 11
022406  012777  020040  157662        6$:     MOV     #20040,@STKPT   ;SET TOP OF STACK
022414  013777  001124  157622                MOV     $GDDAT,@DPC     ;START
022422  004537  024446                        JSR     R5,DELAY        ;GO STALL FOR STACKING
022426  000001                                BITO                    ;COUNT TO 1
022430  005277  157610                        INC     @DPC            ;RESUME
022434  004537  024446                        JSR     R5,DELAY        ;GO STALL FOR POPPING
022440  000001                                BITO                    ;COUNT TO 1
022442  005277  157576                        INC     @DPC            ;RESUME
022446  012737  022406  001110                MOV     #6$,$LPERR      ;SET UP SCOPE LOOP ADRS
022454  005001                                CLR     R1              ; CLR HI ORDER BITS     ** C **
022456  017700  157566                        MOV     @XPOS,R0        ;GET ADRS BLK FROM GRAPHPLOT INC BITS
022462  042700  001777                        BIC     #1777,R0        ;SAVE ONLY ADRS BLK INFORMATION READ
022466  004737  024610                        JSR     PC,DPCON1       ;GO CONVERT BLK ADRS TO A 18 BIT ADRS
022472  010037  001126                        MOV     R0,$BDDAT       ;SAVE 16 BIT ADRS READ
022476  010137  002226                        MOV     R1,B1716        ;SAVE ADRS BIT 17 & 16
022502  023737  001124  001126                CMP     $GDDAT,$BDDAT   ;IS THE 16 BIT ADRS CORRECT?
022510  001004                                BNE     7$              ;BR IF NOT
022512  023737  002224  002226                CMP     G1716,B1716     ;ADRS BIT 17 & 16 CORRECT?
022520  001401                                BEQ     8$              ;BR IF SO
022522  104070                        7$:     ERROR   70              ;VS60 FAILED TO EXECUTE INSTRS AT ADRS INDICATED
022524  062777  000200  157502        8$:     ADD     #200,@KIPAR2    ;SET NEXT 4K BLK
022532  027737  157476  002222                CMP     @KIPAR2,KTMAX   ;HAS INSTR CODE BEEN EXECUTED IN ALL 4K AVAIL BLKS?
022540  101676                                BLOS    4$              ;BR IF NOT
```

```
 2405  022542  005077  157506                CLR     @RLO            ;INSURE RELOCATE REG 0 BEFORE ADVANCING
 2406
 2407                                ;;*****************************************************************
   (3)                               ;*TEST 110      SILO TEST 1 - ALL STATUS READABLE FROM BUS DIRECTLY
   (3)                               ;;*****************************************************************
   (2)  022546  000004               TST110: SCOPE
   (1)  022550  012737  000040  001166        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
   (2)  022556  012737  000110  001206        MOV     #110,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
 2408  022564  012700  032374               MOV     #BUFFER,R0      ;GET ADRS WHERE INSTRS GO
 2409  022570  012720  152525               MOV     #152525,(R0)+   ;NAME OF 2525
 2410  022574  012720  155661               MOV     #155661,(R0)+   ;ROTATE,CHAR SC=1,VEC SC=1
 2411  022600  012720  164374               MOV     #164374,(R0)+   ;CONS0=INT,L.P. INTR EN,L.P. SW INTR EN
 2412  022604  012720  164750               MOV     #164750,(R0)+   ;CONS1=INT,-L.P. INTR EN,-L.P. SW INTR EN
 2413  022610  012720  170063               MOV     #170063,(R0)+   ;ITALICS,MENU
 2414  022614  012720  175200               MOV     #175200,(R0)+   ;COLOR=1
 2415  022620  012720  112235               MOV     #112235,(R0)+   ;LONG VEC,INT=1,BLINK,LINE=1,MODE=2
 2416  022624  005020                       CLR     (R0)+           ;X=0
 2417  022626  005020                       CLR     (R0)+           ;Y=0
 2418  022630  012720  164000               MOV     #164000,(R0)+   ;NOP
 2419  022634  012720  151252               MOV     #151252,(R0)+   ;NAME=1252
 2420  022640  012720  155322               MOV     #155322,(R0)+   ;-ROTATE,CHAR SC=2,VEC SC=2
 2421  022644  012720  164270               MOV     #164270,(R0)+   ;CONS0=-INT,L.P. INTR EN,-L.P. SW INTR EN
 2422  022650  012720  164674               MOV     #164674,(R0)+   ;CONS1=-INT,L.P. INTR EN,L.P. SW INTR EN
 2423  022654  012720  170042               MOV     #170042,(R0)+   ;-ITALICS,-MENU
 2424  022660  012720  175400               MOV     #175400,(R0)+   ;COLOR=2
 2425  022664  012720  146426               MOV     #146426,(R0)+   ;ABS VEC,INT=2,-BLINK,LINE=2,MODE=11
 2426  022670  005020                       CLR     (R0)+           ;X=0
 2427  022672  005020                       CLR     (R0)+           ;Y=0
 2428
 2429  022674  012720  164000               MOV     #164000,(R0)+   ;NOP
 2430  022700  012720  153070               MOV     #153070,(R0)+   ;NAME=3070
 2431  022704  012720  155664               MOV     #155664,(R0)+   ;ROTATE,CHAR SC=1,VEC SC=4
 2432  022710  012720  164374               MOV     #164374,(R0)+   ;CONS0=INT,L.P. INTR EN,L.P. SW INTR EN
 2433  022714  012720  164750               MOV     #164750,(R0)+   ;CONS1=INT,-L.P. INTR EN,-L.P. SW INTR EN
 2434  022720  012720  170063               MOV     #170063,(R0)+   ;ITALICS,MENU
 2435  022724  012720  175200               MOV     #175200,(R0)+   ;COLOR=1
 2436  022730  012720  113035               MOV     #113035,(R0)+   ;LONG VEC,INT=4,BLINK,LINE=1,MODE=2
 2437  022734  005020                       CLR     (R0)+           ;X=0
 2438  022736  005020                       CLR     (R0)+           ;Y=0
 2439  022740  012720  164000               MOV     #164000,(R0)+   ;NOP
 2440  022744  012720  150707               MOV     #150707,(R0)+   ;NAME=0707
 2441  022750  012720  155330               MOV     #155330,(R0)+   ;-ROTATE,CHAR SC=2,VEC SC=10
 2442  022754  012720  164270               MOV     #164270,(R0)+   ;CONS0=-INT,L.P. INTR EN,-L.P. SW INTR EN
 2443  022760  012720  164674               MOV     #164674,(R0)+   ;CONS1=-INT,L.P. INTR EN,L.P. SW INTR EN
 2444  022764  012720  175400               MOV     #175400,(R0)+   ;COLOR=2
 2445  022770  012720  170042               MOV     #170042,(R0)+   ;-ITALICS,-MENU
 2446  022774  012720  146026               MOV     #146026,(R0)+   ;ABS VEC,INT=0,-BLINK,LINE=2,MODE=11
 2447  023000  005020                       CLR     (R0)+           ;X=0
 2448  023002  005020                       CLR     (R0)+           ;Y=0
 2449  023004  012720  161730               MOV     #161730,(R0)+   ;JMP REL TO BUFFER START
 2450
 2451  023010  013700  001252               MOV     $VECT1,R0       ;GET BASIC VECTOR ADRS
 2452  023014  042700  160000               BIC     #160000,R0      ; CLR PSW BITS          ** C **
 2453  023020  012760  023202  000004       MOV     #3$,4(R0)       ;SET UP INTR RETURN ADRS
 2454  023026  012760  000340  000006       MOV     #340,6(R0)      ;SET PRIORITY TO HIGHEST LEVEL ON INTR
 2455  023034  012701  023432               MOV     #BSTART,R1      ;R1 CONTAINS ADRS PTR OF DISPLAY STARTS
```

```
2456  023040  012702  023344                  MOV     #SILOR,R2       ;R2 CONTAINS ADRS PTR OF EXPECTED SILO RESULTS
2457  023044  012703  023444                  MOV     #RCNT,R3        ;R3 CONTAINS ADRS PTR OF # OF RESUMES FOR EA SILO LEVEL
2458  023050  012704  023416                  MOV     #BADR,R4        ;R4 CONTAINS ADRS PTR WHERE REG DATA IS EXPECTED
2459  023054  012705  023344                  MOV     #SILOR,R5       ;R5 WILL UPDATE R2 ON NEXT BUF START ADRS
2460  023060  005037  001164                  CLR     $TMP0           ;$TMP0 CONTAINS SILO LEVEL UNDER TEST
2461  023064  012777  020040  157204  1$:     MOV     #20040,@STKPT   ;TEST DONE WITH MAINT SW 2
2462  023072  012737  000340  177776          MOV     #340,PSW        ;WANT PRIORITY LEVEL AT TOP
2463  023100  011177  157140                  MOV     (R1),@DPC       ;START
2464  023104  011300                          MOV     (R3),R0         ;SET UP RESUME CNTR
2465  023106  005277  157132          2$:     INC     @DPC            ;RESUME
2466  023112  012737  023064  001110          MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
2467  023120  005300                          DEC     R0              ;COUNT RESUME
2468  023122  001371                          BNE     2$              ;BR IF CNT NOT TO ZERO
2469  023124  052777  040000  157144          BIS     #40000,@STKPT   ;SET MAINT SW 3
2470  023132  052777  040000  157126          BIS     #40000,@CONS    ;SET L.P. FLAG 00
2471  023140  005037  177776                  CLR     PSW             ;ALLOW INTR
2472  023144  021616                          CMP     (SP),(SP)       ;LET CPU GRANT REQ
2473  023146  021616                          CMP     (SP),(SP)       ;
2474  023150  021616                          CMP     (SP),(SP)       ;
2475  023152  021616                          CMP     (SP),(SP)       ;
2476  023154  013737  002266  001122          MOV     CONS,$BDADR     ;SET UP REG ADRS 22
2477  023162  012737  144000  001124          MOV     #144000,$GDDAT  ;EXPECT L. P. FLAG
2478  023170  017737  157072  001126          MOV     @CONS,$BDDAT    ;READ REG 22
2479  023176  104001                          ERROR   1               ;L.P. FLAG INTR FAILED USING MAINT MODE 3
2480  023200  000455                          BR      7$              ;NEXT TEST ON INTR FAILURE
2481  023202  022626                  3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI ON INTR
2482  023204  017437  000000  001122  4$:     MOV     @0(R4),$BDADR   ;GET BUS ADRS WHERE DATA IS
2483  023212  017737  155704  001126          MOV     @$BDADR,$BDDAT  ;GET DATA FROM REG
2484  023220  011237  001124                  MOV     (R2),$GDDAT     ;GET EXPECTED DATA FROM SILOR TABLE
2485  023224  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;CORRECT?
2486  023232  001401                          BEQ     5$              ;BR IF SO
2487  023234  104002                          ERROR   2               ;SILO FAILURE AT LEVEL INDICATED
2488  023236  062702  000002          5$:     ADD     #2,R2           ;BUMP ADRS PTR OF EXPECTED SILO RESULTS
2489  023242  062704  000002                  ADD     #2,R4           ;BUMP ADRS PTR OF REG ADRS WHERE DATA IS READ
2490  023246  005714                          TST     (R4)            ;DONE CHECKS?
2491  023250  001355                          BNE     4$              ;BR IF NOT
2492  023252  012704  023416                  MOV     #BADR,R4        ;RESET REG ADRS PTR
2493  023256  005712                          TST     (R2)            ;AT END OF RESULT TABLE?
2494  023260  001002                          BNE     6$              ;BR IF NOT
2495  023262  012702  023344                  MOV     #SILOR,R2       ;RESET ADRS PTR OF EXPECTED RESULTS
2496  023266  062703  000002          6$:     ADD     #2,R3           ;POINT TO NEXT RESUME CNT FOR NEXT SILO LEVEL
2497  023272  005237  001164                  INC     $TMP0           ;ADVANCE TO NEXT SILO LEVEL
2498  023276  022737  000004  001164          CMP     #4,$TMP0        ;ALL 4 LEVELS TESTED AT THIS STARTING ADRS?
2499  023304  001267                          BNE     1$              ;BR IF NOT - REPEAT TEST CHECKING NEXT SILO LEVEL
2500  023306  005037  001164                  CLR     $TMP0           ;RESET SILO LEVEL FOR TESTING WITH DIFF DATA
2501  023312  012703  023444                  MOV     #RCNT,R3        ;RESET RESUME CNT ADRS PTR
2502  023316  062705  000012                  ADD     #12,R5          ;NEW RESULT ADRS FOR NEXT BUF START ADRS
2503  023322  010502                          MOV     R5,R2           ;UPDATE RESULT PTR
2504  023324  062701  000002                  ADD     #2,R1           ;BUMP BUFFER STARTING ADRS PTR - THIS CAUSES
2505                                                                  ;DIFFERENT DATA PATTERNS THRU SILO
2506  023330  005711                          TST     (R1)            ;ALL 4 BUFFER ADRS STARTS BEEN DONE?
2507  023332  001254                          BNE     1$              ;BR IF NOT
2508  023334  004737  024646          7$:     JSR     PC,RSTVEC       ;GO RESTORE VECTOR WITH HALT
2509  023340  000005                          RESET                   ;CLR VS60 BEFORE ADVANCING
2510  023342  000445                          BR      TST111          ;;TO NEXT TEST
2511
```

