KPV11-A

DIAGNOSTICS

MD-11-DVKPA-A

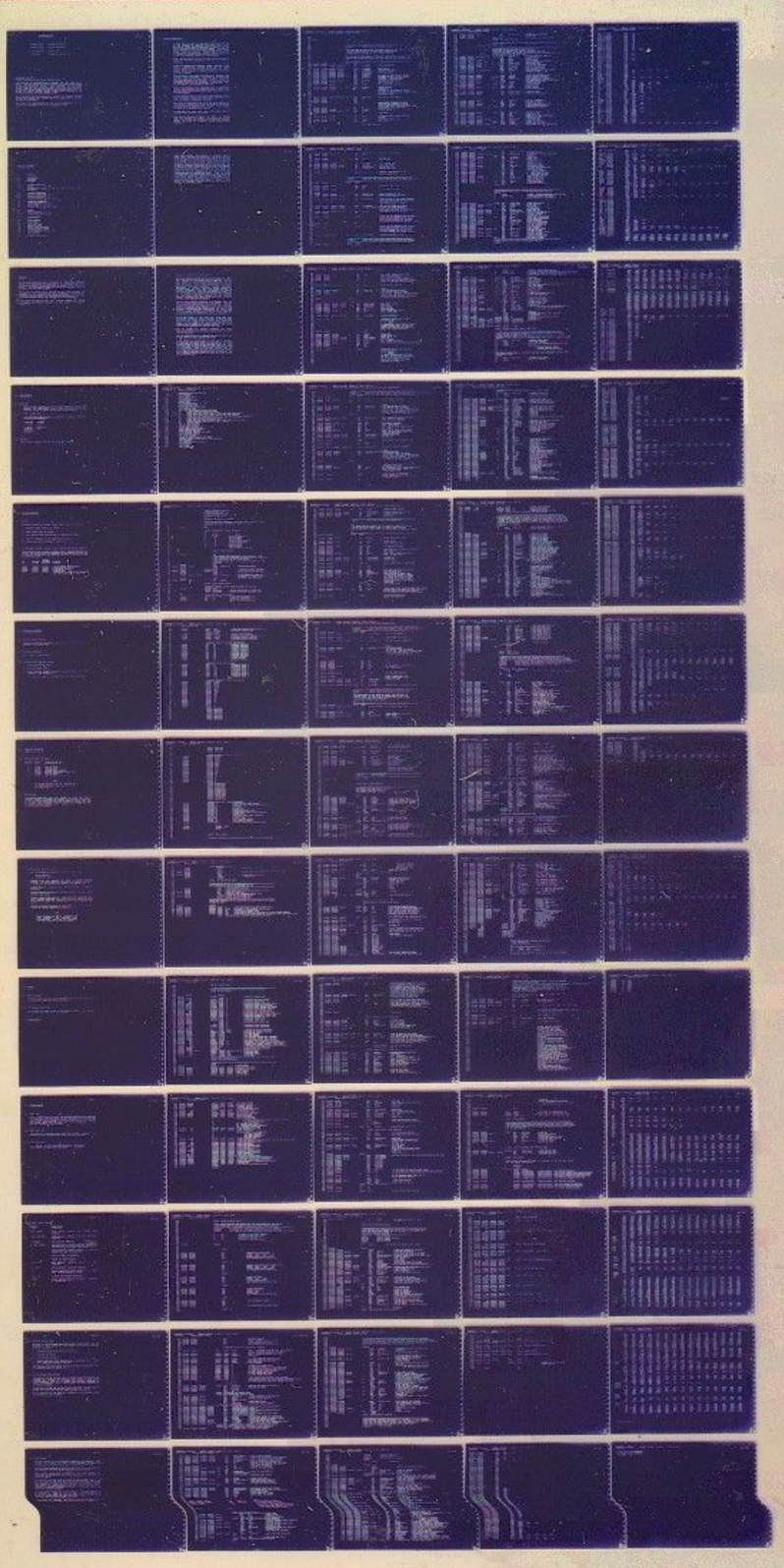
EP DVKPA DL A

FICHE 1 OF 1

JUN 1979

digital

MADE IN USA



IDENTIFICATION

Product Code: MAINDEC-11-DVKPA-A-D

Product Name: KPV11-A Diagnostic

Date Created: January 1977

Maintainer: Diagnostic Engineering

TABLE OF CONTENTS

1.0 ABSTRACT

This program allows the user to checkout or debug the KPV11-A, LSI-11 Power Fail/Line Time Generator. To check-out Power fail, Non-Volatile Memory (i.e., core) must be located in the first 4K memory. The user should check the jumpers on the option as well as jumpers on the LSI-11 module.

Even though the KPV-11A is a LSI-11 option, this program was designed to run on any PDP-11 Family Computer. If the user is unfamiliar with an LSI-11 he should review sections 8.4 and 8.5. A software switch register is included with this program.

Every effort was made to make this program conform to LSI-11 programming restrictions. However, the user should read sections 7.1 and 7.2.

2.0 REQUIREMENTS

2.1 Equipment

1. PDP-11 Family Computer with 4K of core memory (or more) and console I/O facilities (i.e., TTY). If CPU is an LSI-11, Power-up option #1 should be jumpered on CPU module.

2. KPV-11 under test.

   The following jumpers must be installed or removed (Standard Factory Configuration):

INSTALLED	REMOVED
W1,W2,W3	W6,W9,W10
W4,W5,W7	W12
W8,W11,W13	
W14,W15	

2.2 Storage

This program occupies and uses the lower 4K of memory.

3.0 LOADING PROCEDURE

3.1 Method

Standard procedure for normal binary tapes should be followed.

1. Absolute loader must be in memory.
2. Place binary tape in reader.
3. Type address *7500 (*determine by location of loader).
4. Type "G" (program will be loaded into memory).

The program can also be loaded by XXDP, ACT or APT.

3.2 Non-Standard Address, Vector, or Use of Software Switch Register

This program is set to test a KPV11-A with a standard address and vector. If any of these are different on the KPV11-A or KD11 you are testing, change the corresponding location in memory before starting this test.

TAG	ADDRESS	CURRENT CONTENTS	COMMENTS
$BASE	001246	177546	;;BASE ADDRESS OF EQUIPMENT
$VECT1	001242	000100	;;INTERRUPT VECTOR #1
$SWREG	000176	000000	;;MANUAL SWR
noclk	001370	000000	;;indicates if clock to be tested, ;;=0 test clock;;=1 no clock test.

E1
5

4.0 STARTING PROCEDURE

4.1 Control Switch Setting

Before starting the diagnostic, set all switch register bits as desired. See section 5.1.

4.2 Starting Addresses

200 start of Logic Tests

4.3 Program and/or Operator Action

1. Load program into memory.
2. Enter keyboard "ODT".
3. Alter location "SWREG" to reflect desired options of a switch register - See section 5.1.
4. Type starting address, followed by "G" to start program.

5.0 OPERATING PROCEDURE

5.1 Switch Register Function

SWR BIT	OCTAL	FUNCTION WHEN SET
15	100000	HALT ON ERROR
14	040000	LOOP ON TEST
13	020000	INHIBIT ERROR TYPEOUT
11	004000	INHIBIT ITERATIONS (SHORT PASS)
10	002000	BELL ON ERROR
08	000400	LOOP ON TEST IN SWR <7:0>

NOTE

The Switch Register may be changed at any time while the diagnostic is running by typing "^G".

5.2 Scope Loops

If an error occurs and the user wishes to scope the error, "$SWREG" should be altered to "100000" at the start of the test to halt on error, then when the program halts on error and the CPU enters "ODT, "$SWREG" should be altered to "060000" to loop on current test and inhibit error typeout, then type "P" to continue program execution.