```
 2512  023344  032416                SILOR:  BUFFER+22       ;DPC RETURN ADRS
 2513  023346  010631                        10631   ;MODE=2,INT=1,L.P. FLAG,ITALICS,BLINK,LINE=1
 2514  023350  002501                        2501    ;ROTATE,CHAR SC=1,MENU,VCT CS=1
 2515  023352  147004                        147004  ;CONS0=INT,L.P. FLG,L.P. INTR EN,L.P. SW EN,CONS1=INT,COLOR=1
 2516  023354  002525                        2525    ;NAME=2525
 2517  023356  032442                        BUFFER+46       ;DPC RETURN ADRS
 2518  023360  045202                        45202   ;MODE=11,INT=2,L.P. FLG,LINE=2
 2519  023362  001002                        1002    ;CHAR SC=2,VCT SC=2
 2520  023364  044070                        44070   ;CONS0=L.P.FLG,L.P. INTR EN,CONS1=L.P. FLG INTR EN,L.P. SW EN,COLOR=2
 2521  023366  001252                        1252    ;NAME=1252
 2522  023370  032466                        BUFFER+72       ;DPC RETURN ADRS
 2523  023372  012231                        12231   ;MODE=2,INT=3,L.P. FLG,ITALICS,BLINK,LINE=1
 2524  023374  002504                        2504    ;ROTATE,CHAR SC=1,MENU,VEC SC=4
 2525  023376  147004                        147004  ;CONS0=INT,L.P. FLG,L.P. INTR EN,L.P. SW EN,CONS1=INT,COLOR=1
 2526  023400  003070                        3070    ;NAME=3070
 2527  023402  032512                        BUFFER+116      ;DPC RETURN ADRS
 2528  023404  044202                        44202   ;MODE=11,INT=0,L.P. FLG,LINE=2
 2529  023406  001010                        1010    ;CHAR SC=2,VCT SC=4
 2530  023410  044070                        44070   ;CONS0=L.P. FLG,L.P. INTR EN,CONS1=L.P. FLG FLG EN,L.P. SE EN,COLOR=2
 2531  023412  000707                        707     ;NAME=707
 2532  023414  000000                        0       ;TERMINATOR
 2533
 2534  023416  002244                BADR:   DPC     ;THIS TABLE CONTAINS REGS ADRS WHERE DATA IS EXPECTED
 2535  023420  002246                        SREG0
 2536  023422  002256                        SREG1
 2537  023424  002266                        CONS
 2538  023426  002270                        DNAME
 2539  023430  000000                        0
 2540
 2541  023432  032374                BSTART: BUFFER  ;THIS TABLE CONTAINS DIFFERENT STARTING ADRS FOR TESTING
 2542                                        ;EACH SILO LEVEL
 2543  023434  032420                        BUFFER+24
 2544  023436  032444                        BUFFER+50
 2545  023440  032470                        BUFFER+74
 2546  023442  000000                        0
 2547
 2548  023444  000017                RCNT:   17      ;THIS TABLE CONTROLS THE AMOUNT OF RESUMES NEEDED TO
 2549                                        ;CYCLE THRU ALL FOUR LEVELS OF SILO
 2550  023446  000031                        31
 2551  023450  000043                        43
 2552  023452  000055                        55
 2553  023454  000000                        0
 2554
 2555                                ;;*****************************************************************
  (3)                                ;*TEST 111       SILO TEST 2 - ALL STATUS VERIFIED ON STACK
  (3)                                ;;*****************************************************************
  (2)  023456  000004                TST111: SCOPE
  (1)  023460  012737  000040  001166        MOV     #40,$TIMES      ;;DO 40 ITERATIONS
  (2)  023466  012737  000111  001206        MOV     #111,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
 2556  023474  012700  032374                MOV     #BUFFER,R0      ;GET ADRS WHERE INSTR'S GO
 2557  023500  012720  171600                MOV     #171600,(R0)+   ;STOP INTR EN,-L.P. HIT EN
 2558  023504  012720  176357                MOV     #176357,(R0)+   ;QUE CNTRL,-EDGE FLG INTR EN,QUE,CHAR STRING ESC
 2559  023510  012720  164770                MOV     #164770,(R0)+   ;CONS1=INT,L.P. INTR EN
 2560  023514  012720  102200                MOV     #102200,(R0)+   ;CHAR INSTR,INT=1
 2561  023520  012720  007015                MOV     #7015,(R0)+     ;SHIFT OUT & 'CR'
 2562  023524  012720  164000                MOV     #164000,(R0)+   ;NOP OR JSR REL +1 WHEN CHECKING SILO
```

```
2563  023530  012720  164000                    MOV     #164000,(R0)+   ;NOP
2564  023534  012720  171300                    MOV     #171300,(R0)+   ;-STOP INTR EN,L.P. HIT EN
2565  023540  012720  176272                    MOV     #176272,(R0)+   ;-QUE CNTRL,EDGE FLG INTR EN,-QUE,-CHAR STRING ESC
2566  023544  012720  164770                    MOV     #164770,(R0)+   ;CONS1=INT,L.P. INTR EN
2567  023550  012720  102400                    MOV     #102400,(R0)+   ;CHAR INSTR,INT=2
2568  023554  012720  006417                    MOV     #6417,(R0)+     ;SHIFT OUT OFF & 'CR'
2569  023560  012720  164000                    MOV     #164000,(R0)+   ;NOP OR JSR REL +1 WHEN CHECKING SILO
2570  023564  012720  164000                    MOV     #164000,(R0)+   ;NOP
2571  023570  012720  171600                    MOV     #171600,(R0)+   ;STOP INTR EN,-L.P. HIT EN
2572  023574  012720  176357                    MOV     #176357,(R0)+   ;QUE CNTRL,-EDGE FLG INTR EN,QUE,CHAR STRING ESC
2573  023600  012720  164770                    MOV     #164770,(R0)+   ;CONS1=INT,L.P. INTR EN
2574  023604  012720  102600                    MOV     #102600,(R0)+   ;CHAR INSTR,INT=3
2575  023610  012720  007015                    MOV     #7015,(R0)+     ;SHIFT OUT & 'CR'
2576  023614  012720  164000                    MOV     #164000,(R0)+   ;NOP OR JSR REL +1 WHEN CHECKING SILO
2577  023620  012720  164000                    MOV     #164000,(R0)+   ;NOP
2578  023624  012720  171300                    MOV     #171300,(R0)+   ;-STOP INTR EN,L.P. HIT EN
2579  023630  012720  176272                    MOV     #176272,(R0)+   ;-QUE CNTRL,EDGE FLG INTR EN,-QUE,-CHAR STRING ESC
2580  023634  012720  164770                    MOV     #164770,(R0)+   ;CONS1=INT,L.P INTR EN
2581  023640  012720  103000                    MOV     #103000,(R0)+   ;CHAR INSTR,INT=4
2582  023644  012720  006417                    MOV     #6417,(R0)+     ;SHIFT OUT OFF & 'CR'
2583  023650  012720  164000                    MOV     #164000,(R0)+   ;NOP OR JSR REL +1 WHEN CHECKING SILO
2584  023654  012720  161744                    MOV     #161744,(R0)+   ;JMP REL TO BUFFER START
2585  023660  013700  001252                    MOV     $VECT1,R0       ;GET BASIC VECTOR ADRS
2586  023664  042700  160000                    BIC     #160000,R0      ; CLR PSW BITS          ** C **
2587  023670  012760  024050  000004            MOV     #3$,4(R0)       ;SET UP INTR RETURN ADRS
2588  023676  012760  000340  000006            MOV     #340,6(R0)      ;HIGHEST PRIORITY ON INTR
2589  023704  012737  023732  001110            MOV     #1$,$LPERR      ;SET UP SCOPE LOOP ADRS
2590  023712  012701  024256                    MOV     #BSTRT,R1       ;R1 CONTAINS ADRS PTR OF DISPLAY STARTS
2591  023716  012702  024250                    MOV     #SILOR1,R2      ;R2 CONTAINS ADRS PTR OF EXPECTED SILO DATA
2592  023722  012703  024270                    MOV     #RCNT1,R3       ;R3 CONTAINS ADRS PTR OF # OF RESUMES FOR EA SILO LEVEL
2593  023726  005037  001164                    CLR     $TMP0           ;$TMP0 CONTAINS SILO LEVEL UNDER TEST
2594  023732  012777  020040  156336  1$:       MOV     #20040,@STKPT   ;SET MAINT SW 2 & TOP OF STACK
2595  023740  012737  000340  177776            MOV     #340,PSW        ;WANT PRIORITY AT TOP
2596  023746  011177  156272                    MOV     (R1),@DPC       ;START
2597  023752  011300                            MOV     (R3),R0         ;SET UP RESUME CNTR
2598  023754  005277  156264          2$:       INC     @DPC            ;RESUME
2599  023760  004537  024446                    JSR     R5,DELAY        ;STALL
2600  023764  000004                            BIT2                    ;COUNT TO 4
2601  023766  005300                            DEC     R0              ;COUNT RESUME
2602  023770  001371                            BNE     2$              ;BR IF COUNT NOT TO ZERO
2603  023772  052777  040000  156276            BIS     #40000,@STKPT   ;SET MAINT SW 3
2604  024000  052777  000400  156260            BIS     #400,@CONS      ;SET L.P. FLAG 01
2605  024006  005037  177776                    CLR     PSW             ;ALLOW INTR
2606  024012  021616                            CMP     (SP),(SP)       ;LET CPU GRANT REQ
2607  024014  021616                            CMP     (SP),(SP)       ;
2608  024016  021616                            CMP     (SP),(SP)       ;
2609  024020  021616                            CMP     (SP),(SP)       ;
2610  024022  013737  002266  001122            MOV     CONS,$BDADR     ;SET UP ADRS REG 22
2611  024030  012737  101440  001124            MOV     #101440,$GDDAT  ;EXPECT L.P. FLAG 01,INTS,L.P. EN
2612  024036  017737  156224  001126            MOV     @CONS,$BDDAT    ;READ REG 22
2613  024044  104001                            ERROR   1               ;L.P. FLAG 01 FAILED TO INTR USING MAINT MODE 3
2614  024046  000474                            BR      7$              ;NEXT TEST ON INTR FAILURE
2615  024050  022626                    3$:     CMP     (SP)+,(SP)+     ;FIX STACK SINCE NO RTI
2616  024052  017737  156166  024246            MOV     @DPC,DPCSAV     ;GET DPC ADRS
2617  024060  012777  163000  000160            MOV     #163000,@DPCSAV ;SET UP JSR REL AT SILO RESTORED DPC
2618  024066  013777  024246  156150            MOV     DPCSAV,@DPC     ;SHOULD DO A JSR NOW(RESTORED SILO DATA TO STACK)
```

```
2619  024074  004537  024446              JSR     R5,DELAY        ;STALL FOR STACKING
2620  024100  000001                      BIT0                    ;COUNT TO 1
2621  024102  012777  164000  000136      MOV     #164000,@DPCSAV ;RESTORE NOP INSTR FOR NEXT STARTS
2622  024110  013737  002272  001122      MOV     STKVAL,$BDADR   ;SET UP REG ADRS 26
2623  024116  052777  000037  156152      BIS     #37,@STKPT      ;DATA IS IN BYTE 3 OF STACK LEVEL 1
2624  024124  017737  156142  001126      MOV     @STKVAL,$BDDAT  ;READ SAVE STATUS FROM STACK
2625  024132  011237  001124              MOV     (R2),$GDDAT     ;GET EXPECTED STATUS
2626  024136  023737  001124  001126      CMP     $GDDAT,$BDDAT   ;CORRECT?
2627  024144  001401                      BEQ     4$              ;BR IF SO
2628  024146  104002                      ERROR   2               ;SILO FAILED AT LEVEL INDICATED - DATA VERIFIED FROM STK
2629  024150  062702  000002          4$: ADD     #2,R2           ;BUMP ADRS PTR OF EXPECTED SILO RESULTS
2630  024154  005712                      TST     (R2)            ;AT END OR RESULT TABLE?
2631  024156  001002                      BNE     5$              ;BR IF NOT
2632  024160  012702  024250              MOV     #SILOR1,R2      ;RESET ADRS PTR OF EXPECTED RESULTS
2633  024164  062703  000002          5$: ADD     #2,R3           ;POINT TO NEXT RESUME CNT FOR NEXT SILO LEVEL
2634  024170  005237  001164              INC     $TMP0           ;ADVANCE TO NEXT SILO LEVEL
2635  024174  022737  000004  001164      CMP     #4,$TMP0        ;ALL 4 LEVELS TESTED AT THIS STARTING ADRS?
2636  024202  001253                      BNE     1$              ;BR IF NOT - REPEAT TEST CHECKING NEXT SILO LEVEL
2637  024204  005037  001164              CLR     $TMP0           ;RESET SILO LEVEL FOR TESTING WITH DIFF DAT
2638  024210  012703  024270              MOV     #RCNT1,R3       ;RESET RESUME CNT ADRS PTR
2639  024214  062702  000002              ADD     #2,R2           ;NEW RESULT ADRS FOR NEXT BUF STARTING ADRS
2640  024220  005712                      TST     (R2)            ;END OF TABLE?
2641  024222  001002                      BNE     6$              ;BR IF NOT
2642  024224  012702  024250              MOV     #SILOR1,R2      ;RESET RESULT ADRS
2643  024230  062701  000002          6$: ADD     #2,R1           ;BUMP BUFFER STARTING ADRS PTR - THIS
2644                                                              ;CAUSES DIFFERNT DATA PATTERNS THRU SILO
2645  024234  005711                      TST     (R1)            ;ALL 4 BUFFER ADRS STARTS BEEN DONE?
2646  024236  001235                      BNE     1$              ;BR IF NOT
2647  024240  004737  024646          7$: JSR     PC,RSTVEC       ;GO RESTORE VECTOR WITH HALT
2648  024244  000416                      BR      TST112          ;;GO TO NEXT TEST
2649
2650  024246  000000              DPCSAV: 0       ;THIS LOCATION SAVES THE DPC THAT WAS RESTORED FROM THE SILO
2651
2652  024250  053465              SILOR1: 53465   ;SHIFT OUT,CHAR STRING ESC,-EDGE FLG INTR EN,QUE,QUE CNTRL
2653                                              ;L.P. INTR EN 01,INT 00&01,-L.P. HIT DIS,STOP INTR EN
2654  024252  005066                      5066    ;-SHIFT OUT,-CHAR STRING ESC,EDGE FLG INTR EN,-QUE,-QUE CNTRL,
2655                                              ;L.P. INTR EN 01,INT 00&01,L.P. HIT DIS,-STOP INTR EN
2656  024254  000000                      0       ;TERMINATOR
2657
2658  024256  032374              BSTRT:  BUFFER  ;THIS TABLE CONTAINS DIFFERENT STARTING ADRS FOR TESTING
2659                                              ;EACH SILO LEVEL
2660  024260  032412                      BUFFER+16
2661  024262  032430                      BUFFER+34
2662  024264  032446                      BUFFER+52
2663  024266  000000                      0
2664
2665  024270  000011              RCNT1:  11      ;THIS TABLE CONTROLS THE  AMOUNT OF RESUMES NEEDED TO
2666  024272  000020                      20      ;CYCLE THRU ALL FOUR LEVELS OF SILO
2667  024274  000027                      27
2668  024276  000036                      36
2669  024300  000000                      0
2670
2671                          ;;*****************************************************************
 (3)                          ;*TEST 112      TEST FOR HALT AT END OF DIAGNOSTIC (SW12 SET)
 (3)                          ;;*****************************************************************
 (2)  024302  000004          TST112: SCOPE
```

```
  (2)  024304  012737  000112  001206          MOV     #112,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
 2672  024312  032777  010000  154620          BIT     #BIT12,@SWR     ;IS SR12 SET?
 2673  024320  001401                          BEQ     $EOP            ;BR IF NOT
 2674  024322  000000                          HALT                    ;COMPLETED PASS - SW12 SAYS HALT
```

```
 2676                                   .SBTTL  END OF PASS ROUTINE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*INCREMENT THE PASS NUMBER ($PASS)
  (1)                                   ;*TYPE 'END PASS #XXXXX'' (WHERE XXXXX IS A DECIMAL NUMBER)
  (1)                                   ;*IF THERES A MONITOR GO TO IT
  (1)                                   ;*IF THERE ISN'T JUMP TO RSTART
  (1)
  (1)  024324                           $EOP:
  (1)  024324  000004                           SCOPE
  (1)  024326  005037  001102                   CLR     $TSTNM          ;;ZERO THE TEST NUMBER
  (1)  024332  005037  001166                   CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
  (1)  024336  005237  001210                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
  (1)  024342  042737  100000  001210           BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
  (1)  024350  005327                           DEC     (PC)+           ;;LOOP?
  (1)  024352  000001           $EOPCT: .WORD   1
  (1)  024354  003022                           BGT     $DOAGN          ;;YES
  (1)  024356  012737                           MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
  (1)  024360  000001           $ENDCT: .WORD   1
  (1)  024362  024352                           $EOPCT
  (1)  024364  104401  024431                   TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
  (2)  024370  013746  001210                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
  (2)  024374  104405                           TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
  (1)  024376  104401  024426                   TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
  (1)  024402  013700  000042           $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
  (1)  024406  001405                           BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
  (1)  024410  000005                           RESET                   ;;CLEAR THE WORLD
  (1)  024412  004710           $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
  (1)  024414  000240                           NOP                     ;;SAVE ROOM
  (1)  024416  000240                           NOP                     ;;FOR
  (1)  024420  000240                           NOP                     ;;ACT11
  (1)  024422                           $DOAGN:
  (1)  024422  000137                           JMP     @(PC)+          ;;RETURN
  (1)  024424  003244           $RTNAD: .WORD   RSTART
  (1)  024426     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
  (1)  024431     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
```

```
2678                                    ;;*****************************************************************
2679                                    ;THIS ROUTINE WILL STALL FOR THE NO OF COUNTS IN THE ADRS OF R5
2680                                    ;;*****************************************************************
2681  024446  012537  002206           DELAY:  MOV     (R5)+,DLYCNT    ;GET DELAY COUNT
2682  024452  005337  002206           1$:     DEC     DLYCNT          ;START COUNTING
2683  024456  001375                           BNE     1$              ;BR IF MORE COUNTS
2684  024460  000205                           RTS     R5              ;EXIT DELAY ROUTINE
2685
2686                                    ;;*****************************************************************
2687                                    ;THIS ROUTINE SETS UP JSR, POP & TEST INSTR FOR POP-RESTORE TESTS
2688                                    ;;*****************************************************************
2689  024462  012777  020040  155606   POPSET: MOV     #20040,@STKPT   ;SET TOP OF STK
2690  024470  012737  163000  032376           MOV     #163000,BUFFER+2        ;SET UP JSR REL INSTR
2691  024476  012737  166000  032402           MOV     #166000,BUFFER+6        ;SET UP POP-RESTORE INSTR
2692  024504  012437  032400                   MOV     (R4)+,BUFFER+4  ;SET UP INSTRUCTION UNDER TEST
2693  024510  000204                           RTS     R4              ;RETURN
2694
2695                                    ;;*****************************************************************
2696                                    ;THIS ROUTINE CONTROLS JSR, POP & TEST INSTR SEQUENCE FOR POP-RESTORE TESTS
2697                                    ;;*****************************************************************
2698  024512  012777  020040  155556   DO:     MOV     #20040,@STKPT   ;SET TOP OF STK
2699  024520  012777  032374  155516           MOV     #BUFFER,@DPC    ;START -SET BITS UNDER TEST
2700  024526  052777  020000  155542           BIS     #20000,@STKPT   ;FUMBLE
2701  024534  005277  155504                   INC     @DPC            ;ADVANCE TO JSR INSTR
2702  024540  004537  024446                   JSR     R5,DELAY        ;STALL FOR STACKING
2703  024544  000004                           BIT2                    ;COUNT TO 4
2704  024546  012777  032400  155470           MOV     #BUFFER+4,@DPC  ;START -RESET BITS UNDER TEST
2705  024554  052777  020000  155514           BIS     #20000,@STKPT   ;FUMBLE
2706  024562  005277  155456                   INC     @DPC            ;ADVANCE TO POP INSTR
2707  024566  004537  024446                   JSR     R5,DELAY        ;STALL FOR RESTORING
2708  024572  000004                           BIT2                    ;COUNT TO 4
2709  024574  000207                           RTS     PC              ;RETURN
2710
2711                                    ;;*****************************************************************
2712                                    ;THIS ROUTINE CONVERTS A KT11 PAGE ADRS TO A 18 BIT ADRS
2713                                    ;;*****************************************************************
2714  024576  005001                   DPCONV: CLR     R1              ;CLR HI ORDER ADRS
2715  024600  006300                           ASL     R0              ;START SHIFTING LEFT
2716  024602  006300                           ASL     R0              ;
2717  024604  006300                           ASL     R0              ;
2718  024606  006300                           ASL     R0              ;
2719  024610  100002                   DPCON1: BPL     1$              ;BR IF ADRS BIT 17 IS CLR
2720  024612  012701  000002                   MOV     #2,R1           ;SET ADRS BIT 17 INDICATION IN BIT 1 - R1
2721  024616  006300                   1$:     ASL     R0              ;LOOK AT ADRS BIT 16
2722  024620  100001                           BPL     2$              ;BR IF ADRS BIT 16 IS CLR
2723  024622  005201                           INC     R1              ;SET ADRS BIT 16 INDICATION IN BIT 0 - R1
2724  024624  006300                   2$:     ASL     R0              ;COMPLETE LEFT JUSTIFICATION
2725  024626  000207                           RTS     PC              ;RETURN
2726
```

```
2728                                    ;;*****************************************************************
2729                                    ;THIS CODE REPORTS ANY KT11 ERROR AND HALTS
2730                                    ;;*****************************************************************
2731  024630  042737  001400  177572  KTSER:  BIC     #1400,@#KTSR0   ;DISABLE KT11
2732  024636  104401  027511                  TYPE    ,MSG2           ;GO REPORT KT11 THAT A ERROR TRAP OCCURRED
2733  024642  000000                  1$:     HALT                    ;KT11 ERROR TRAP - RUN KT11 DIAG OR
2734  024644  000776                          BR      1$              ;RESTART
2735
2736                                    ;;*****************************************************************
2737                                    ;THIS CODE RESETS ALL VS60 VECTORS WITH HALTS
2738                                    ;;*****************************************************************
2739  024646  012737  000340  177776  RSTVEC: MOV     #340,PSW        ;RESET PRIORITY TO HIGHEST LEVEL
2740  024654  012701  000004                  MOV     #4,R1           ;SET UP VECTOR LOCATION COUNT
2741  024660  013700  001252                  MOV     $VECT1,R0       ;GET 1ST VS60 VECTOR ADRS
2742  024664  042700  160000                  BIC     #160000,R0      ; CLR PSW BITS          ** C **
2743  024670  010010                  1$:     MOV     R0,(R0)         ;SET UP ADRS TO HALT
2744  024672  062720  000002                  ADD     #2,(R0)+        ;POINT IT TO HALT
2745  024676  005020                          CLR     (R0)+           ;SET UP HALT
2746  024700  005301                          DEC     R1              ;COUNT THIS VECTOR RESTORE
2747  024702  001372                          BNE     1$              ;BR IF MORE TO RESTORE
2748  024704  000207                          RTS     PC              ;RETURN IF ALL DONE
2749
2750                                    ;;*****************************************************************
2751                                    ;THIS ROUTINE MAKES UP A 12 BIT TANGENT FROM THE X OR Y POS
2752                                    ;REGS AND THE X OR Y DYNAMIC OFFSET REGS (HI ORDER 2 BITS) AND
2753                                    ;LEAVES THE RESULT IN $BDDAT
2754                                    ;;*****************************************************************
2755  024706  042737  176000  001126  TANCON: BIC     #176000,$BDDAT  ;SAVE LOW ORDER BITS ONLY
2756  024714  042705  147777                  BIC     #147777,R5      ;ONLY WANT BITS 10 & 11
2757  024720  006205                          ASR     R5              ;ROTATE 2 RIGHT
2758  024722  006205                          ASR     R5              ;
2759  024724  060537  001126                  ADD     R5,$BDDAT       ;MAKE UP 12 BIT TANGENT
2760  024730  000207                          RTS     PC              ;RETURN
2761
```

```
                        ;;*****************************************************************
                        ;THIS ROUTINE UPDATES THE TANGENT IN & GDDAT FOR EVERY INCREASE
                        ; IN VECTOR OF X OR Y
                        ;;*****************************************************************
024732  001010           CALTAN: BNE     1$              ;BR AND SET UP TANGENT IF VECTOR NON-ZERO
024734  012703  000400           MOV     #400,R3         ;R3 COUNTS EVERY 256+4 ADDS TO TANGENT
024740  012737  000001  001124   MOV     #1,$GDDAT       ;TANGENT WILL BE 1 WHEN VECTOR LENGTH 0
024746  005037  025042           CLR     TANGNT          ;TANGENT HOLDING LOC
024752  000432                   BR      3$              ;RETURN WITH ZERO TANGENT
024754  013737  025042  001124   1$:     MOV     TANGNT,$GDDAT   ;GET LAST VALUE
024762  062737  000005  001124           ADD     #5,$GDDAT       ;ADD 4 AND 1 FOR FRACTIONAL ROUNDING IN HARDWARE
024770  062737  000004  025042           ADD     #4,TANGNT       ;ADD 4 ON EVERY COUNT OF VECTOR LENGTH
024776  005303                   DEC     R3              ;ROUND UP EVERY 11.25 DEGREES
025000  001010                   BNE     2$              ;BR IF NOT REQUIRED
025002  062737  000001  001124   ADD     #1,$GDDAT       ;ROUND UP EVERY 256
025010  062737  000001  025042   ADD     #1,TANGNT       ;SAME
025016  012703  000400           MOV     #400,R3         ;RESET 256 COUNT
025022  032737  010000  001124   2$:     BIT     #10000,$GDDAT   ;TEST TANGENT FOR GREATER THAN 12 BITS
025030  001403                   BEQ     3$              ;BR IF NOT
025032  012737  007777  001124   MOV     #7777,$GDDAT    ;CAN'T GET GREATER THAN 7777
025040  000207           3$:     RTS     PC              ;RETURN WITH TANGENT IN $GDDAT

025042  000000           TANGNT: 0                       ;LOC MAINTAINS TANGENT MINUS 1

                        ;;*****************************************************************
                        ;THIS ROUTINE STARTS THE DISPLAY WITH MAINT. SW 2, THEN SETS
                        ;MAINT. SW 3, AND THEN ISSUES A # OF RESUMES SPECIFIED
                        ;BY R5 AT WHICH TIME IT WILL EXIT
                        ;;*****************************************************************
025044                  EXECUTE:
025044  012537  025114           MOV     (R5)+,CNTR      ;SET UP THE RESUME COUNTER
025050  012777  020040  155220   MOV     #20040,@STKPT   ;SET MAINT SW 2 AND TOP OF STACK
025056  012777  032374  155160   MOV     #BUFFER,@DPC    ;START DISPLAY
025064  052777  040000  155204   1$:     BIS     #40000,@STKPT   ;SET MAINT SW 3
025072  052777  040000  155176           BIS     #40000,@STKPT   ;AGAIN
025100  005277  155140           INC     @DPC            ;RESUME
025104  005337  025114           DEC     CNTR            ;COUNT RESUME
025110  001365                   BNE     1$              ;BR IF MORE RESUMES TO FOLLOW
025112  000205                   RTS     R5              ;EXIT