5.3 Program and/or Operator Action

1. When the program is initially started it will type:

```
"MD-11-DVKPA-A
SWR=000000  NEW="
```

2. Program waits for operator to enter a switch register setting. If the program is restarted, no typeout will occur to change the Switch Register setting, see section 8.6.

3. Program executes a pass of logic tests, subtest iterations inhibited.

4. Program reports any errors it detects as well as any power failures that occur.

5. Program reports "END PASS XX POWER FAILS YY" where XX = the total passes completed and YY is the total number of power fails detected while the program has run. Both numbers are octal.

6. Program continues execution of steps 3-5 only with subtest iterations (unless inhibited by operator).

NOTE

Power failures CAN NOT be generated by this program. The operator MUST generate them. They can be inflected only after step #2 has been executed.

6.0 ERRORS

6.1 Error printout

Printout varies with the error detected. The error PC typed out is the actual location of the error call.

6.2 Non-Standard Error Halt

Bus errors will cause a Halt in the routine "IOTRD". The address that caused this trap will be in "TRTO".

7.0 RESTRICTIONS

None.

8.0 MISCELLANEOUS

8.1 Power Fail

After a power failure occurs, the program execution will continue at the point where the power failure occurred. The program will type "pf-OK" if no messages were being typed out. If a message was being typed out at the time of the power fail, it will complete typeout of that message.

8.2 XXDP, ACT, APT

The program is chainable under XXDP, ACT, or APT. Although "APT HOOKS" have been installed, they have not been tested.

8.3 Execution Time

0.1 Minutes (6 sec) Iteration Inhibited - No Errors
0.75 Minutes (50 sec) With Iterations - No Errors

8.4 LSI-11 "ODT" Commands

FORMAT	DESCRIPTION
<CR> RETURN	Close opened location and accept next command.
<LF> LINE FEED	Close current location; open next sequential location.
^(UPARROW)	Open previous location.
< (LEFT ARROW)	Take contents of opened location, indexed by contents of PC, and open that location.
@	Take contents of opened location as absolute address and open that location.
R/	Open the word at location R.
/	Reopen the last location.
SN/ or RN/	Open general register N(0-7) or S(PS register).
RSG or RG	GOTO location R and start program.
NL	Execute Bootstrap loader using N as device CSR, Console device is 177560.
SP or P	Proceed with program execution.
RUBOUT	Erases previous numeric character. Response is a backslash (\).

8.5 Entering LSI-11 "ODT"

The halt or ODT microcode state of the KD11 (LSI-11 CPU) can be entered in four different ways (others are a subset of these) from the run state:

1. Execution of an LSI-11 halt instruction.
2. A double bus error.
3. As a power up option.
4. ASCII break with DLV11 framing error asserting the B halt line (enabled by jumper of DLV11).

Upon entering the halt state, the KD11F responds through the set of commands listed in section 8.4.

8.6 Use of Program Software SWR

The software switch register may be changed while the program is running by typing ^G (control and letter G keys typed simultaneously). When ^G is typed, the program responds by typing "SWR=XXXXXX" where XXXXXX equals the former contents of the switch register.

If you wish to keep the current value, type <CR>. If you wish to change the value, type the new value followed by a <CR>.

It is important to note that the diagnostic is not running after the ^G until a <CR> is typed.

8.7 Trap Catcher

The Trap Catcher in this diagnostic employs a new concept. This concept will enable the user of this diagnostic to gain more knowledge of the events that lead the program to this area.

The Trap Catch consists of PC+2 and JSR PC,R0 (i.e., Location 300 would contain 302 and location 302 would contain 4700).

When a device unexpectedly interrupts to the Trap Catcher, it would pick up the PC+2 of the trap as an address of the interrupt service routine.

The program would then pick up "4700" as the new PSW. Bit 7 of the new PSW having been set, would prevent further interrupts from happening. When the CPU attempts to execute "4700" (JSR PC,R0), a Buss-time-out trap will occur to location 4. Location 4 contains a pointer to "IOTRD", a routine that will report the trap as an error.

To guard against "Real" Bus errors routing through location 4 to "IOTRD", we check to see if the trap that brought us to location 4 really came from the Trap Catcher area. If not we'll halt and leave the Trap Address in "TRTO".

More about the interrupt error can be found in the description of the error in the program listing in the routine "IOTRD".

9.0 PROGRAM DESCRIPTION

TST1 — In this test we write all available memory from the end of the program to address 17470 with either a zero pattern or a ones pattern depending on whether we are on an even pass number or odd pass number. We restrict memory checking to the lower 4K of memory since other memory could be volatile. "MILIM" (address 1342) may be changed to reflect an end address greater than 17470 if more non-volatile memory is in your system.

After writing memory, it is read back to assure that it could be written correctly. If not, an error is reported.

During the memory write period, power failures could occur. Appropriate software flags are set while writing memory so that the power up routine can check memory after a power up without confusing the pattern written in memory.

TST2 — Here we check to see that the clock will respond with "B RPLY" when addressed. This test is in two parts: Part One addresses the clock with "B DIN"; Part Two addresses the clock with "B DOUT". If this test fails, no further tests will be performed. The operator must fix the error. He may loop on test two.

TST3 — Here we try setting, then clearing bit 6 (interrupt enable) of the Control and Status Register (CSR). If a power failure occurs during this test, the test will be performed again.

TST4 — In this test we wait to see if the clock CSR bit 7 will set, interrupt enable is also set. After CSR bit 7 sets, we expect an interrupt to occur.

TST5 — This test is designed to see if we can clear CSR bit 7. To do this, we first must allow the clock to interrupt, assuring that the flag just set. Then we can attempt to clear it by writing it to a zero.

TST6 — In this test we make sure that the clock doesn't interrupt when CSR bit 6 is clear.

TST7 — This test is the same as Part 2 of TST1. We verify that the pattern in memory is the same as we wrote there. Only this time we spend a lot of time reading memory, in hopes of catching a power failure while reading it.

PDOWN Power down routine. This routine is entered when a Power down sequence has occurred. The first thing we check (if this is not the first power down) is to see if the last power down had had enough time to complete this routine. If not, we halt. Next we set a flag to indicate that we started this power down routine. Now we save all general registers, as well as error message locations in case we were reporting an error before the power failure occurred. A pointer to the power up routne is put in location 24 so that on power up, program control is transferred there. Next we clear the power down flag indicating that this power down was successfully completed. Last we wait for power to go away while doing a Branch-self. If "POWER OK" isn't asserted before the power goes away, we stand a good chance at changing memory locations since memory was cycling during a power loss.

PUP Routine to handle power recoveries

The first thing we check for is to make sure the power down routine had had enough time to complete. Next we put a pointer to a routine that will take care of the case where we didn't have enough time to start our power up routine. Now we check to see that we didn't experience 5 power failures without completely finishing one. In that case the stack pointer is getting dangerously low and we couldn't put anymore information with it. With the preliminaries done, we put a pointer to out power down routine into location 24. Next we'll verify the contents of our writable memory, checking the flags to be sure that we weren't writing a new pattern into memory.

When the power failure occurred, Flag "BLKCK" was set when we changed pass numbers but not pattern. When the new pattern was written into at least one location, BLKCK is cleared. Flag "MEMWR" is set when we started writing a new pattern and cleared when we fully have written a new pattern. If MEMWR is set, we will get the last address to check from "LWAW", thus checking only memory that was written with the new pattern.

Now we will check to see that the clock CSR will initialize properly. First, however, we check to make sure that the clock passes "TST2" (the addressing test). If it didn't we won't touch the clock here since we can't afford any time-out traps if the clock should prove incapable of passing that test.

Next we restore all general registers and other information that was stored on the stack when the power failure occurred.

Last we check to see if a message was being typed out at the time the power failure occurred.