025114  000000           CNTR:   0                       ;CNTR USED ABOVE
```

```
  2806                                  .SBTTL  SCOPE HANDLER ROUTINE
   (1)
   (2)                                  ;;**********************************************************************
   (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)
   (1)  025116                          $SCOPE:
   (1)  025116  032777  040000  154014  1$:     BIT     #BIT14,@SWR       ;;LOOP ON PRESENT TEST?
   (1)  025124  001114                          BNE     $OVER             ;;YES IF SW14=1
   (1)                                  ;#####START OF CODE FOR THE XOR TESTER#####
   (1)  025126  000416                  $XTSTR: BR      6$                ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
   (1)                                                                    ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
   (1)  025130  013746  000004                  MOV     @#ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
   (1)  025134  012737  025154  000004          MOV     #5$,@#ERRVEC      ;;SET FOR TIMEOUT
   (1)  025142  005737  177060                  TST     @#177060          ;;TIME OUT ON XOR?
   (1)  025146  012637  000004                  MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
   (1)  025152  000463                          BR      $SVLAD            ;;GO TO THE NEXT TEST
   (1)  025154  022626                  5$:     CMP     (SP)+,(SP)+       ;;CLEAR THE STACK AFTER A TIME OUT
   (1)  025156  012637  000004                  MOV     (SP)+,@#ERRVEC    ;;RESTORE THE ERROR VECTOR
   (1)  025162  000423                          BR      7$                ;;LOOP ON THE PRESENT TEST
   (1)  025164                          6$:;#####END OF CODE FOR THE XOR TESTER#####
   (1)  025164  032777  000400  153746          BIT     #BIT08,@SWR       ;;LOOP ON SPEC. TEST?
   (1)  025172  001404                          BEQ     2$                ;;BR IF NO
   (1)  025174  127737  153740  001102          CMPB    @SWR,$TSTNM       ;;ON THE RIGHT TEST?    SWR<7:0>
   (1)  025202  001465                          BEQ     $OVER             ;;BR IF YES
   (1)  025204  105737  001103          2$:     TSTB    $ERFLG            ;;HAS AN ERROR OCCURRED?
   (1)  025210  001421                          BEQ     3$                ;;BR IF NO
   (1)  025212  123737  001115  001103          CMPB    $ERMAX,$ERFLG     ;;MAX. ERRORS FOR THIS TEST OCCURRED?
   (1)  025220  101015                          BHI     3$                ;;BR IF NO
   (1)  025222  032777  001000  153710          BIT     #BIT09,@SWR       ;;LOOP ON ERROR?
   (1)  025230  001404                          BEQ     4$                ;;BR IF NO
   (1)  025232  013737  001110  001106  7$:     MOV     $LPERR,$LPADR     ;;SET LOOP ADDRESS TO LAST SCOPE
   (1)  025240  000446                          BR      $OVER
   (1)  025242  105037  001103          4$:     CLRB    $ERFLG            ;;ZERO THE ERROR FLAG
   (1)  025246  005037  001166                  CLR     $TIMES            ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
   (1)  025252  000415                          BR      1$                ;;ESCAPE TO THE NEXT TEST
   (1)  025254  032777  004000  153656  3$:     BIT     #BIT11,@SWR       ;;INHIBIT ITERATIONS?
   (1)  025262  001011                          BNE     1$                ;;BR IF YES
   (1)  025264  005737  001210                  TST     $PASS             ;;IF FIRST PASS OF PROGRAM
   (1)  025270  001406                          BEQ     1$                ;;       INHIBIT ITERATIONS
   (1)  025272  005237  001104                  INC     $ICNT             ;;INCREMENT ITERATION COUNT
   (1)  025276  023737  001166  001104          CMP     $TIMES,$ICNT      ;;CHECK THE NUMBER OF ITERATIONS MADE
   (1)  025304  002024                          BGE     $OVER             ;;BR IF MORE ITERATION REQUIRED
   (1)  025306  012737  000001  001104  1$:     MOV     #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
   (1)  025314  013737  025372  001166          MOV     $MXCNT,$TIMES     ;;SET NUMBER OF ITERATIONS TO DO
   (1)  025322  105237  001102          $SVLAD: INCB    $TSTNM            ;;COUNT TEST NUMBERS
   (1)  025326  113737  001102  001206          MOVB    $TSTNM,$TESTN     ;;SET TEST NUMBER IN APT MAILBOX
   (1)  025334  011637  001106                  MOV     (SP),$LPADR       ;;SAVE SCOPE LOOP ADDRESS
```

```
(1)  025340  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  025344  005037  001170                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  025350  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  025356  013777  001102  153556  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  025364  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(1)  025370  000002                          RTI                     ;;FIXES PS
(1)  025372  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
```

```
 2808                                   .SBTTL  APT COMMUNICATIONS ROUTINE
  (1)
  (2)                                   ;;*****************************************************************
  (1)  025374  112737  000001  025640  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
  (1)  025402  112737  000001  025636  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
  (1)  025410  000403                          BR      $ATYC
  (1)  025412  112737  000001  025640  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
  (1)  025420                          $ATYC:
  (3)  025420  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (3)  025422  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (1)  025424  105737  025636                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
  (1)  025430  001450                          BEQ     5$              ;;IF NOT:  BR
  (1)  025432  122737  000001  001222          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
  (1)  025440  001031                          BNE     3$              ;;IF NOT:  BR
  (1)  025442  132737  000100  001223          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
  (1)  025450  001425                          BEQ     3$              ;;IF NOT:  BR
  (1)  025452  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
  (1)  025456  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
  (1)  025464  005737  001202          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
  (1)  025470  001375                          BNE     1$              ;;IF NOT:  WAIT
  (1)  025472  010037  001216                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
  (1)  025476  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
  (1)  025500  001376                          BNE     2$
  (1)  025502  163700  001216                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
  (1)  025506  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
  (1)  025510  010037  001220                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
  (1)  025514  012737  000004  001202          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
  (1)  025522  000413                          BR      5$
  (1)  025524  017637  000004  025550  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
  (1)  025532  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
  (3)  025540  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
  (1)  025544  004737  026200                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
  (1)  025550  000000                  4$:     .WORD   0
  (1)  025552                          5$:
  (1)  025552  105737  025640          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
  (1)  025556  001416                          BEQ     12$             ;;IF NOT:  BR
  (1)  025560  005737  001222                  TST     $ENV            ;;RUNNING UNDER APT?
  (1)  025564  001413                          BEQ     12$             ;;IF NOT:  BR
  (1)  025566  005737  001202          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
  (1)  025572  001375                          BNE     11$             ;;IF NOT:  WAIT
  (1)  025574  017637  000004  001204          MOV     @4(SP),$FATAL   ;;GET ERROR #
  (1)  025602  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
  (1)  025610  005237  001202                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
  (1)  025614  105037  025640          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
  (1)  025620  105037  025637                  CLRB    $LFLG           ;;CLEAR LOG FLAG
  (1)  025624  105037  025636                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
  (3)  025630  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
  (3)  025632  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  (1)  025634  000207                          RTS     PC              ;;RETURN
  (1)  025636     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
  (1)  025637     000          $LFLG:  .BYTE   0               ;;LOG FLAG
  (1)  025640     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
  (1)          025642                  .EVEN
  (1)          000200          APTSIZE=200
  (1)          000001          APTENV=001
  (1)          000100          APTSPOOL=100
```

```
  (1)           000040                      APTCSUP=040
 2809                                       .SBTTL  ERROR HANDLER ROUTINE
  (1)
  (2)                                       ;;**********************************************************************
  (1)                                       ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
  (1)                                       ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
  (1)                                       ;*AND GO TO $ERRTYP ON ERROR
  (1)                                       ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
  (1)                                       ;*SW15=1          HALT ON ERROR
  (1)                                       ;*SW13=1          INHIBIT ERROR TYPEOUTS
  (1)                                       ;*SW10=1          BELL ON ERROR
  (1)                                       ;*SW09=1          LOOP ON ERROR
  (1)                                       ;*CALL
  (1)                                       ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
  (1)
  (1)   025642                              $ERROR:
  (2)   025642  113737  001102  002204              MOVB $TSTNM,TSTNUM
  (1)   025650  105237  001103              7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
  (1)   025654  001775                              BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
  (1)   025656  013777  001102  153256              MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
  (1)   025664  032777  002000  153246              BIT     #BIT10,@SWR     ;;BELL ON ERROR?
  (1)   025672  001402                              BEQ     1$              ;;NO - SKIP
  (1)   025674  104401  001172                      TYPE    ,$BELL          ;;RING BELL
  (1)   025700  005237  001112              1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
  (1)   025704  011637  001116                      MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
  (1)   025710  162737  000002  001116              SUB     #2,$ERRPC
  (1)   025716  117737  153174  001114              MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
  (1)   025724  032777  020000  153206              BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
  (1)   025732  001004                              BNE     20$             ;;SKIP TYPEOUTS
  (1)   025734  004737  026044                      JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
  (1)   025740  104401  001177                      TYPE    ,$CRLF
  (1)   025744                              20$:
  (1)   025744  122737  000001  001222              CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)   025752  001007                              BNE     2$              ;;NO,SKIP APT ERROR REPORT
  (1)   025754  113737  001114  025766              MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
  (1)   025762  004737  025412                      JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
  (1)   025766     000                    21$:     .BYTE   0
  (1)   025767     000                             .BYTE   0
  (1)   025770  000777                      22$:    BR      22$             ;;APT ERROR LOOP
  (1)   025772  005777  153142              2$:     TST     @SWR            ;;HALT ON ERROR
  (1)   025776  100001                              BPL     3$              ;;SKIP IF CONTINUE
  (1)   026000  000000                              HALT                    ;;HALT ON ERROR!
  (1)   026002  032777  001000  153130      3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
  (1)   026010  001402                              BEQ     4$              ;;BR IF NO
  (1)   026012  013716  001110                      MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
  (1)   026016  005737  001170              4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
  (1)   026022  001402                              BEQ     5$              ;;BR IF NONE
  (1)   026024  013716  001170                      MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
  (1)   026030                              5$:
  (1)   026030  022737  024412  000042              CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
  (1)   026036  001001                              BNE     6$              ;;BRANCH IF NO
  (1)   026040  000000                              HALT                    ;;YES
  (1)   026042                              6$:
  (1)   026042  000002                              RTI                     ;;RETURN
```

```
2811                                    .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1)                                     ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1)                                     ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  026044                             $ERRTYP:
(1)  026044  104401  001177                     TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  026050  010046                             MOV     R0,-(SP)        ;;SAVE R0
(1)  026052  005000                             CLR     R0              ;;PICKUP THE ITEM INDEX
(1)  026054  153700  001114                     BISB    @#$ITEMB,R0
(1)  026060  001004                             BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                                     ;;TYPE THE PC OF THE ERROR
(2)  026062  013746  001116                     MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                                     ;;ERROR ADDRESS
(2)  026066  104402                             TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  026070  000426                             BR      6$              ;;GET OUT
(1)  026072  005300                     1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1)  026074  006300                             ASL     R0              ;;      WORK FOR THE ERROR TABLE
(1)  026076  006300                             ASL     R0
(1)  026100  006300                             ASL     R0
(1)  026102  062700  001264                     ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(1)  026106  012037  026116                     MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
(1)  026112  001404                             BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1)  026114  104401                             TYPE                    ;;TYPE THE "ERROR MESSAGE"
(1)  026116  000000                     2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
(1)  026120  104401  001177                     TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  026124  012037  026134             3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
(1)  026130  001404                             BEQ     5$              ;;SKIP TYPEOUT IF 0
(1)  026132  104401                             TYPE                    ;;TYPE THE "DATA HEADER"
(1)  026134  000000                     4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
(1)  026136  104401  001177                     TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  026142  011000                     5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
(1)  026144  001004                             BNE     7$              ;;GO TYPE THE DATA
(1)  026146  012600                     6$:     MOV     (SP)+,R0        ;;RESTORE R0
(1)  026150  104401  001177                     TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  026154  000207                             RTS     PC              ;;RETURN
(1)  026156                             7$:
(2)  026156  013046                             MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
(2)  026160  104402                             TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  026162  005710                             TST     (R0)            ;;IS THERE ANOTHER NUMBER?
(1)  026164  001770                             BEQ     6$              ;;BR IF NO
(1)  026166  104401  026174                     TYPE    ,8$             ;;TYPE TWO(2) SPACES
(1)  026172  000771                             BR      7$              ;;LOOP
(1)  026174  020040      000           8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
(1)          026200                             .EVEN
```

```
 2813                                   .SBTTL  TYPE ROUTINE
  (1)
  (2)                                   ;;*********************************************************************
  (1)                                   ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
  (1)                                   ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
  (1)                                   ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
  (1)                                   ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
  (1)                                   ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
  (1)                                   ;*
  (1)                                   ;*CALL:
  (1)                                   ;*1) USING A TRAP INSTRUCTION
  (1)                                   ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
  (1)                                   ;*OR
  (1)                                   ;*      TYPE
  (1)                                   ;*      MESADR
  (1)                                   ;*
  (1)
  (1)  026200  105737  001157           $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
  (1)  026204  100002                           BPL     1$              ;;BR IF YES
  (1)  026206  000000                           HALT                    ;;HALT HERE IF NO TERMINAL
  (1)  026210  000430                           BR      3$              ;;LEAVE
  (1)  026212  010046                   1$:     MOV     R0,-(SP)        ;;SAVE R0
  (1)  026214  017600  000002                   MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
  (1)  026220  122737  000001  001222           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)  026226  001011                           BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
  (1)  026230  132737  000100  001223           BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
  (1)  026236  001405                           BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
  (1)  026240  010037  026250                   MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
  (1)  026244  004737  025402                   JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
  (1)  026250  000000                   61$:    .WORD   0               ;;MESSAGE ADDRESS
  (1)  026252  132737  000040  001223   62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
  (1)  026260  001003                           BNE     60$             ;;YES,SKIP TYPE OUT
  (1)  026262  112046                   2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
  (1)  026264  001005                           BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
  (1)  026266  005726                           TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
  (1)  026270  012600                   60$:    MOV     (SP)+,R0        ;;RESTORE R0
  (1)  026272  062716  000002           3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
  (1)  026276  000002                           RTI                     ;;RETURN
  (1)  026300  122716  000011           4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
  (1)  026304  001430                           BEQ     8$
  (1)  026306  122716  000200                   CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
  (1)  026312  001006                           BNE     5$
  (1)  026314  005726                           TST     (SP)+           ;;POP  <CR><LF> EQUIV
  (1)  026316  104401                           TYPE                    ;;TYPE A CR AND LF
  (1)  026320  001177                           $CRLF
  (1)  026322  105037  026456                   CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
  (1)  026326  000755                           BR      2$              ;;GET NEXT CHARACTER
  (1)  026330  004737  026412           5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
  (1)  026334  123726  001156           6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
  (1)  026340  001350                           BNE     2$              ;;IF NO GO GET NEXT CHAR.
  (1)  026342  013746  001154                   MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
  (1)                                           ;;AND THE NULL CHAR.
  (1)  026346  105366  000001           7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
  (1)  026352  002770                           BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
  (1)  026354  004737  026412                   JSR     PC,$TYPEC       ;;GO TYPE A NULL
  (1)  026360  105337  026456                   DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
```

```
(1)  026364  000770                          BR      7$              ;;LOOP
(1)
(1)                                          ;HORIZONTAL TAB PROCESSOR
(1)
(1)  026366  112716  000040                 8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  026372  004737  026412                 9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  026376  132737  000007  026456                 BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  026404  001372                                 BNE     9$              ;;TAB STOP
(1)  026406  005726                                 TST     (SP)+           ;;POP SPACE OFF STACK
(1)  026410  000724                                 BR      2$              ;;GET NEXT CHARACTER
(1)  026412  105777  152532                 $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  026416  100375                                 BPL     $TYPEC
(1)  026420  116677  000002  152524                 MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  026426  122766  000015  000002                 CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  026434  001003                                 BNE     1$              ;;BRANCH IF NO
(1)  026436  105037  026456                         CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  026442  000406                                 BR      $TYPEX          ;;EXIT
(1)  026444  122766  000012  000002         1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  026452  001402                                 BEQ     $TYPEX          ;;BRANCH IF YES
(1)  026454  105227                                 INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  026456  000000                 $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  026460  000207                 $TYPEX: RTS     PC
(1)
```