If no message was being typed, we will type: "PF-OK". If a message was being typed, we will back up the ASCII string pointer to pick up the character that was lost when the power failure occurred, then continue to type out the message. NOTE: If the power failure did not occur in the serial line unit controlling the console I/O device, a duplicate character will be typed of the last character typed before the power fail that caused the CPU to go away.

```
                                .TITLE  MAINDEC-11-DVKPA-A
                                ;*COPYRIGHT (C) 1977
                                ;*DIGITAL EQUIPMENT CORP,
                                ;*MAYNARD, MASS, 01754
                                ;*
                                ;*PROGRAM BY EDWARD C, BADGER
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976,
                                ;*
000001                          STN=1

                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<7:0>

                                .SBTTL  TRAP CATCHER

000000                          .=0
                                ;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ".+2"
                                ;*AND "JSR PC,R0" SEQUENCE TO CATCH ILLEGAL INTERRUPTS,
                                ;*AND INTERRUPTS TO THE WRONG VECTOR,
                                ;*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
                                ;*VECTORS
        000004                  .=4
000004  006614  000200          .WORD   IOTRD,200       ;HANDLE BUS8 ERROR,
        000174                  .=174
000174  000000          DISPREG:.WORD   0               ;;SOFTWARE DISPLAY REGISTER,
000176  000000          SWREG:  .WORD   0               ;;SOFTWARE SWITCH REGISTER,
        000100                  .=100
000100  000104  000200  000002  .WORD   104,200,2       ;IF "B EVENT" ON Q-BUS IS
                                                        ;CONNECTED, WE NEED A WAY OF
        000200                  .=200                   ;IGNORING ITS INTERRUPTS,
000200  000137  001374          JMP     START

                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                                ;*MISCELLANEOUS DEFINITIONS
000011                          HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                          LF=     12              ;;CODE FOR LINE FEED
000015                          CR=     15              ;;CODE FOR CARRIAGE RETURN
```

```
                        ;CRLF=  200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
                        ;PS=    177776          ;;PROCESSOR STATUS WORD
                        ;.EQUIV  PS,PSW
                        ;STKLMT= 177774         ;;STACK LIMIT REGISTER
                        ;PIRQ=   177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
                        ;DSWR=   177570         ;;HARDWARE SWITCH REGISTER
                        ;DDISP=  177570         ;;HARDWARE DISPLAY REGISTER
                        ;
                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
                        ;R0=     %0             ;;GENERAL REGISTER
                        ;R1=     %1             ;;GENERAL REGISTER
                        ;R2=     %2             ;;GENERAL REGISTER
                        ;R3=     %3             ;;GENERAL REGISTER
                        ;R4=     %4             ;;GENERAL REGISTER
                        ;R5=     %5             ;;GENERAL REGISTER
                        ;R6=     %6             ;;GENERAL REGISTER
                        ;R7=     %7             ;;GENERAL REGISTER
                        ;SP=     %6             ;;STACK POINTER
                        ;PC=     %7             ;;PROGRAM COUNTER
                        ;
                        ;*PRIORITY LEVEL DEFINITIONS
                        ;PR0=    0              ;;PRIORITY LEVEL 0
                        ;PR1=    40             ;;PRIORITY LEVEL 1
                        ;PR2=    100            ;;PRIORITY LEVEL 2
                        ;PR3=    140            ;;PRIORITY LEVEL 3
                        ;PR4=    200            ;;PRIORITY LEVEL 4
                        ;PR5=    240            ;;PRIORITY LEVEL 5
                        ;PR6=    300            ;;PRIORITY LEVEL 6
                        ;PR7=    340            ;;PRIORITY LEVEL 7
                        ;
                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
                        ;SW15=   100000
                        ;SW14=   40000
                        ;SW13=   20000
                        ;SW12=   10000
                        ;SW11=   4000
                        ;SW10=   2000
                        ;SW09=   1000
                        ;SW08=   400
                        ;SW07=   200
                        ;SW06=   100
                        ;SW05=   40
                        ;SW04=   20
                        ;SW03=   10
                        ;SW02=   4
                        ;SW01=   2
                        ;SW00=   1
                        ;.EQUIV  SW09,SW9
                        ;.EQUIV  SW08,SW8
                        ;.EQUIV  SW07,SW7
                        ;.EQUIV  SW06,SW6
                        ;.EQUIV  SW05,SW5
                        ;.EQUIV  SW04,SW4
                        ;.EQUIV  SW03,SW3
```

```
                                .EQUIV  SW02,SW2
                                .EQUIV  SW01,SW1
                                .EQUIV  SW00,SW0

                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                          BIT15=  100000
040000                          BIT14=  40000
020000                          BIT13=  20000
010000                          BIT12=  10000
004000                          BIT11=  4000
002000                          BIT10=  2000
001000                          BIT09=  1000
000400                          BIT08=  400
000200                          BIT07=  200
000100                          BIT06=  100
000040                          BIT05=  40
000020                          BIT04=  20
000010                          BIT03=  10
000004                          BIT02=  4
000002                          BIT01=  2
000001                          BIT00=  1
                                .EQUIV  BIT09,BIT9
                                .EQUIV  BIT08,BIT8
                                .EQUIV  BIT07,BIT7
                                .EQUIV  BIT06,BIT6
                                .EQUIV  BIT05,BIT5
                                .EQUIV  BIT04,BIT4
                                .EQUIV  BIT03,BIT3
                                .EQUIV  BIT02,BIT2
                                .EQUIV  BIT01,BIT1
                                .EQUIV  BIT00,BIT0

                                ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                          TBITVEC=14              ;;"T" BIT
000014                          TRTVEC= 14              ;;TRACE TRAP
000014                          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                          PWRVEC= 24              ;;POWER FAIL
000030                          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                          TRAPVEC=34              ;;"TRAP" TRAP
000060                          TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                          TPVEC=  64              ;;TTY PRINTER VECTOR
000240                          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR

177546                                  ABASE=  177546
000100                                  AVECT1= 100
000001                                  $TN=1


                                .SBTTL  ACT11 HOOKS

                                ;;**************************************************************
```

```
                                ;HOOKS REQUIRED BY ACT11
000204                                  $SVPC=.                 ;SAVE PC
000046                                  .=46
000046  003324                          $ENDAD                  ;;1)SET LOC,46 TO ADDRESS OF $ENDAD IN .$EOP
000052                                  .=52
000052  000000                          .WORD   0               ;;2)SET LOC,52 TO ZERO
000204                                  .=$SVPC                 ;; RESTORE PC
001000                                  .=1000
                                .SBTTL  APT PARAMETER BLOCK

                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
001000                                  .$X=.   ;;SAVE CURRENT LOCATION
000024                                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
000044                                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
001000                                  .=.,$X  ;;RESET LOCATION COUNTER
                                ;;******************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001174                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000074                  $TSTM:  .WORD   60.     ;;RUN TIM OF LONGEST TEST
001006  000170                  $PASTM: .WORD   120.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000170                  $UNITM: .WORD   120.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000031                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                ;SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  001100  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                         .WORD   0
001102     000          001102  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103     000          001103  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          001104  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          001106  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          001110  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          001112  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114     000          001114  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115     001          001115  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          001116  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          001120  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          001122  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          001124  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          001126  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0                       ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134     000          001134  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135     000          001135  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          001140  $WR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          001142  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          001144  $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          001146  $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          001150  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          001152  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154     000          001154  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155     002          001155  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156     012          001156  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157     000          001157  $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          001160  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001162  000000          001162  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001164  177607  000377  001164  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001170     077          001170  $QUES:  .ASCII  /?/              ;;QUESTION MARK
001171     015          001171  $CRLF:  .ASCII  <15>             ;;CARRIAGE RETURN
001172  000012          001172  $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*****************************************************************
                                .EVEN
001174                  001174  $MAIL:                          ;;APT MAILBOX
001174  000000          001174  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001176  000000          001176  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001200  000000          001200  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001202  000000          001202  $PASS:  .WORD   APASS           ;;PASS COUNT
001204  000000          001204  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
```

```
000000                  $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
000000                  $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
000000                  $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
                        $ETABLE:                ;;APT ENVIRONMENT TABLE
000                     $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
000                     $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                        ;*                      BITS 15-11=CPU TYPE
                        ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                        ;*                              11/70=06,PDQ=07,G=10
                        ;*                      BIT 10=REAL TIME CLOCK
                        ;*                      BIT  9=FLOATING POINT PROCESSOR
                        ;*                      BIT  8=MEMORY MANAGEMENT
000                     $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M,S, BYTE
000                     $MTYP1: .BYTE   AMTYP1  ;;MEM, TYPE,BLK#1
                        ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                        ;*                              900 NSEC CORE=001
                        ;*                              300 NSEC BIPOLAR=002
                        ;*                              500 NSEC MOS=003
000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                        ;*                        MEM,LAST ADDR,#3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
000                     $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M,S, BYTE
000                     $MTYP2: .BYTE   AMTYP2  ;;MEM,TYPE,BLK#2
000000                  $MADR2: .WORD   AMADR2  ;;MEM,LAST ADDRESS,BLK#2
000                     $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M,S,BYTE
000                     $MTYP3: .BYTE   AMTYP3  ;;MEM,TYPE,BLK#3
000000                  $MADR3: .WORD   AMADR3  ;;MEM,LAST ADDRESS,BLK#3
000                     $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M,S,BYTE
000                     $MTYP4: .BYTE   AMTYP4  ;;MEM,TYPE,BLK#4
000000                  $MADR4: .WORD   AMADR4  ;;MEM,LAST ADDRESS,BLK#4
000100                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
177546                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
000000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001256                  $ETEND:
                        .MEXIT
```

```
                        .SBTTL  ERROR POINTER TABLE

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

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


001256                  $ERRTB:

                                ;ITEM   1

001256  006754                  EM1                     ;MEMORY READ/WRITE ERROR
001260  007165                  DH1                     ;ERRPC   ADDR    WAS     S/B
001262  007324                  DT1                     ;$ERRPC,RADDR,$BDDAT,$GDDAT
001264  007364                  DF0                     ;ALL NUMBERS ARE IN OCTAL FORM

                                ;ITEM   2

001266  007005                  EM2                     ;MEMORY DATA ERROR AFTER POWER FAIL
001270  007165                  DH1                     ;ERRPC   ADDR    WAS     S/B
001272  007324                  DT1                     ;$ERRPC,RADDR,$BDDAT,$GDDAT
001274  007364                  DF0                     ;ALL NUMBERS ARE IN OCTAL FORM

                                ;ITEM   3

001276  007051                  EM3                     ;CLOCK CSR ERROR
001300  007222                  DH3                     ;ERRPC   WAS     S/B
001302  007336                  DT3                     ;$ERRPC,$BDDAT,$GDDAT
001304  007364                  DF0                     ;ALL NUMBERS ARE IN OCTAL FORM

                                ;ITEM   4

001306  007072                  EM4                     ;CLOCK ADDR ERROR
001310  007247                  DH4                     ;ERRPC   ADDR
001312  007346                  DT4                     ;$ERRPC,LKS
001314  007364                  DF0                     ;ALL NUMBERS ARE IN OCTAL FORM

                                ;ITEM   5

001316  007115                  EM5                     ;CLOCK INTERRUPT ERROR
001320  007247                  DH4
001322  007346                  DT4
001324  007364                  DF0                     ;ALL NUMBERS ARE IN OCTAL FORM

                                ;ITEM   6
```

```
                                        EM6                     ;INTERRUPT ERROR
                                        DH6                     ;TEST    ERRPC   TO      FROM
                                        DT6                     ;TSTNM,$ERRPC,TRTO,TRFRO
                                        DF0                     ;ALL NUMBERS ARE IN OCTAL FORM

                                        ;REGISTER DEFINITION, CONSTANTS, STORAGE,

001336  177546                LKS:    .WORD   ABASE           ;>NO    <;ADDRESS OF CLOCK CSR, IF DIFFERENT
                                                                ;>PATCH <;CHANGE CONTENTS OF "$BASE".
001340  000100                KVECT:  .WORD   AVECT1          ;>NO    <;VECTOR ADDRESS OF CLOCK.
001342  000102                KVECTP: .WORD   AVECT1+2        ;>PATCH <;

001344  017470                HILIM:  .WORD   017470          ;HI ADDR, CORE FOR WRITE PATTERN
001346  007370                LOLIM:  .WORD   MEMST           ;LOW ADDR CORE FOR MEM TEST.
001350  007372                LOW2:   .WORD   MEMST+2         ;NEXT ADDRESS.

001352  000000                PCOUNT: .WORD   0               ;COUNTS # OF POWER FAILS.
001354  000000                PFLAG:  .WORD   0               ;=1 INDICATES PFAIL. 0 = NO POWER FAIL
001356  000000                ADCK:   .WORD   0               ;=1 WHEN WE HAVE TESTED CLOCK ADDRESS
001360  000001                MEMWR:  .WORD   1               ;=1 WHEN MEMORY BEING WRITTEN.
001362  000000                RADDR:  .WORD   0               ;=ADDR, DURING MEM CHECK
001364  000000                BAD:    .WORD   0               ;=1 WHEN POWER DOWN BAD.
001366  000000                BLKCK:  .WORD   0
001370  000000                LWWR:   .WORD   0               ;LAST MEMORY ADDR.WRITTEN W/MEM PATTERN.
001372  000000                NOCLK:  .WORD   0               ;INDICATES WEITHER THE CLOCK SHOULD BE TESTED OR NOT.
                                                                ;=0 THEN YES;;=1 NO.USER MAY CHANGE THIS LOC.