```
 2815                                   .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)                                   ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
  (1)                                   ;*CALL:
  (1)                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                   ;*      TYPOS                   ;;CALL FOR TYPEOUT
  (1)                                   ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
  (1)                                   ;*      .BYTE   M               ;;M=1 OR 0
  (1)                                   ;*                                      ;;1=TYPE LEADING ZEROS
  (1)                                   ;*                                      ;;0=SUPPRESS LEADING ZEROS
  (1)                                   ;*
  (1)                                   ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
  (1)                                   ;*$TYPOS OR $TYPOC
  (1)                                   ;*CALL:
  (1)                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                   ;*      TYPON                   ;;CALL FOR TYPEOUT
  (1)                                   ;*
  (1)                                   ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
  (1)                                   ;*CALL:
  (1)                                   ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                   ;*      TYPOC                   ;;CALL FOR TYPEOUT
  (1)
  (1)  026462  017646  000000           $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
  (1)  026466  116637  000001  026705           MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
  (1)  026474  112637  026707                   MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
  (1)  026500  062716  000002                   ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
  (1)  026504  000406                           BR      $TYPON
  (1)  026506  112737  000001  026705   $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
  (1)  026514  112737  000006  026707           MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
  (1)  026522  112737  000005  026704   $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
  (1)  026530  010346                           MOV     R3,-(SP)        ;;SAVE R3
  (1)  026532  010446                           MOV     R4,-(SP)        ;;SAVE R4
  (1)  026534  010546                           MOV     R5,-(SP)        ;;SAVE R5
  (1)  026536  113704  026707                   MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
  (1)  026542  005404                           NEG     R4
  (1)  026544  062704  000006                   ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
  (1)  026550  110437  026706                   MOVB    R4,$OMODE       ;;SAVE IT FOR USE
  (1)  026554  113704  026705                   MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
  (1)  026560  016605  000012                   MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
  (1)  026564  005003                           CLR     R3              ;;CLEAR THE OUTPUT WORD
  (1)  026566  006105                   1$:     ROL     R5              ;;ROTATE MSB INTO ''C''
  (1)  026570  000404                           BR      3$              ;;GO DO MSB
  (1)  026572  006105                   2$:     ROL     R5              ;;FORM THIS DIGIT
  (1)  026574  006105                           ROL     R5
  (1)  026576  006105                           ROL     R5
  (1)  026600  010503                           MOV     R5,R3
  (1)  026602  006103                   3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
  (1)  026604  105337  026706                   DECB    $OMODE          ;;TYPE THIS DIGIT?
  (1)  026610  100016                           BPL     7$              ;;BR IF NO
  (1)  026612  042703  177770                   BIC     #177770,R3      ;;GET RID OF JUNK
  (1)  026616  001002                           BNE     4$              ;;TEST FOR 0
  (1)  026620  005704                           TST     R4              ;;SUPPRESS THIS 0?
  (1)  026622  001403                           BEQ     5$              ;;BR IF YES
```

```
(1)  026624  005204                    4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  026626  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  026632  052703  000040            5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  026636  110337  026702                    MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  026642  104401  026702                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  026646  105337  026704            7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  026652  003347                            BGT     2$              ;;BR IF MORE TO DO
(1)  026654  002402                            BLT     6$              ;;BR IF DONE
(1)  026656  005204                            INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  026660  000744                            BR      2$              ;;GO DO THE LAST DIGIT
(1)  026662  012605                    6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  026664  012604                            MOV     (SP)+,R4        ;;RESTORE R4
(1)  026666  012603                            MOV     (SP)+,R3        ;;RESTORE R3
(1)  026670  016666  000002  000004            MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  026676  012616                            MOV     (SP)+,(SP)
(1)  026700  000002                            RTI                     ;;RETURN
(1)  026702     000                    8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  026703     000                            .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  026704     000                    $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  026705     000                    $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  026706  000000                    $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
2817                                    .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 (1)                                    ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 (1)                                    ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 (1)                                    ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 (1)                                    ;*REPLACED WITH SPACES.
 (1)                                    ;*CALL:
 (1)                                    ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
 (1)                                    ;*      TYPDS                   ;;GO TO THE ROUTINE
 (1)
 (1)  026710                            $TYPDS:
 (3)  026710  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 (3)  026712  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 (3)  026714  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 (3)  026716  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 (3)  026720  010546                            MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 (1)  026722  012746  020200                    MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 (1)  026726  016605  000020                    MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 (1)  026732  100004                            BPL     1$              ;;BR IF INPUT IS POS.
 (1)  026734  005405                            NEG     R5              ;;MAKE THE BINARY NUMBER POS.
 (1)  026736  112766  000055  000001            MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
 (1)  026744  005000                    1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
 (1)  026746  012703  027124                    MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
 (1)  026752  112723  000040                    MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
 (1)  026756  005002                    2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
 (1)  026760  016001  027114                    MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
 (1)  026764  160105                    3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
 (1)  026766  002402                            BLT     4$              ;;BR IF DONE
 (1)  026770  005202                            INC     R2              ;;INCREASE THE BCD DIGIT BY 1
 (1)  026772  000774                            BR      3$
 (1)  026774  060105                    4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
 (1)  026776  005702                            TST     R2              ;;CHECK IF BCD DIGIT=0
 (1)  027000  001002                            BNE     5$              ;;FALL THROUGH IF 0
 (1)  027002  105716                            TSTB    (SP)            ;;STILL DOING LEADING 0'S?
 (1)  027004  100407                            BMI     7$              ;;BR IF YES
 (1)  027006  106316                    5$:     ASLB    (SP)            ;;MSD?
 (1)  027010  103003                            BCC     6$              ;;BR IF NO
 (1)  027012  116663  000001  177777            MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
 (1)  027020  052702  000060            6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
 (1)  027024  052702  000040            7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
 (1)  027030  110223                            MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
 (1)  027032  005720                            TST     (R0)+           ;;JUST INCREMENTING
 (1)  027034  020027  000010                    CMP     R0,#10          ;;CHECK THE TABLE INDEX
 (1)  027040  002746                            BLT     2$              ;;GO DO THE NEXT DIGIT
 (1)  027042  003002                            BGT     8$              ;;GO TO EXIT
 (1)  027044  010502                            MOV     R5,R2           ;;GET THE LSD
 (1)  027046  000764                            BR      6$              ;;GO CHANGE TO ASCII
 (1)  027050  105726                    8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
 (1)  027052  100003                            BPL     9$              ;;BR IF NO
 (1)  027054  116663  177777  177776            MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
 (1)  027062  105013                    9$:     CLRB    (R3)            ;;SET THE TERMINATOR
 (3)  027064  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
 (3)  027066  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
 (3)  027070  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
```

```
(3)  027072  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  027074  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  027076  104401  027124                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
(1)  027102  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  027110  012616                          MOV     (SP)+,(SP)
(1)  027112  000002                          RTI                     ;;RETURN TO USER
(1)  027114  023420                $DTBL:    10000.
(1)  027116  001750                          1000.
(1)  027120  000144                          100.
(1)  027122  000012                          10.
(1)  027124  000004                $DBLK:    .BLKW   4
```

```
 2819                                   .SBTTL  TRAP DECODER
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*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)
  (1)  027134  010046                   $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
  (1)  027136  016600  000002                   MOV     2(SP),R0        ;;GET TRAP ADDRESS
  (1)  027142  005740                           TST     -(R0)           ;;BACKUP BY 2
  (1)  027144  111000                           MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
  (1)  027146  006300                           ASL     R0              ;;POSITION FOR INDEXING
  (1)  027150  016000  027170                   MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
  (1)  027154  000200                           RTS     R0              ;;GO TO ROUTINE
  (1)
  (1)
  (1)                                   ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
  (1)
  (1)  027156  011646                   $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
  (1)  027160  016666  000004  000002           MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
  (1)  027166  000002                           RTI                     ;;RESTORE THE PSW
  (1)
  (3)                                   .SBTTL  TRAP TABLE
  (3)
  (3)                                   ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
  (3)                                   ;*BY THE ''TRAP'' INSTRUCTION.
  (3)
  (3)                                   ;       ROUTINE
  (3)                                   ;       -------
  (3)  027170  027156                   $TRPAD: .WORD   $TRAP2
  (3)  027172  026200                           $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
  (3)  027174  026506                           $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
  (3)  027176  026462                           $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
  (3)  027200  026522                           $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
  (3)  027202  026710                           $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
  (1)
  (1)
```

```
 2821                                   .SBTTL  POWER DOWN AND UP ROUTINES
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;POWER DOWN ROUTINE
  (1)  027204  012737  027350  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
  (1)  027212  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (3)  027220  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (3)  027222  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (3)  027224  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (3)  027226  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (3)  027230  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
  (3)  027232  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
  (3)  027234  017746  151700                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
  (1)  027240  010637  027354                  MOV     SP,$SAVR6       ;;SAVE SP
  (1)  027244  012737  027256  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
  (1)  027252  000000                          HALT
  (1)  027254  000776                          BR      .-2             ;;HANG UP
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;POWER UP ROUTINE
  (1)  027256  012737  027350  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
  (1)  027264  013706  027354                  MOV     $SAVR6,SP       ;;GET SP
  (1)  027270  005037  027354                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
  (1)  027274  005237  027354          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
  (1)  027300  001375                          BNE     1$              ;;OF  WORD
  (3)  027302  012677  151632                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
  (3)  027306  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
  (3)  027310  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
  (3)  027312  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
  (3)  027314  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
  (3)  027316  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
  (3)  027320  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
  (1)  027322  012737  027204  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
  (1)  027330  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (1)  027336  104401                          TYPE                    ;;REPORT THE POWER FAILURE
  (1)  027340  027356                  $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
  (1)  027342  012716                          MOV     (PC)+,(SP)      ;;RESTART AT RSTART
  (1)  027344  003244                  $PWRAD: .WORD   RSTART          ;;RESTART ADDRESS
  (1)  027346  000002                          RTI
  (1)  027350  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
  (1)  027352  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
  (1)  027354  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
 2822  027356  005015  042522  052123          PWRMSG: .ASCIZ  <15><12>/RESTARTED AFTER A POWER FAILURE/<15><12><12>
 2823
```

```
                                                .SBTTL  ASCII MESSAGES

027423     015  051412  040524  MSG1:   .ASCIZ  <15><12>/START OF CZVSC-C   VS60 INSTRUCTION TEST PART III/<15><12>
027511     015  045412  030524  MSG2:   .ASCIZ  <15><12>/KT11 ER TRAP/
027530  005015  042515  047515  MSG3:   .ASCIZ  <15><12>/MEMORY SIZE =/
027550  006513  000012          MSG4:   .ASCIZ  /K/<15><12>
027554  027114  027120  043040  EM1:    .ASCIZ  /L.P. FLAG INTR FAILURE/
027603     123  046111  020117  EM2:    .ASCIZ  /SILO FAILURE AT LEVEL INDICATED/
027643     123  045524  050040  EM3:    .ASCIZ  /STK PTR DEC ER/
027662  052123  041501  020113  EM4:    .ASCIZ  /STACK OVERFLOW ER/
027704  050104  020103  052123  EM5:    .ASCIZ  /DPC STKING ER/
027722  050104  020125  040516  EM6:    .ASCIZ  /DPU NAME STKING ER/
027745     115  042117  020105  EM7:    .ASCIZ  /MODE STKING ER/
027764  042526  052103  051117  EM10:   .ASCIZ  /VECTOR SCALE STKING ER/
030013     103  040510  040522  EM11:   .ASCIZ  /CHARACTER SCALE STKING ER/
030045     103  040510  040522  EM12:   .ASCIZ  /CHARACTER ROTATE STKING ER/
030100  047111  042524  051516  EM13:   .ASCIZ  /INTENSITY LEVEL STKING ER/
030132  047503  047514  020122  EM14:   .ASCIZ  /COLOR LEVEL STKING ER/
030160  052111  046101  041511  EM15:   .ASCIZ  /ITALICS STKING ER/
030202  042515  052516  051440  EM16:   .ASCIZ  /MENU STKING ER/
030221     102  044514  045516  EM17:   .ASCIZ  /BLINK ENA STKING ER/
030245     123  047524  020120  EM20:   .ASCIZ  /STOP INT ENA STKING ER/
030274  050114  044040  052111  EM21:   .ASCIZ  /LP HIT DISABLE STKING ER/
030325     114  020104  052123  EM22:   .ASCIZ  /LD STAUTS ENA STKING ER/
030355     105  043504  020105  EM23:   .ASCIZ  /EDGE INT ENA STKING ER/
030404  044103  051101  051440  EM24:   .ASCIZ  /CHAR STRING ESCAPE STKING ER/
030441     104  050105  044124  EM25:   .ASCIZ  /DEPTH QUE STKING ER/
030465     114  047111  020105  EM26:   .ASCIZ  /LINE TYPE STKING ER/
030511     111  052116  047105  EM27:   .ASCIZ  /INTENSITY ENA STKING ER/
030541     114  050056  020056  EM30:   .ASCIZ  /L.P. INTR ENA STKING ER/
030571     114  050056  020056  EM31:   .ASCIZ  /L.P. SW INTR ENA STKING ER/
030624  044123  043111  020124  EM32:   .ASCIZ  /SHIFT OUT STKING ER/
030650  052123  020113  052120  EM33:   .ASCIZ  /STK PTR INC ER/
030667     123  045524  052440  EM34:   .ASCIZ  /STK UNDERFLOW ER/
030710  050104  020103  047520  EM35:   .ASCIZ  /DPC POP ER/
030723     116  046501  020105  EM36:   .ASCIZ  /NAME POP ER/
030737     126  041505  047524  EM37:   .ASCIZ  /VECTOR SCALE POP ER/
030763     103  040510  020122  EM40:   .ASCIZ  /CHAR SCALE POP ER/
031005     103  040510  020122  EM41:   .ASCIZ  /CHAR ROTATE POP ER/
031030  047503  047514  020122  EM42:   .ASCIZ  /COLOR LEVEL POP ER/
031053     111  040524  044514  EM43:   .ASCIZ  /ITALICS POP ER/
031072  042515  052516  050040  EM44:   .ASCIZ  /MENU POP ER/
031106  047111  042524  051516  EM45:   .ASCIZ  /INTENSITY ENA POP ER/
031133     114  050056  020056  EM46:   .ASCIZ  /L.P. INTR ENA POP ER/
031160  027114  027120  051440  EM47:   .ASCIZ  /L.P. SW INTR ENA POP ER/
031210  044514  042516  052040  EM50:   .ASCIZ  /LINE TYPE POP ER/
031231     111  052116  047105  EM51:   .ASCIZ  /INTENSITY LEVEL POP ER/
031260  046102  047111  020113  EM52:   .ASCIZ  /BLINK POP ER/
031275     115  042117  020105  EM53:   .ASCIZ  /MODE POP ER/
031311     123  047524  020120  EM54:   .ASCIZ  /STOP INTR ENA POP ER/
031336  042504  052120  020110  EM55:   .ASCIZ  /DEPTH QUE POP ER/
031357     105  043504  020105  EM56:   .ASCIZ  /EDGE INTR ENA POP ER/
031404  044103  051101  051440  EM57:   .ASCIZ  /CHAR STRING ESC POP ER/
031433     114  050056  020056  EM60:   .ASCIZ  /L.P. HIT DISABLE POP ER/
031463     123  044510  052106  EM61:   .ASCIZ  /SHIFT OUT POP ER/
031504  042524  046522  047111  EM62:   .ASCIZ  /TERMINATE CHAR POP ER/
```