001374  013703  001346        START:  MOV     LOLIM,R3        ;SET FOR MEM CHECK ON POWER UP.
001400  012713  177777                MOV     #-1,(3)         ;PATTERN ALL ONES
001404  010337  001370                MOV     R3,LWWR
001410  005000                        CLR     R0
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS (#CMTAG) AREA
001412  012706  001100                MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001416  005026                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001420  022706  001140                CMP     #$WR,R6 ;;DONE?
001424  001374                        BNE     .-6             ;;LOOP BACK IF NO
001426  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
001432  012737  004354  000020        MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001440  012737  000340  000022        MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001446  012737  004032  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001454  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001462  012737  006674  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001470  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
001476  005037  001160                CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001502  005037  001162                CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001506  112737  000001  001115        MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
001514  012737  001514  001106        MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001522  012737  001522  001110        MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001530  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
```

```
                                MOV     #648,@#ERRVEC   ;;SET UP ERROR VECTOR
                                MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
                                MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
                                CMP     #-1,@#SWR       ;;TRY TO REFERENCE HARDWARE SWR
                                BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
                                BR      65$             ;;BRANCH IF NO TIMEOUT
001570  012716  001576    64$:  MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001574  000002                  RTI
001576  012737  000176  001140  65$:  MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001604  012737  000174  001142        MOV     #DISPREG,DISPLAY
001612  012637  000004          66$:  MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

001616  005037  001202                CLR     $PASS           ;;CLEAR PASS COUNT
001622  132737  000200  001215        BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
001630  001403                        BEQ     67$             ;;YES,USE NON-APT SWITCH
001632  012737  001216  001140        MOV     #68$WREG,SWR    ;;NO,USE APT SWITCH REGISTER
001640                          67$:

001640  012737  003344  000024        MOV     #PDOWN,PWRVEC   ;INITIALIZE POWER FAIL VECTOR,
001646  012737  000340  000026        MOV     #340,PWRVEC+2   ;NO INTERRUPTS ALLOWED ON POWER FAIL,
001654  005037  001202                CLR     $PASS           ;CLEAR PASS COUNT
001660  005037  001352                CLR     PCOUNT          ;CLEAR POWER FAIL COUNT
001664  005037  001356                CLR     ADCR            ;CLOCK ADDR, NOT TESTED,
001670  005037  001364                CLR     BAD

                                                        ;/-PR-
001674  012746  000340                MOV     #340,-(6)       ;/PUT NEW STATUS ON STACK
001700  012746  001706                MOV     #68$,-(6)       ;/PUT RETURN ADDR, ON STACK,
001704  000002                        RTI                     ;/DO AN RTI   CPU WILL RETURN TO NEXT
                                                        ;/ADDR, WITH NEW STATUS,  IF STATUS
                                                        ;/=0 THEN INTERRUPTS ALLOWED,
001706                          68$:

                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001706  005227  177777                INC     #-1             ;;FIRST TIME?
001712  001033                        BNE     69$             ;;BRANCH IF NO
001714  104401  001762                TYPE    ,70$            ;;TYPE ASCII STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001720  005737  000042                TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
001724  001012                        BNE     71$             ;;BRANCH IF YES
001726  123727  001214  000001        CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
001734  001406                        BEQ     71$             ;;BRANCH IF YES
001736  023727  001140  000176        CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
001744  001005                        BNE     72$             ;;BRANCH I/ NO
001746  104406                        GTSWR                   ;;GET SOFT-SWR SETTINGS
001750  000403                        BR      72$
001752  112737  000001  001134  71$:  MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
001760                          72$:
001760  000410                        BR      69$             ;;GET OVER THE ASCII
                                ;;70$:  .ASCII  <CRLF>#MD11-DVKPA-A$<CRLF>
002002                          69$:
```

```
                                ;;*****************************************************************
                                ;*TEST 1         *WRITE CORE WITH MEMORY PATTERN
                                ;*
                                ;*IN THIS TEST WE WILL WRITE AVAILABLE CORE WILL ALL ONES OR ALL
                                ;*ZEROS DEPENDING ON ODD OR EVEN PASS (RESPECTIVELY).
                                ;*AFTER CORE IS WRITTEN, WE WILL VERIFY RESULTS.
                                ;*
                                ;*IF WE ARE IN THE PROCESS OF CHANGING CORE PATTERN DURING A POWER
                                ;*FAIL, ONLY THOSE LOCATIONS THAT WERE WRITTEN WILL BE CHECKED.
                                ;*
                                ;;*****************************************************************
002002  000240                  TST1:   NOP
002004  012737  000001  001160          MOV     #1,@TIMES       ;;DO 1 ITERATION
002012  112737  000001  001102          MOVB    #1,@TSTNM       ;SET TEST #1
002020  012737  000001  001200          MOV     #1,@TESTN       ;ALSO IN MAIL BOX!
002026  012737  002042  001106          MOV     #18,@LPADR      ;LOOP FOR ERROR
002034  012737  002042  001110          MOV     #18,@LPERR
002042  012737  000001  001360  18:     MOV     #1,MEMWR        ;=1 WHEN WE ARE CHANGING MEMORY
002050  013737  001346  001370          MOV     LOLIM,LMWR      ;PATTERN -- PICK UP LOW ADDRESS
002056  005001                          CLR     R1              ;PATTERN = 0 ON EVEN PASS
002060  132737  000001  001202          BITB    #1,@PASS        ;IS THIS AN EVEN PASS?
002066  001402                          BEQ     28              ;YES - CONTINUE
002070  012701  177777                  MOV     #-1,R1          ;NO - ODD PASS, CHANGE TO -1

002074  010177  177270          28:     MOV     R1,@LMWR        ;WRITE PATTERN
002100  005037  001366                  CLR     BLKCK           ;INDICATE
                                        ;PATTERN OK FOR POWER UP CLOCK,
                                        ;IF WAS = 1 THEN REVERSE PATTERN WOULD
                                        ;HAVE BEEN USED.

002104  023737  001370  001344  38:     CMP     LMWR,HILIM      ;ARE WE AT HIGH LIMIT?
002112  001411                          BEQ     58
002114  103402                          BLO     48
002116  000000                          HALT                    ;SOFT ERROR! ADDRESS IN LOLIM
                                        ;GREATER THAN ADDRESS IN HILIM
                                        ;FATAL CONDITION - DID YOU PATCH THESE
002120  000771                          BR      38              ;LOCATIONS? I HAD THEM RIGHT.
002122  010177  177242          48:     MOV     R1,@LMWR        ;WRITE PATTERN
002126  062737  000002  001370          ADD     #2,LMWR
002134  000763                          BR      38              ;LOOP BACK.

002136  005037  001360          58:     CLR     MEMWR           ;INDICATE WE'RE THROUGH WRITING
                                        ;MEMORY
002142  013703  001346                  MOV     LOLIM,R3        ;PUT LOWER LIMIT IN R3
002146  010302                          MOV     R3,R2           ;AND IN R2

                                ;NOW LETS CHECK MEM
```

```
                002150  020237  001344          68:     CMP     R2,HILIM        ;ARE WE AT HIGH ADDR?
                002154  001415                          BEQ     TST2            ;;
                002156  022201                          CMP     (2)+,R1         ;PATTERN OK?
                002160  001773                          BEQ     68              ;YES - LOOP.
                002162  010237  001362                  MOV     R2,RADDR        ;NO - RECORD ADDR.
                002166  162737  000002  001362          SUB     #2,RADDR        ;-2 = REAL ADDR.
                002174  017737  177162  001126          MOV     @RADDR,$BDDAT   ;RECORD BAD DATA.
                002202  010137  001124                  MOV     R1,$GDDAT       ;RECORD GOOD DATA.
                002206  104001                          ERROR   1               ;PROBLEM WITH MEMORY BEFORE
                                                                                ;EVEN A POWER FAIL.
                                                ;;**************************************************************
                                                ;*TEST 2         *TEST THAT THE CLOCK GIVES "B RPLY" ON "DIN" AND "DOUT"
                                                ;;**************************************************************
                002210  000004                  TST2:   SCOPE
                002212  005737  001372                  TST     NOCLK           ;SHOULD WE TEST CLOCK (=0 YES)
                002216  001402                          BEQ     48
                002220  000137  003020                  JMP     NCLK            ;=1 NO TEST OF CLOCK.
                002224                          48:
                002224  013746  000004                  MOV     ERRVEC,-(6)     ;SAVE CONTENTS OF LOC. 4.
                002230  012737  002270  000004          MOV     #18,ERRVEC      ;SET FOR BUS ERROR ON "DIN"
                002236  017737  177074  001126          MOV     @LKS,$BDDAT     ;ISSUE "B DIN L" AND CLOCK ADDRESS.
                                                                                ;WILL TRAP FROM HERE IF CLOCK DOES
                                                                                ;NOT RESPOND WITH "B RPLY L"
                002244  012737  002274  000004          MOV     #28,ERRVEC      ;SET FOR BUS ERROR ON "DOUT"
                002252  012777  000001  177056          MOV     #1,@LKS ;ISSUE "B DOUT" AND CLOCK ADDRESS.
                                                                                ;WILL TRAP FROM HERE IF CLOCK DOES
                002260  012737  000001  001356          MOV     #1,ADCK         ;NOT RESPOND WITH "B RPLY L"
                002266  000403                          BR      38
                002270  104004                  18:     ERROR   4               ;CLOCK DID NOT RESPOND WITH "B RPLY L"
                                                                                ;WHEN ISSUED ADDRESS AND "B DIN L"
                                                                                ;CHECK ADDRESS SELECTION JUMPERS
                                                                                ;AND ADDRESS ENTERED IN LOCATION "$BASE".
                002272  000401                          BR      38
                002274  104004                  28:     ERROR   4               ;CLOCK DID NOT RESPOND WITH "B RPLY L"
                                                                                ;WHEN ISSUED ADDRESS AND "B DOUT L"
                                                                                ;WE KNOW ADDRESS AND "B DIN" DID
                                                                                ;WORK SO ADDRESS MUX SHOULD BE GOOD.
                002276  012637  000004          38:     MOV     (6)+,ERRVEC     ;RESTORE CONTENTS OF LOC 4.
                                                ;;**************************************************************
                                                ;*TEST 3         *TEST THAT BITS (INTR, ENBL) CAN BE SET AND CLEARED
                                                ;;**************************************************************
                002302  000004                  TST3:   SCOPE
```

```
                                
002304  005737  001356        18$:    TST     ADCK            ;DID CLOCK ADDRESS TEST PASS?
002310  001002                        BNE     2$              ;YES - DO THIS TEST.
002312  000000                        HALT                    ;NO - HALT! WE CANNOT PROCEED IF
002314  000773                        BR      1$              ;CLOCK CAN NOT BE ADDRESSED, PLEASE
                                                              ;LOOP ON LAST TEST OR FIX PROBLEM.

002316  005037  001354        2$:     CLR     PFLAG           ;CLEAR POWER FAIL FLAG

                                                              ;/-PR-
002322  012746  000340                MOV     #340,-(6)       ;/PUT NEW STATUS ON STACK
002326  012746  002334                MOV     #6$,-(6)        ;/PUT RETURN ADDR, ON STACK.
002332  000002                        RTI                     ;/DO AN RTI   CPU WILL RETURN TO NEXT
                                                              ;/ADDR, WITH NEW STATUS.  IF STATUS
                                                              ;/=0 THEN INTERRUPTS ALLOWED.

002334                          6$:


002334  052777  000100  176774        BIS     #BIT6,@LKS      ;SET INTR, ENABLE
002342  017737  176770  001126        MOV     @LKS,#BDDAT     ;READ CSR.
002350  032777  000100  176760        BIT     #BIT6,@LKS      ;DID IT SET?
002356  001010                        BNE     3$              ;YES CONTINUE
002360  005737  001354                TST     PFLAG           ;DID A POWER FAIL OCCUR?
002364  001354                        BNE     2$              ;YES - REDO TEST.

002366  012737  000100  001124        MOV     #100,#GDDAT     ;RECORD EXPECTED.

002374  104003                        ERROR   3               ;INTERRUPT ENABLE BIT (BIT06)
                                                              ;FAILED TO SET.
                                                              ;       NOTE
                                                              ;A DISCREPANCY COULD OCCUR IN ERROR
                                                              ;TYPEOUT SINCE S/B = 100 BUT IF
                                                              ;READY FLAG WAS SET BAD DATA WOULD
                                                              ;EQUAL 200 DISPLAYING READY
002376  000420                        BR      TST4            ;;

002400  005037  001124        3$:     CLR     #GDDAT          ;CLEAR S/B.
002404  005077  176726                CLR     @LKS            ;CLEAR CLOCK CSR.
002410  017737  176722  001126        MOV     @LKS,#BDDAT     ;READ CSR
002416  032737  000100  001126        BIT     #BIT6,#BDDAT    ;DID IT CLEAR.
002424  001004                        BNE     4$              ;NO - REPORT AN ERROR.
002426  005737  001354                TST     PFLAG           ;YES - MAKE SURE NO POWER FAIL OCCURRED.
002432  001331                        BNE     2$
002434  000401                        BR      TST4            ;;

002436  104003                  4$:   ERROR   3               ;INTERRUPT ENABLE BIT FAILED TO
                                                              ;CLEAR (BIT06).
                                                              ;       NOTE:
                                                              ;IN ERROR TYPEOUT BIT7 MAY ALSO
                                                              ;BE SET, THIS IS NOT A BIT7 ERROR.
                                                              ;WE CAN NOT PREDICT WHEN BIT7
                                                              ;WILL SET OR CLEAR AND DO NOT
                                                              ;REFLECT IT IN S/B.
```

```
                        ;;*******************************************************************
                        ;*TEST 4        *TEST THAT THE CLOCK READY FLAG WILL SET AND THE CLOCK INTERRUPTS
                        ;;*******************************************************************
002440  000004          TST4:   SCOPE

002442  005037  001354  10:      CLR     PFLAG           ;CLEAR POWER FAIL FLAG,
002446                  20:

                                                        ;/-PR-
002446  012746  000340          MOV     #340,-(6)       ;/PUT NEW STATUS ON STACK
002452  012746  002460          MOV     #648,-(6)       ;/PUT RETURN ADDR, ON STACK,
002456  000002                  RTI                     ;/DO AN RTI   CPU WILL RETURN TO NEXT
                                                        ;/ADDR, WITH NEW STATUS.  IF STATUS
                                                        ;/=0 THEN INTERRUPTS ALLOWED,
002460                  648:

002460  012777  000100  176650      MOV     #BIT6,@LKS      ;SET INTERRUPT ENABLE ON CLOCK,
002466  005002                  CLR     R2              ;KEEP TIME-OUT COUNTER,

002470  105777  176642  40:      TSTB    @LKS            ;DID READY FLAG SET?
002474  100416                  BMI     50              ;YES - NEXT CHECK,
002476  005202                  INC     R2              ;NO - DID WE ALLOW ENOUGH TIME?
002500  001373                  BNE     40              ;NO - LOOP,
002502  005737  001354          TST     PFLAG           ;YES - DID A POWER FAIL OCCUR?
002506  001355                  BNE     10              ;YES - DO TEST OVER, POWER FAIL
                                                        ;INITIALIZED CLOCK
002510  017737  176622  001126      MOV     @LKS,@BDDAT     ;RECORD CLOCK CSR,
002516  012737  000300  001124      MOV     #300,@GDDAT     ;RECORD S/B

002524  104003                  ERROR   3               ;CLOCK READY FLAG FAILED TO SET,

002526  000430                  BR      TST5            ;;

002530  012777  002564  176602  50:      MOV     #668,@KVECT     ;SET UP FOR CLOCK INTERRUPT,

                                                        ;/-PR-
002536  012746  000000          MOV     #0,-(6) ;/PUT NEW STATUS ON STACK
002542  012746  002550          MOV     #658,-(6)       ;/PUT RETURN ADDR, ON STACK,
002546  000002                  RTI                     ;/DO AN RTI   CPU WILL RETURN TO NEXT
                                                        ;/ADDR, WITH NEW STATUS.  IF STATUS
                                                        ;/=0 THEN INTERRUPTS ALLOWED,