```
 2881  031532  052123  020113  053117  EM63:   .ASCIZ  /STK OVFLO INT ER/
 2882  031553     123  045524  052440  EM64:   .ASCIZ  /STK UNFLO INT ER/
 2883  031574  042504  052114  020101  EM65:   .ASCIZ  /DELTA LENGTH ER/
 2884  031614  040524  043516  047105  EM66:   .ASCIZ  /TANGENT ER/
 2885  031627     104  041520  030440  EM67:   .ASCIZ  /DPC 16-17 ER/
 2886  031644  051526  030066  046440  EM70:   .ASCIZ  /VS60 MEM ADRS FAILURE/
 2887  031672  052123  050117  043040  EM71:   .ASCIZ  /STOP FLAG FAILED TO SET/
 2888  031722  052123  051101  020124  EM72:   .ASCIZ  /START FAILED TO SET 'TOP OF STK'/
 2889  031763     105  051122  041520  DH1:    .ASCIZ  /ERRPC   TSTNUM  BUSADRS EXPCT   RCVD/
 2890  032030  051105  050122  020103  DH2:    .ASCIZ  /ERRPC   TSTNUM  BUSADRS EXPCT   RCVD    STK SEL/
 2891  032110  051105  050122  020103  DH3:    .ASCIZ  /ERRPC   TSTNUM  GDPC-HI GDPC    BDPC-HI BDPC/
 2892  032165     105  051122  041520  DH4:    .ASCIZ  /ERRPC   TSTNUM  VECTOR  EXPCT   RCVD/
 2893  032232  051105  050122  020103  DH5:    .ASCIZ  /ERRPC   TSTNUM  BUSADRS EXPCT   RCVD    LEVEL/
 2894                                          .EVEN
 2895  032310  001116  002204  001122  DT1:    .WORD   $ERRPC,TSTNUM,$BDADR,$GDDAT,$BDDAT,0
 2896  032324  001116  002204  001122  DT2:    .WORD   $ERRPC,TSTNUM,$BDADR,$GDDAT,$BDDAT,$TMP0,0
 2897  032342  001116  002204  002224  DT3:    .WORD   $ERRPC,TSTNUM,G1716,$GDDAT,B1716,$BDDAT,0
 2898  032360  001116  002204  001164  DT4:    .WORD   $ERRPC,TSTNUM,$TMP0,$GDDAT,$BDDAT,0
 2899
 2900                                  ;;*****************************************************************
 2901                                          ;THIS IS THE WORKING AREA FOR ALL VS60 NPR'S (INSTR & DATA)
 2902                                          ;FROM HERE TO THE END OF MEMORY
 2903                                  ;;*****************************************************************
 2904  032374  000000                 BUFFER: 0                       ;LOCATIONS START HERE FOR TESTING
 2905          000001                          .END                    ;END OF PROGRAM
```

```
ABASE = 172000              17#     29
ACDW1 = 000000              29
ACDW2 = 000000              29
ACPUOP= 000000              29
ADDW0 = 000000              29
ADDW1 = 000000              29
ADDW10= 000000              29
ADDW11= 000000              29
ADDW12= 000000              29
ADDW13= 000000              29
ADDW14= 000000              29
ADDW15= 000000              29
ADDW2 = 000000              29
ADDW3 = 000000              29
ADDW4 = 000000              29
ADDW5 = 000000              29
ADDW6 = 000000              29
ADDW7 = 000000              29
ADDW8 = 000000              29
ADDW9 = 000000              29
ADEVCT= 000000              29
ADEVM = 000000              29
AENV  = 000000              29
AENVM = 000000              29
AFATAL= 000000              29
AMADR1= 000000              29
AMADR2= 000000              29
AMADR3= 000000              29
AMADR4= 000000              29
AMAMS1= 000000              29
AMAMS2= 000000              29
AMAMS3= 000000              29
AMAMS4= 000000              29
AMSGAD= 000000              29
AMSGLG= 000000              29
AMSGTY= 000000              29
AMTYP1= 000000              29
AMTYP2= 000000              29
AMTYP3= 000000              29
AMTYP4= 000000              29
ANAME   002264            480#
APASS = 000000              29
APRIOR= 000000              29
APTCSU= 000040            2808#   2813
APTENV= 000001            2808#   2809    2813
APTSIZ= 000200             492    2808#
APTSPO= 000100            2808#   2813
ASWREG= 000000              29
ATESTN= 000000              29
AUNIT = 000000              29
AUSWR = 000000              29
AVECT1= 100320              18#     29
AVECT2= 000000              29
BADR    023416            2458    2492    2534#
BIT0  = 000001              16#     595     614     648     651     670     673     676     698     728     757     785     813
                            840     867     894     921     948     975     1002    1025    1051    1078    1105    1132    1157
```

```
                        1183    1210    1238    1265    1292    1319    1346    1377    1435    1438    1791    1814    1837
                        1860    1883    1906    1964    1988    2012    2047    2275    2319    2322    2385    2388    2620
BIT00 = 000001            16#
BIT01 = 000002            16#
BIT02 = 000004            16#
BIT03 = 000010            16#
BIT04 = 000020            16#
BIT05 = 000040            16#
BIT06 = 000100            16#
BIT07 = 000200            16#
BIT08 = 000400            16#   2806
BIT09 = 001000            16#   2806    2809
BIT1  = 000002            16#
BIT10 = 002000            16#   2809
BIT11 = 004000            16#   2806
BIT12 = 010000            16#   2672
BIT13 = 020000            16#   2809
BIT14 = 040000            16#   2806
BIT15 = 100000            16#
BIT2  = 000004            16#   1937    2600    2703    2708
BIT3  = 000010            16#   1374
BIT4  = 000020            16#   1969    1993
BIT5  = 000040            16#
BIT6  = 000100            16#
BIT7  = 000200            16#
BIT8  = 000400            16#
BIT9  = 001000            16#
BPTVEC= 000014            16#
BSTART  023432          2455    2541#
BSTRT   024256          2590    2658#
BUFFER  032374-          591*    593     611*    612     640*    643     646     662*    663     668     687*    688*    694
                         711*    717*    722*    723*    724     747*    748*    753     770*    775*    776*    781     798*
                         803*    804*    809     824*    830*    831*    836     838     851*    857*    858*    863     878*
                         884*    885*    890     905*    911*    912*    917     932*    938*    939*    944     946     959*
                         965*    966*    971     986*    992*    993*    998     1013*   1018*   1019*   1021    1033    1035*
                        1041*   1042*   1047    1062*   1068*   1069*   1074    1089*   1095*   1096*   1101    1116*   1122*
                        1123*   1128    1143*   1149*   1150*   1152    1165    1167*   1173*   1174*   1179    1181    1194*
                        1200*   1201*   1206    1221*   1228*   1229*   1234    1249*   1255*   1256*   1261    1276*   1282*
                        1283*   1288    1303*   1309*   1310*   1315    1330*   1336*   1337*   1342    1357*   1363*   1364*
                        1365*   1370    1388*   1395*   1396    1411*   1412    1429    1431*   1432*   1433    1436    1450*
                        1451*   1461*   1470*   1471*   1481*   1491*   1492*   1503*   1512*   1521*   1522*   1532*   1533*
                        1544*   1553*   1554*   1563*   1572*   1581*   1582*   1591*   1600*   1601*   1610*   1619*   1620*
                        1627*   1638*   1639*   1646*   1657*   1658*   1665*   1676*   1677*   1684*   1695*   1696*   1705*
                        1706*   1716*   1726*   1727*   1738*   1745*   1754*   1757*   1766*   1767*   1768*   1770*   1785*
                        1789    1800*   1801*   1808*   1812    1823*   1824*   1831*   1835    1846*   1847*   1854*   1858
                        1869*   1870*   1877*   1881    1892*   1893*   1900*   1904    1915*   1916*   1924    1933    1946*
                        1947*   1952    1953*   1954*   1955*   1959*   1960    1978    1979*   1980*   1981*   1984    1985*
                        2008*   2010    2043*   2045    2073    2084*   2101    2112*   2129    2141*   2142*   2167    2177*
                        2202    2212*   2305*   2313    2408    2512    2517    2522    2527    2541    2543    2544    2545
                        2556    2658    2660    2661    2662    2690*   2691*   2692*   2699    2704    2795    2904#
B1716   002226           459#   2396*   2399    2897
CALTAN  024732          2178    2213    2767#
CENAB   002216           455#    503*    522*   2257    2297
CNTR    025114          2793*   2799*   2803#
CONS    002266           481#   1527    1535    1587    1593    1606    1612    1625    1631    1644    1650    1663    1669
                        1682    1688    2470*   2476    2478    2537    2604*   2610    2612
```

```
CR     = 000015              16#   2813
CRLF   = 000200              16#   2813
CSIZR    002616             503#
DDISP  = 177570              16#     29    492
DELAY    024446             594     613    647    669    672    697    727    756    784    812    839    866    893
                            920     947    974   1001   1024   1050   1077   1104   1131   1156   1182   1209   1237
                           1264    1291   1318   1345   1373   1376   1434   1437   1790   1813   1836   1859   1882
                           1905    1936   1963   1968   1987   1992   2011   2046   2274   2318   2321   2384   2387
                           2599    2619  2681#   2702   2707
DH1      031763              34      48     55    218    225    232    239    246    253    260    267    274    281
                            288     295    302    309    316    323    330    337    344    351    358    365    372
                            379     386    393    414    428    435  2889#
DH2      032030              62      69     76     83     90     97    104    111    119    126    133    140    147
                            154     161    168    175    182    189    196    203    211  2890#
DH3      032110             421    2891#
DH4      032165             400     407   2892#
DH5      032232              41    2893#
DISPLA   001142              29#    492*  2806*  2809*
DISPRE   000174              21#    492
DLYCNT   002206             451#  2681*  2682*
DNAME    002270             482#  1446   1453   2538
DO       024512            1452    1472   1493   1513   1534   1555   1573   1592   1611   1630   1649   1668   1687
                           1707    1728   1748   1769   1788   1811   1834   1857   1880   1903  2698#
DPC      002244             472#    497    501    593*   612*   631*   646*   668*   671*   694*   696*   724*   726*
                            753*   755*   781*   783*   809*   811*   836*   838*   863*   865*   890*   892*   917*
                            919*   944*   946*   971*   973*   998*  1000*  1021*  1023*  1047*  1049*  1074*  1076*
                           1101*  1103*  1128*  1130*  1152*  1155*  1179*  1181*  1206*  1208*  1234*  1236*  1261*
                           1263*  1288*  1290*  1315*  1317*  1342*  1344*  1370*  1372*  1375*  1396*  1412*  1428
                           1433*  1436*  1439   1771*  1789*  1812*  1835*  1858*  1881*  1904*  1933*  1935*  1960*
                           1962*  1965*  1966   1967*  1970   1984*  1986*  1989*  1990   1991*  1994   2010*  2045*
                           2235*  2248*  2273*  2315*  2316   2317*  2320*  2323   2383*  2386*  2389*  2463*  2465*
                           2534   2596*  2598*  2616   2618*  2699*  2701*  2704*  2706*  2795*  2798*
DPCONV   024576            2373   2714#
DPCON1   024610            2394   2719#
DPCSAV   024246           2616*  2617*  2618   2621*  2650#
DSWR   = 177570              16#     29    492
DT1      032310              35      49     56    219    226    233    240    247    254    261    268    275    282
                            289     296    303    310    317    324    331    338    345    352    359    366    373
                            380     387    394    415    429    436  2895#
DT2      032324              42      63     70     77     84     91     98    105    112    120    127    134    141
                            148     155    162    169    176    183    190    197    204    212  2896#
DT3      032342             422    2897#
DT4      032360             401     408   2898#
EMTVEC=  000030              16#    492*
EM1      027554              33   2831#
EM10     027764              82   2838#
EM11     030013              89   2839#
EM12     030045              96   2840#
EM13     030100             103   2841#
EM14     030132             110   2842#
EM15     030160             118   2843#
EM16     030202             125   2844#
EM17     030221             132   2845#
EM2      027603              40   2832#
EM20     030245             139   2846#
EM21     030274             146   2847#
```

```
EM22     030325              153     2848#
EM23     030355              160     2849#
EM24     030404              167     2850#
EM25     030441              174     2851#
EM26     030465              181     2852#
EM27     030511              188     2853#
EM3      027643               47     2833#
EM30     030541              195     2854#
EM31     030571              202     2855#
EM32     030624              210     2856#
EM33     030650              217     2857#
EM34     030667              224     2858#
EM35     030710              231     2859#
EM36     030723              238     2860#
EM37     030737              245     2861#
EM4      027662               54     2834#
EM40     030763              252     2862#
EM41     031005              259     2863#
EM42     031030              266     2864#
EM43     031053              273     2865#
EM44     031072              280     2866#
EM45     031106              287     2867#
EM46     031133              294     2868#
EM47     031160              301     2869#
EM5      027704               61     2835#
EM50     031210              308     2870#
EM51     031231              315     2871#
EM52     031260              322     2872#
EM53     031275              329     2873#
EM54     031311              336     2874#
EM55     031336              343     2875#
EM56     031357              350     2876#
EM57     031404              357     2877#
EM6      027722               68     2836#
EM60     031433              364     2878#
EM61     031463              371     2879#
EM62     031504              378     2880#
EM63     031532              385     2881#
EM64     031553              392     2882#
EM65     031574              399     2883#
EM66     031614              406     2884#
EM67     031627              413     2885#
EM7      027745               75     2837#
EM70     031644              420     2886#
EM71     031672              427     2887#
EM72     031722              434     2888#
ERRVEC=  000004               16#     492*    505*    516*    527*    538*    540*    552*    2806*
EXECUT   025044             2085     2113    2143    2179    2214    2792#
GNS   =  ****** U             21     2819
G1716    002224              458#    2375*   2399    2897
HT    =  000011               16#    2813
IOTVEC=  000020               16#     492*
KIPAR0   002230              463#    529*
KIPAR2   002234              465#    531*    546*    547     553*    554     2357*   2365    2367*   2370*   2372    2402*   2403
KIPAR7   002240              467#    533*    547
KIPDR0   002232              464#    530*
```

```
KIPDR2  002236            466#  532*
KIPDR7  002242            468#  534*
KTENAB  002220            456#  504*  513*  523   537*  557*  2353  2356*
KTERRV= 000250            445#  526*
KTMAX   002222            457#  554*  555   559   2365  2403
KTSER   024630            526   2731#
KTSIZR  002736            524   526#
KTSR0 = 177572            446#  528   542*  545*  550*  2360*  2364*  2731*
LDRSV1  002210            452#  2270*  2284  2310*  2330
LDRSV2  002212            453#  2311*  2331
LF    = 000012             16#  2813
MAINT1= 010000            442#  2233
MAINT2= 020000            443#
MEMMAX  002214            454#  518*  565   2262  2301
MSG1    027423            495   2827#
MSG2    027511           2732   2828#
MSG3    027530            573   2829#
MSG4    027550            576   2830#
OPEN  = 000000            441#  450   451   452   453   454   455   456   457   458   459   472   473
                          474   475   476   477   478   479   480   481   482   483   484   485   486
                          487
PIRQ  = 177772             16#
PIRQVE= 000240             16#
POPSET  024462           1448  1468  1489  1510  1530  1551  1570  1589  1608  1628  1647  1666  1685
                         1703  1724  1746  1764  1786  1809  1832  1855  1878  1901  2689#
PR0   = 000000             16#
PR1   = 000040             16#
PR2   = 000100             16#
PR3   = 000140             16#
PR4   = 000200             16#
PR5   = 000240             16#
PR6   = 000300             16#
PR7   = 000340             16#
PS    = 177776             16#
PSW   = 177776             16#  579*  606*  1920*  2009*  2013*  2044*  2048*  2462*  2471*  2595*  2605*  2739*
PWRMSG  027356           2821  2822#
PWRVEC= 000024              16#  492*  2821*
RCNT    023444           2457  2501  2548#
RCNT1   024270           2592  2638  2665#
RESVEC= 000010              16#
RLO     002254            476#  2234*  2247*  2254*  2371*  2378*  2381*  2405*
RSTART  003244             22   572   577#  2676  2821
RSTVEC  024646           2032  2067  2508  2647  2739#
SETPTR  002576            498#  502
SILOR   023344           2456  2459  2495  2512#
SILOR1  024250           2591  2632  2642  2652#
SREG0   002246            473#  1549  1556  1701  1708  1721  1729  1743  1749  1762  1773  2028  2063
                         2535
SREG1   002256            477#  607   617   621   1413  1416  1420  1466  1473  1486  1494  1508  1514
                         1568  1574  2004  2016  2021  2039  2051  2056  2236  2237  2250  2536
STACK = 001100             16#  492   578
START   002304             21   492#
START1  002566            494   496#
STKLMT= 177774             16#
STKPT   002276            485#  588   592*  596   608*  628   630*  632   645*  649*  667*  674*  693*
                          699*  721*  729*  752*  758*  780*  786*  808*  814*  835*  841*  862*  868*
```

```
                        889*   895*   916*   922*   943*   949*   970*   976*   997*  1003*  1020*  1026*  1046*
                       1052*  1073*  1079*  1100*  1106*  1127*  1133*  1151*  1158*  1178*  1184*  1205*  1211*
                       1233*  1239*  1260*  1266*  1287*  1293*  1314*  1320*  1341*  1347*  1369*  1378*  1393*
                       1397   1398   1408*  1430*  1792*  1815*  1838*  1861*  1884*  1907*  1932*  1940*  1957*
                       1983*  2005*  2041*  2233*  2272*  2276*  2314*  2382*  2461*  2469*  2594*  2603*  2623*
                       2689*  2698*  2700*  2705*  2794*  2796*  2797*
STKVAL  002272          483#    639    650    661    675    695    700    725    730    754    759    782    787
                        810    815    837    842    864    869    891    896    918    923    945    950    972
                        977    999   1004   1022   1027   1048   1053   1075   1080   1102   1107   1129   1134
                       1153   1159   1180   1185   1207   1212   1226   1235   1240   1262   1267   1289   1294
                       1316   1321   1343   1348   1371   1379   1783   1793   1806   1816   1829   1839   1852
                       1862   1875   1885   1898   1908   1922   1941   2260   2277   2622   2624
SWR     001140           29#    492*  2672   2806   2809   2821*
SWREG   000176           21#    492
SW0   = 000001           16#
SW00  = 000001           16#
SW01  = 000002           16#
SW02  = 000004           16#
SW03  = 000010           16#
SW04  = 000020           16#
SW05  = 000040           16#
SW06  = 000100           16#
SW07  = 000200           16#
SW08  = 000400           16#
SW09  = 001000           16#
SW1   = 000002           16#
SW10  = 002000           16#
SW11  = 004000           16#
SW12  = 010000           16#
SW13  = 020000           16#
SW14  = 040000           16#
SW15  = 100000           16#
SW2   = 000004           16#
SW3   = 000010           16#
SW4   = 000020           16#
SW5   = 000040           16#
SW6   = 000100           16#
SW7   = 000200           16#
SW8   = 000400           16#
SW9   = 001000           16#
TANCON  024706          2185    2220   2755#
TANGNT  025042          2770*   2772   2774*  2778*  2785#
TBITVE= 000014            16#
TERMCH  002274           484#   1958*  1982*
TKVEC = 000060            16#
TPVEC = 000064            16#
TRAPVE= 000034            16#    492*
TRTVEC= 000014            16#
TSTNUM  002204           450#   2809*  2895   2896   2897   2898
TST1    003264           586#
TST10   004546           740    745#
TST100  020400          2098#
TST101  020554          2126#
TST102  020752          2153   2163#
TST103  021144          2193   2198#
TST104  021322          2229#
```

```
TST105  021516            2256#
TST106  021732            2258    2292    2296#
TST107  022204            2298    2339    2343#
TST11   004744             769     773#
TST110  022546            2354    2407#
TST111  023456            2510    2555#
TST112  024302            2648    2671#
TST12   005144             797     801#
TST13   005334             828#
TST14   005534             855#
TST15   005732             882#
TST16   006122             909#
TST17   006310             936#
TST2    003412             605#
TST20   006502             963#
TST21   006670             990#
TST22   007060            1016#
TST23   007230            1034    1039#
TST24   007420            1066#
TST25   007606            1093#
TST26   007776            1120#
TST27   010166            1147#
TST3    003546             622     627#
TST30   010346            1171#
TST31   010534            1198#
TST32   010724            1225#
TST33   011122            1253#
TST34   011312            1280#
TST35   011502            1307#
TST36   011672            1334#
TST37   012062            1361#
TST4    003632             635     638#
TST40   012272            1392#
TST41   012404            1407#
TST42   012534            1427#
TST43   012652            1441    1444#
TST44   013006            1459    1464#
TST45   013142            1479    1484#
TST46   013300            1501    1506#
TST47   013426            1520    1525#
TST5    003772             560#
TST50   013564            1542    1547#
TST51   013712            1562    1566#
TST52   014040            1580    1585#
TST53   014166            1599    1604#
TST54   014314            1618    1623#
TST55   014442            1637    1642#
TST56   014570            1656    1661#
TST57   014716            1675    1680#
TST6    004144             686#
TST60   015044            1694    1699#
TST61   015172            1714    1719#
TST62   015330            1736    1741#
TST63   015456            1756    1760#
TST64   015624            1781#
TST65   015772            1799    1804#
```

```
TST66   016142            1822    1827#
TST67   016312            1845    1850#
TST7    004342             710     714#
TST70   016462            1868    1873#
TST71   016632            1891    1896#
TST72   017002            1914    1919#
TST73   017220            1951#
TST74   017406            1977#
TST75   017554            1996    1999#
TST76   020000            2034#
TST77   020224            2069#
TYPCOR  003202             521     525     539     558     565#
TYPDS = 104405             575    2676    2819#
TYPE  = 104401             495     573     576    2676    2732    2809    2811    2813    2815    2817    2819#   2821
TYPMEM  003216             564     570#
TYPOC = 104402            2811    2819#
TYPON = 104404            2819#
TYPOS = 104403            2819#
XDOFF   002260             478#   2195    2201
XPOS    002250             474#   2072    2088    2157    2194    2200    2392
YDOFF   002262             479#   2166
YPOS    002252             475#   2100    2116    2128    2165
ZDOFF   002302             487#
ZPOS    002300             486#
$APTHD  001000              27#
$ASTAT= ****** U          2808
$ATYC   025420            2808#
$ATY1   025374            2808#
$ATY3   025402            2808#   2813
$ATY4   025412            2808#   2809
$AUTOB  001134              29#
$BASE   001256              29#    496
$BDADR  001122              29#    588*    607*    628*    639*    661*    695*    725*    754*    782*    810*    837*    864*
                           891*    918*    945*    972*    999*   1022*   1048*   1075*   1102*   1129*   1153*   1180*   1207*
                          1226*   1235*   1262*   1289*   1316*   1343*   1371*   1397*   1413*   1428*   1446*   1466*   1486*
                          1508*   1527*   1549*   1568*   1587*   1606*   1625*   1644*   1663*   1682*   1701*   1721*   1743*
                          1762*   1783*   1806*   1829*   1852*   1875*   1898*   1922*   1966*   1990*   2004*   2039*   2072*
                          2100*   2146*   2182*   2217*   2236*   2260*   2316*   2476*   2482*   2483    2610*   2622*   2895
                          2896
$BDDAT  001126              29#    596*    597*    598     619*    624*    632*    633*    634     650*    651*    652     675*
                           676*    677     700*    701*    702     730*    731*    732     759*    760*    761     787*    788*
                           789     815*    816*    817     842*    843*    844     869*    870*    871     896*    897*    898
                           923*    924*    925     950*    951*    952     977*    978*    979    1004*   1005*   1006    1027*
                          1028*   1029    1053*   1054*   1055    1080*   1081*   1082    1107*   1108*   1109    1134*   1135*
                          1136    1159*   1160*   1161    1185*   1186*   1187    1212*   1213*   1214    1240*   1241*   1242
                          1267*   1268*   1269    1294*   1295*   1296    1321*   1322*   1323    1348*   1349*   1350    1379*
                          1380*   1381    1398*   1399*   1400    1418*   1423*   1439*   1440    1453*   1454*   1455    1473*
                          1474*   1475    1494*   1495*   1496    1514*   1515*   1516    1535*   1536*   1537    1556*   1557*
                          1558    1574*   1575*   1576    1593*   1594*   1595    1612*   1613*   1614    1631*   1632*   1633
                          1650*   1651*   1652    1669*   1670*   1671    1688*   1689*   1690    1708*   1709*   1710    1729*
                          1730*   1731    1749*   1750*   1751    1773*   1774*   1775    1793*   1794*   1795    1816*   1817*
                          1818    1839*   1840*   1841    1862*   1863*   1864    1885*   1886*   1887    1908*   1909*   1910
                          1941*   1942*   1943    1970*   1971    1994*   1995    2016*   2017*   2023*   2030*   2051*   2052*
                          2058*   2065*   2088*   2089    2116*   2117    2147*   2148    2183*   2186    2218*   2221    2237*
                          2238*   2239    2250*   2251*   2277*   2278    2323*   2324    2395*   2397    2478*   2483*   2485
                          2612*   2624*   2626    2755*   2759*   2895    2896    2897    2898
```

```
$BELL    001172              29#   2809
$CDW1    001262              29#
$CHARC   026456             2813#*
$CKSWR=  ****** U           2819
$CMTAG   001100              29#    492
$CM1   = 000001              29#
$CM2   = 000002              29#
$CM3   = 000001              29#
$CM4   = 000001              29#
$CPUOP   001230              29#
$CRLF    001177              29#   2809    2811    2813
$DBLK    027124             2817#
$DEVCT   001212              29#
$DEVM    001260              29#
$DOAGN   024422             2676#
$DTBL    027114             2817#
$ENDAD   024412               24   2676#   2809
$ENDCT   024360              492   2676#
$ENDMG   024431             2676#
$ENULL   024426             2676#
$ENV     001222              29#   2808    2809    2813
$ENVM    001223              29#    492    2808    2813
$EOP     024324             2673   2676#
$EOPCT   024352              492*  2676#
$ERFLG   001103              29#   2806*   2809*
$ERMAX   001115              29#    492*   2806*
$ERROR   025642              492   2809#
$ERRPC   001116              29#   2809*   2811    2895    2896    2897    2898
$ERRTB   001264              29#   2811
$ERRTY   026044             2809   2811#
$ERTTL   001112              29#   2809*
$ESCAP   001170              29#    492*   2806*   2809
$ETABL   001222              29#
$ETEND   001264              27     29#
$FATAL   001204              29#   2808*
$FFLG    025640             2808#*
$FILLC   001156              29#   2813
$FILLS   001155              29#   2813
$GDADR   001120              29#
$GDDAT   001124              29#    590*    598     602*    609*    623*    629*    634     643*    652     663*    677     689*
                            702     709*    715*    732     742*    746*    761     768*    774*    789     796*    802*    817
                            825*    829*    844     852*    856*    871     879*    883*    898     906*    910*    925     933*
                            937*    952     960*    964*    979     987*    991*   1006    1014*   1017*   1029    1040*   1055
                           1063*   1067*   1082    1090*   1094*   1109    1117*   1121*   1136    1144*   1148*   1161    1172*
                           1187    1195*   1199*   1214    1222*   1227*   1242    1250*   1254*   1269    1277*   1281*   1296
                           1304*   1308*   1323    1331*   1335*   1350    1358*   1362*   1381    1389*   1394*   1400    1403*
                           1404    1409*   1422*   1429*   1440    1447*   1451    1455    1458*   1467*   1471    1475    1478*
                           1487*   1496    1499*   1509*   1516    1519*   1528*   1537    1540*   1550*   1554    1558    1561*
                           1569*   1576    1579*   1588*   1595    1598*   1607*   1614    1617*   1626*   1633    1636*   1645*
                           1652    1655*   1664*   1671    1674*   1683*   1690    1693*   1702*   1706    1710    1713*   1722*
                           1731    1735*   1744*   1751    1755*   1763*   1768    1775    1778*   1784*   1795    1798*   1807*
                           1818    1821*   1830*   1841    1844*   1853*   1864    1867*   1876*   1887    1890*   1899*   1910
                           1913*   1923*   1943    1948*   1952*   1971    1978*   1995    2007*   2026*   2042*   2061*   2083*
                           2089    2093*   2111*   2117    2121*   2140*   2148    2150    2158*   2160*   2186    2221    2230*
                           2239    2242*   2246*   2268*   2278    2308*   2324    2374*   2383    2397    2477*   2484*   2485
                           2611*   2625*   2626    2769*   2772*   2773*   2777*   2780    2782*   2895    2896    2897    2898
```

```
$GET42  024402            2676#
$GTSWR= ****** U          2819
$HD   = 000000              12
$HIBTS  001000              27#
$ICNT   001104              29#   2806*
$ILLUP  027350            2821#
$INTAG  001135              29#
$ITEMB  001114              29#   2809*   2811
$LF     001200              29#   2809    2813
$LFLG   025637            2808#*
$LPADR  001106              29#    492*   2806*
$LPERR  001110              29#    492*    587*   644*   666*   690*   718*   749*   777*   805*   832*   859*   886*
                           913*    940*    967*   994*  1043*  1070*  1097*  1124*  1175*  1202*  1230*  1257*  1284*
                          1311*   1338*   1366*  1445*  1465*  1485*  1507*  1526*  1548*  1567*  1586*  1605*  1624*
                          1643*   1662*   1681*  1700*  1720*  1742*  1761*  1782*  1805*  1828*  1851*  1874*  1897*
                          1921*   1961*   2071*  2099*  2127*  2164*  2199*  2232*  2245*  2259*  2299*  2390*  2466*
                          2589*   2806*   2809
$MADR1  001234              29#
$MADR2  001240              29#
$MADR3  001244              29#
$MADR4  001250              29#
$MAIL   001202              27     29#    492    586    605    627    638    660    686    714    745    773    801
                           828    855    882    909    936    963    990   1016   1039   1066   1093   1120   1147
                          1171   1198   1225   1253   1280   1307   1334   1361   1392   1407   1427   1444   1464
                          1484   1506   1525   1547   1566   1585   1604   1623   1642   1661   1680   1699   1719
                          1741   1760   1781   1804   1827   1850   1873   1896   1919   1951   1977   1999   2034
                          2069   2098   2126   2163   2198   2229   2256   2296   2343   2407   2555   2671   2806
                          2809   2813
$MAMS1  001232              29#
$MAMS2  001236              29#
$MAMS3  001242              29#
$MAMS4  001246              29#
$MBADR  001002              27#
$MFLG   025636            2808#*
$MSGAD  001216              29#   2808*
$MSGLG  001220              29#   2808*
$MSGTY  001202              29#   2808*
$MTYP1  001233              29#
$MTYP2  001237              29#
$MTYP3  001243              29#
$MTYP4  001247              29#
$MXCNT  025372            2806#
$NULL   001154              29#   2813
$NWTST= 000001             586#    605#    627#   638#   660#   686#   714#   745#   773#   801#   828#   855#   882#
                           909#    936#    963#   990#  1016#  1039#  1066#  1093#  1120#  1147#  1171#  1198#  1225#
                          1253#   1280#   1307#  1334#  1361#  1392#  1407#  1427#  1444#  1464#  1484#  1506#  1525#
                          1547#   1566#   1585#  1604#  1623#  1642#  1661#  1680#  1699#  1719#  1741#  1760#  1781#
                          1804#   1827#   1850#  1873#  1896#  1919#  1951#  1977#  1999#  2034#  2069#  2098#  2126#
                          2163#   2198#   2229#  2256#  2296#  2343#  2407#  2555#  2671#
$OCNT   026704            2815#*
$OMODE  026706            2815#*
$OVER   025356            2806#
$PASS   001210              29#    492*   2676*  2806
$PASTM  001006              27#
$PWRAD  027344            2821#
$PWRDN  027204             492    2821#
```

```
$PWRMG  027340          2821#
$PWRUP  027256          2821#
$QUES   001176            29#   2809    2813
$RDCHR= ****** U        2819
$RDDEC= ****** U        2819
$RDLIN= ****** U        2819
$RDOCT= ****** U        2819
$REGAD  001160            29#
$REGO   001162            29#
$RTNAD  024424          2676#
$R2A  = ****** U        2819
$SAVRE= ****** U        2819
$SAVR6  027354          2821#*
$SCOPE  025116           492   2806#
$SETUP= 000037           491#   492    2676    2806    2809
$STUP = 177777           491#
$SVLAD  025322          2806#
$SVPC = 000210            24#
$SWR  = 167400             9#    12     20     29     492     586     605     627     638     660     686     714     745
                         773    801    828    855     882     909     936     963     990    1016    1039    1066    1093
                        1120   1147   1171   1198    1225    1253    1280    1307    1334    1361    1392    1407    1427
                        1444   1464   1484   1506    1525    1547    1566    1585    1604    1623    1642    1661    1680
                        1699   1719   1741   1760    1781    1804    1827    1850    1873    1896    1919    1951    1977
                        1999   2034   2069   2098    2126    2163    2198    2229    2256    2296    2343    2407    2555
                        2671   2676   2806   2809    2821
$SWREG  001224            29#   492
$SWRMK= 000000            20   2806
$TESTN  001206            29#   586*   605*   627*   638*   660*   686*   714*   745*   773*   801*   828*   855*
                         882*   909*   936*   963*   990*  1016*  1039*  1066*  1093*  1120*  1147*  1171*  1198*
                        1225*  1253*  1280*  1307*  1334*  1361*  1392*  1407*  1427*  1444*  1464*  1484*  1506*
                        1525*  1547*  1566*  1585*  1604*  1623*  1642*  1661*  1680*  1699*  1719*  1741*  1760*
                        1781*  1804*  1827*  1850*  1873*  1896*  1919*  1951*  1977*  1999*  2034*  2069*  2098*
                        2126*  2163*  2198*  2229*  2256*  2296*  2343*  2407*  2555*  2671*  2806*
$TIMES  001166            29#   492*   686*   714*   745*   773*   828*   855*  1147*  1407*  1444*  1464*  1760*
                        1951*  1999*  2034*  2069*  2098*  2126*  2163*  2198*  2229*  2343*  2407*  2555*  2676*
                        2806*
$TKB    001146            29#
$TKS    001144            29#
$TMPO   001164            29#   654*   679*   704*   734*   763*   791*   819*   846*   873*   900*   927*   954*
                         981*  1008*  1031*  1057*  1084*  1111*  1138*  1163*  1189*  1216*  1244*  1271*  1298*
                        1325*  1352*  1383*  2091*  2119*  2150*  2188*  2223*  2267*  2276   2282*  2307*  2328*
                        2460*  2497*  2498   2500*  2593*  2634*  2635   2637*  2896   2898
$TN   = 000113             10#    12    586#   605#   622     627#   635     638#   660#   686#   710     714#   740
                         745#   769    773#   797    801#    828#   855#    882#   909#   936#   963#    990#  1016#
                        1034   1039#  1066#  1093#  1120#   1147#  1171#   1198#  1225#  1253#  1280#   1307#  1334#
                        1361#  1392#  1407#  1427#  1441    1444#  1459    1464#  1479   1484#  1501    1506#  1520
                        1525#  1542   1547#  1562   1566#   1580   1585#   1599   1604#  1618   1623#   1637   1642#
                        1656   1661#  1675   1680#  1694    1699#  1714    1719#  1736   1741#  1756    1760#  1781#
                        1799   1804#  1822   1827#  1845    1850#  1868    1873#  1891   1896#  1914    1919#  1951#
                        1977#  1996   1999#  2034#  2069#   2098#  2126#   2153   2163#  2193   2198#   2229#  2256#
                        2258   2292   2296#  2298   2339    2343#  2354    2407#  2510   2555#  2648    2671#
$TPB    001152            29#   2813*
$TPFLG  001157            29#   2813
$TPS    001150            29#   2813
$TRAP   027134           492    2819#
$TRAP2  027156          2819#
```

```
$TRP  = 000006            2819#
$TRPAD  027170            2819#
$TSTM   001004              27#
$TSTNM  001102              29#     580*    2676*    2806*    2809
$TYPBN= ****** U          2819
$TYPDS  026710            2817#   2819
$TYPE   026200            2808    2813#   2819
$TYPEC  026412            2813#
$TYPEX  026460            2813#
$TYPOC  026506            2815#   2819
$TYPON  026522            2815#   2819
$TYPOS  026462            2815#   2819
$UNIT   001214              29#
$UNITM  001010              27#
$USWR   001226              29#
$VECT1  001252              29#   2000     2035     2451     2585     2741
$VECT2  001254              29#
$XTSTR  025126            2806#
$$GET4= 000000            2676#
$OFILL  026705            2815#*
$40CAT= ****** U          2806    2809
.       = 032376              21#     24#      26#      27#      29#      492      2676     2806     2808#    2809     2811#    2813     2817#
                          2821
.$ASTA= ****** U          2808
.$X     = 001000            27#
```

```
COMMEN    16#
ENDCOM    16#
ERROR     16#   600   620   625   636   655   680   705   735   764   792   820   847   874   901
           928   955   982  1009  1032  1058  1085  1112  1139  1164  1190  1217  1245  1272  1299
          1326  1353  1384  1402  1419  1424  1442  1457  1477  1498  1518  1539  1560  1578  1597
          1616  1635  1654  1673  1692  1712  1733  1753  1777  1797  1820  1843  1866  1889  1912
          1945  1973  1997  2018  2024  2031  2053  2059  2066  2092  2120  2151  2189  2224  2241
          2253  2280  2326  2401  2479  2487  2613  2628
ESCAPE    16#
GETPRI    16#
GETSWR    16#
MULT      16#
NEWTST    16#   586   605   627   638   660   686   714   745   773   801   828   855   882   909
           936   963   990  1016  1039  1066  1093  1120  1147  1171  1198  1225  1253  1280  1307
          1334  1361  1392  1407  1427  1444  1464  1484  1506  1525  1547  1566  1585  1604  1623
          1642  1661  1680  1699  1719  1741  1760  1781  1804  1827  1850  1873  1896  1919  1951
          1977  1999  2034  2069  2098  2126  2163  2198  2229  2256  2296  2343  2407  2555  2671
POP       16#  2808  2817  2821
PUSH      16#  2808  2817  2821
REPORT    16#
SCOPE     16#   586   605   627   638   660   686   714   745   773   801   828   855   882   909
           936   963   990  1016  1039  1066  1093  1120  1147  1171  1198  1225  1253  1280  1307
          1334  1361  1392  1407  1427  1444  1464  1484  1506  1525  1547  1566  1585  1604  1623
          1642  1661  1680  1699  1719  1741  1760  1781  1804  1827  1850  1873  1896  1919  1951
          1977  1999  2034  2069  2098  2126  2163  2198  2229  2256  2296  2343  2407  2555  2671
          2676
SETPRI    16#
SETTRA  2819#
SETUP     16#   492
SKIP      16#   622   635   710   740   769   797  1034  1441  1459  1479  1501  1520  1542  1562
          1580  1599  1618  1637  1656  1675  1694  1714  1736  1756  1799  1822  1845  1868  1891
          1914  1996  2153  2193  2258  2292  2298  2339  2354  2510  2648
SLASH     16#
SPACE     16#
STARS     16#    24    27    29   586   605   627   638   660   686   714   745   773   801   828
           855   882   909   936   963   990  1016  1039  1066  1093  1120  1147  1171  1198  1225
          1253  1280  1307  1334  1361  1392  1407  1427  1444  1464  1484  1506  1525  1547  1566
          1585  1604  1623  1642  1661  1680  1699  1719  1741  1760  1781  1804  1827  1850  1873
          1896  1919  1951  1977  1999  2034  2069  2098  2126  2163  2198  2229  2256  2296  2343
          2407  2555  2671  2676  2678  2680  2686  2688  2695  2697  2711  2713  2728  2730  2736
          2738  2750  2754  2763  2766  2787  2791  2806  2808  2809  2811  2813  2815  2817  2819
          2821  2900  2903
SWRSU     16#  492#
TRMTRP  2819#
TYPBIN    16#
TYPDEC    16#  2676
TYPNAM    16#
TYPNUM    16#
TYPOCS    16#
TYPOCT    16#  2811
TYPTXT    16#
$$CMRE    29#
$$CMTM    29#
$$ESCA    16#
$$NEWT    16#   586   605   627   638   660   686   714   745   773   801   828   855   882   909
           936   963   990  1016  1039  1066  1093  1120  1147  1171  1198  1225  1253  1280  1307
```

```
            1334    1361    1392    1407    1427    1444    1464    1484    1506    1525    1547    1566    1585    1604    1623
            1642    1661    1680    1699    1719    1741    1760    1781    1804    1827    1850    1873    1896    1919    1951
            1977    1999    2034    2069    2098    2126    2163    2198    2229    2256    2296    2343    2407    2555    2671
$$SET       2819#
$$SETM       492#
$$SKIP        16#     622     635     710     740     769     797    1034    1441    1459    1479    1501    1520    1542    1562
            1580    1599    1618    1637    1656    1675    1694    1714    1736    1756    1799    1822    1845    1868    1891
            1914    1996    2153    2193    2258    2292    2298    2339    2354    2510    2648
.EQUAT         5#      16
.HEADE         5#      12
.SETUP         5#     491
.SWRHI         5#      20
.SWRLO        20#
.$ACT1         8#      24
.$APTB         8#      29#
.$APTH         8#      27
.$APTY         8#    2808
.$CATC         5#      21
.$CMTA         5#      29
.$EOP          6#    2676
.$ERRO         6#    2809
.$ERRT         7#    2811
.$POWE         6#    2821
.$SCOP         6#    2806
.$TRAP         6#    2819
.$TYPD         7#    2817
.$TYPE         7#    2813
.$TYPO         7#    2815


. ABS.  032376     000     OVR   RO    REL   GBL   D

 ERRORS DETECTED:  0

 CZVSCC,CZVSCC/CRF=CZVSCC
 RUN-TIME: 31 20 1 SECONDS
 RUN-TIME RATIO: 107/54=1.9
 CORE USED:  26K  (51 PAGES)
```