002550                  658:

002550  000240          NOP                             ;INTERRUPT FROM HERE
002552  005737  001354          TST     PFLAG           ;NO INTERRUPT, DID A POWER FAIL OCCUR?
002556  001331                  BNE     10              ;YES - DO TEST OVER, POWER FAIL
                                                        ;INITIALIZED CLOCK,

002560  104005                  ERROR   5               ;CLOCK FAILED TO INTERRUPT
                                                        ;READY FLAG DID SET, COULD YOU HAVE
                                                        ;A JUMPER WRONG?
002562  000402                  BR      70
```

```
                                ;*TEST THAT THE CLOCK READY FLAG WILL SET AND THE CLOCK INTERRUPTS

002564  062706  000004          68$:    ADD     #4,R6           ;UPDATE STACK POINTER
002570  013777  001340  176542  78$:    MOV     KVECT,@KVECT    ;RESTORE VECTOR TO LOOK FOR
002576  062777  000002  176534          ADD     #2,@KVECT       ;ILLEGAL INTERRUPTS.
002604  005077  176526                  CLR     @LKS            ;CLEAR CSR.

                                ;;**********************************************************************
                                ;*TEST 5         *TEST THAT WE CAN CLEAR READY FLAG

                                ;*
                                ;*IN THIS TEST WE'RE GONNA TRY TO CLEAR THE READY BIT (BIT07)
                                ;*TO DO THIS WE'RE GONNA CLEAR IT, STALL TIME ALLOWING
                                ;*IT TO SET AGAIN, THEN CLEAR IT TESTING THAT IT ACTUALLY CLEARED.
                                ;*
                                ;;**********************************************************************
002610  000004          TST5:   SCOPE

002612  005037  001354  1$:     CLR     PFLAG           ;CLEAR POWER FAIL FLAG.

002616  005077  176514          CLR     @LKS            ;CLEAR CLOCK
002622  005002                  CLR     R2              ;CLEAR TIME OUT FLAG.

002624  105777  176506  2$:     TSTB    @LKS            ;DID READY FLAG SET?
002630  100415                  BMI     3$
002632  005302                  DEC     R2              ;TIME OUT (ALL ALLOWED TIME EXPIRE?)
002634  001373                  BNE     2$              ;NO - LOOP
002636  005737  001354          TST     PFLAG           ;YES - DID A POWER FAIL OCCUR?
002642  001363                  BNE     1$              ;YES - DO TEST OVER.

002644  017737  176466  001126  3$:     MOV     @LKS,$BDDAT     ;RECORD CSR.
002652  012737  000200  001124          MOV     #BIT7,$GDDAT    ;RECORD S/B.

002660  104003                  ERROR   3               ;CLOCK READY FLAG FAILED TO SET.

002662  000420                  BR      TST6            ;;

002664  005077  176446  38$:     CLR     @LKS            ;THE FLAG JUST SET NOW WE WILL
002670  005037  001124          CLR     $GDDAT          ;TRY TO CLEAR IT.
002674  017737  176436  001126  4$:     MOV     @LKS,$BDDAT     ;READ THE CSR IF ZERO - OK.
002702  001004                  BNE     48              ;IF NOT WE'LL REPORT AN ERROR.
002704  005737  001354          TST     PFLAG           ;OF COURSE WE'VE GOT TO CHECK
                                ;THE POWER FAIL FLAG, SINCE A
                                ;POWER FAIL COULD HAVE CLEARED
                                ;THE FLAG!
002710  001340                  BNE     1$              ;IF SO REDO TEST
002712  000404                  BR      TST6            ;;

002714  005737  001354  48$:     TST     PFLAG           ;CHECK POWER FAIL FLAG-IF POWER FAIL OCCURED
002720  001334                  BNE     1$              ;DURING TEST AN ERROR COULD RESULT.
                                ;IF NO POWER FAIL OCCURED REPORT ERROR.

002722  104003                  ERROR   3               ;COULD NOT CLEAR CLOCK
                                ;READY FLAG (BIT07)
```

```
                        ;;*****************************************************************
                        ;*TEST 6         *TEST THAT THE CLOCK DOESN'T INTR, WHEN INTR, ENABLE CLEAR
                        ;;*****************************************************************
002724  000004           TST6:   SCOPE
002726  012737  000020  001160          MOV     #20,@#TIMES     ;;DO 20 ITERATIONS

002734  005037  001354   1$:     CLR     PFLAG           ;CLEAR POWER FAIL FLAG.

002740  012777  002776  176372          MOV     #3$,@#KVECT     ;SET FOR ILLEGAL INTERRUPT.


                                                        ;/-PR-
002746  012746  000000                  MOV     #0,-(6) ;/PUT NEW STATUS ON STACK
002752  012746  002760                  MOV     #64$,-(6)       ;/PUT RETURN ADDR, ON STACK,
002756  000002                          RTI                     ;/DO AN RTI   CPU WILL RETURN TO NEXT
                                                                ;/ADDR, WITH NEW STATUS,  IF STATUS
                                                                ;/=0 THEN INTERRUPTS ALLOWED.
002760                          64$:

002760  005002                          CLR     R2
002762  105202                  2$:     INCB    R2              ;STALL
002764  001376                          BNE     2$
002766  005737  001354                  TST     PFLAG           ;NO INTERRUPT - DID POWER FAIL
002772  001360                          BNE     1$              ;YES - REDO TEST.
002774  000403                          BR      4$

002776  062706  000004          3$:     ADD     #4,R6           ;UPDATE STACK POINTER.

003002  104005                          ERROR   5               ;CLOCK INTERRUPTED WHEN
                                                                ;BIT6 (INTR ENBL) CLEAR.
003004  013777  001340  176326  4$:     MOV     KVECT,@#KVECT   ;RESTORE VECTOR
003012  062777  000002  176320          ADD     #2,@#KVECT

003020  012737  000006  001102  NCLK:   MOV     #6,@#TSTNM      ;ENTRY POINT IF CLOCK NOT TESTED.


                        ;;*****************************************************************
                        ;*TEST 7         *VERIFY MEMORY LOCATIONS WRITTEN IN TEST1
                        ;*
                        ;*IN THIS MACRO WE ARE GOING TO VERIFY THE CONTENTS OF
                        ;*MEMORY AGAIN, WE ARE GOING TO DEVOTE MOST OF OUR TIME
                        ;*OF THE TOTAL TEST TIME HERE HOPING WE CAN CATCH A POWER
                        ;*FAIL WHILE WE'RE READING MEMORY.
                        ;*
                        ;;*****************************************************************
003026  000004           TST7:   SCOPE
003030  012737  002000  001160          MOV     #002000,@#TIMES ;;DO 002000 ITERATIONS


003036  005037  001354          1$:     CLR     PFLAG
003042  005001                          CLR     R1              ;ZERO PATTERN IS EVEN PASS.
003044  132737  000001  001202          BITB    #BIT0,@#PASS    ;EVEN PASS?
003052  001402                          BEQ     1$              ;YES - CONTINUE
```

```
                                                MOV     #-1,R1          ;NO - PATTERN ALL ONES.
                                        18:     MOV     LOLIM,R2        ;PICK-UP LOW ADDRESS
                                        28:     CMP     R2,HILIM        ;ARE WE AT HIGH ADDRESS?
                                                BEQ     TST10           ;;
                                                CMP     (2)+,R1         ;PATTERN OK?
                                                BEQ     28              ;YES - LOOP.
                                                MOV     R2,RADDR        ;NO - RECORD ADDR.
                                                SUB     #2,RADDR        ;-2 = READ ADDR.
                                                MOV     @RADDR,$BDDAT   ;GET CONTENTS
                                                MOV     R1,$GDDAT       ;GOOD DATA.
                                                ERROR   1               ;PROBLEM WITH MEMORY, DATA
                                                                        ;READ NOT DATA DEPOSITED.
                                ;;**************************************************************
                                ;*TEST 10        *END OF TESTS
                                ;;**************************************************************
                                TST10:  SCOPE


                                .SBTTL  END OF PASS ROUTINE

                                ;;**************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO LOOP

                                $EOP:
                                        INC     $LKCK
                                        CLR     $TSTNM          ;;ZERO THE TEST NUMBER
                                        CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
                                        INC     $PASS           ;;INCREMENT THE PASS NUMBER
                                        BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
                                        DEC     (PC)+           ;;LOOP?
                                $EOPCT: .WORD   1
                                        BGT     $DOAGN          ;;YES
                                        MOV     (PC)+,0(PC)+    ;;RESTORE COUNTER
                                $ENDCT: .WORD   1
                                        $EOPCT

                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <200>$END PASS  @
                                64$:
                                        MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                        TYPE    ,67$            ;;TYPE ASCIZ STRING
                                        BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  @  POWER FAILS  @
                                66$:
                                        MOV     PCOUNT,-(SP)    ;;SAVE PCOUNT FOR TYPEOUT
                                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
826  003254  005737  001372            TST     NOCLK                   ;WAS CLOCK TESTED?
827  003260  001415                    BEQ     1$                      ;YES SKIP TYPEOUT.
828                                                                     ;NO INDICATE CLOCK NOT TESTED.
829  003262  104401  003270            TYPE    ,698$           ;;TYPE ASCIZ STRING
830  003266  000412                    BR      68$             ;;GET OVER THE ASCIZ
831                              ;;698$: .ASCIZ  @  CLOCK NOT TESTED@
832  003314                      68$:
833  003314                      1$:
834  003314  013700  000042      $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
835  003320  001405                      BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
836  003322  000005                      RESET                   ;;CLEAR THE WORLD
837  003324  004710              $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
838  003326  000240                      NOP                     ;;SAVE ROOM
839  003330  000240                      NOP                     ;;FOR
840  003332  000240                      NOP                     ;;ACT11
841  003334                      $DOAGN:
842  003334  000137                      JMP     @(PC)+          ;;RETURN
843  003336  002002              $RTNAD: .WORD   LOOP
844  003340    377     377     000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
845          003344                      .EVEN
846
847                                      ;*
848                                      ;*THIS ROUTINE HANDLES POWER DOWNS
849                                      ;*
850
851  003344  005737  001364      PDOWN:  TST     BAD             ;HAVE WE HAD ENOUGH TIME TO
852  003350  001402                      BEQ     PD2             ;DOWN ERROR ON PREVIOUS POWER FAIL?
853  003352  000000              PDBAD:  HALT                    ;YES - ERROR POWER DOWN ROUTINE
854  003354  000773                      BR      PDOWN           ;DID NOT HAVE TIME TO COMPLETE
855                                                              ;SEQUENCE BEFORE CPU STOPPED.
856
857  003356  005237  001364      PD2:    INC     BAD             ;SET POWER DOWN "NOT ENOUGH TIME" FLAG
858
859  003362  005737  001366              TST     BLKCK           ;IN PROCESS OF CHANGING PATTERN?
860  003366  001410                      BEQ     1$              ;NO GET OUT.
861
862  003370  027777  175752  175752      CMP     @LOLIN,@LOW2    ;PATTERN =?
863  003376  001404                      BEQ     1$
864  003400  005037  001366              CLR     BLKCK           ;NO-HAD WRITTEN NEW PATTERN-BUT NOT
865  003404  005237  001360              INC     NEWWR           ;HAD TIME ENOUGH TIME TO CLR BLKCK
866  003410                      1$:
867  003410  005077  175722              CLR     @LKS            ;STOP CLOCK,
868  003414  010046                      MOV     R0,-(6)         ;SAVE GENERAL REGISTERS
869  003416  010146                      MOV     R1,-(6)
870  003420  010246                      MOV     R2,-(6)
871  003422  010346                      MOV     R3,-(6)
872  003424  010446                      MOV     R4,-(6)
873  003426  010546                      MOV     R5,-(6)
874  003430  013746  001124              MOV     $GDDAT,-(6)
875  003434  013746  001126              MOV     $BDDAT,-(6)
876  003440  013746  001362              MOV     RADDR,-(6)
877  003444  010637  003504              MOV     R6,$R6
878  003450  012700  000024              MOV     #20.,R0         ;NOW WE WILL SIMULATE DOING
879  003454  013737  001124  001124  2$: MOV     $GDDAT,$GDDAT   ;60. INSTRUCTIONS, THE POWER DOWN
```

```
                                        DEC     R0              ;SEQUENCE SHOULD ALLOW US AT LEAST
                                        BNE     2$              ;2 MILLISECONDS OF TIME TO DO A POWER
                                                                ;FAIL ROUTINE,HOWEVER,ONE MUST
                                                                ;BE WATCHFUL THE THE SYSTEM DOESN'T
                                                                ;DO MEMORY REFRESH CYCLES ON OTHER
                                                                ;MEMORIES DURING THIS TIME,THUS STEALING
                                                                ;TIME FROM THE POWER DOWN ROUTINE.
                                        MOV     @PUP,PWRVEC     ;POINT TO POWER UP ROUTINE.
                                        CLR     BAD             ;INDICATE WE HAD ENOUGH TIME

                                3$:     HALT                    ;HALT AT END OF POWER DOWN ROUTINE.
                                        BR      3$              ;ONE MUST DO THIS OR THERE IS A CHANCE
                                                                ;THAT THE INSTRUCTION OR MEMORY LOCATION
                                                                ;REFERENCED BY THAT INSTRUCTION
                                                                ;MAY BE DESTROYED WHEN POWER GOES AWAY.

                                SR6:    0                       ;SAVE STACK

                                        ;*
                                        ;*THIS ROUTINE HANDLES POWER UPS.
                                        ;*
                                PUP:    MOV     SR6,R6          ;RESTORE STACK
                                        TST     BAD             ;POWER DOWN OK?
                                        BNE     PDBAD           ;NO - GO THERE.
                                        MOV     @PUPBD,PWRVEC   ;SET UP FOR BAD POWER DOWN
                                                                ;BEFORE WE CAN POWER UP.
                                        INC     PFLAG           ;INDICATE POWER FAIL.


                                PU2:    CMP     PFLAG,#5        ;HAVE WE DONE TO MANY POWER
                                        BNE     PD3             ;UPS WITHOUT DOING COMPLETE ONES?
                                        MOV     @PD2,PWRVEC     ;STAY HERE FROM NOW ON ON POWER SEQUENCES
                                                                ;FORCE OPERATOR TO RESTART PROGRAM.
                                        HALT                    ;TOO MANY POWER FAILS OCCURRED BEFORE
                                        BR      PU2             ;ONE COULD BE COMPLETED 5 * MAX.

                                PD3:    MOV     @PDOWN,PWRVEC   ;SET UP FOR POWER DOWN.


                                        .SBTTL  MEMORY TEST ON POWER UP.

                                        CLR     @GDDAT          ;PATTERN ZERO ON EVEN PASS
                                        BITB    #1,@PASS        ;EVEN PASS
                                        BEQ     1$              ;YES - CONTINUE
                                        MOV     #-1,@GDDAT      ;NO - LOOK FOR ALL ONES.
                                1$:     TST     BLKCK           ;WERE WE STARTING TO WRITE MEM?
                                        BEQ     2$              ;NO - CONTINUE
                                        COM     @GDDAT          ;YES - USE OPPOSITE PATTERN.
                                        MOV     HILIM,R5        ;PICK UP HIGH ADDR.
                                        MOV     LOLIM,R4        ;PICK UP LO ADDR
                                        BR      CK1
                                2$:     MOV     LOLIM,R4        ;PICK UP LOW ADDR.
                                        MOV     HILIM,R5        ;PICK UP HIGH ADDR.
                                        TST     MEMWR           ;WERE WE CHANGING PATTERN?
```

```
                                                BEQ     CK1             ;NO - OK
                                                MOV     LMWR,R5         ;YES - R3 WILL HAVE LAST ADDR.
                                        CK1:    CMP     R4,R5           ;DONE CHECK?
                                                BEQ     CKEND           ;YES END

                                                CMP     (4)+,@GDDAT     ;MEMORY OK
                                                BEQ     CK1             ;YES - CHECK NEXT ADDR.
                                                MOV     -2(4),@BDDAT    ;NO - GET FAILING PATTERN
                                                MOV     R4,RADDR        ;GET ADDR
                                                SUB     #2,RADDR        ;READ = -2

                                                ERROR   2               ;READ PATTERN ERROR MEMORY
                                                                        ;CONTAINED OTHER THAN WRITTEN.
                                                BR      CK1

                                        CKEND:  INC     PCOUNT          ;UPDATE POWER FAIL COUNT.

                                                .SBTTL  CLOCK CHECK AFTER POWER UP

                                                TST     ADCK            ;HAS CLOCK BEEN TESTED?
                                                BEQ     PUEND           ;NO - LETS NOT TEST IT HERE!

                                                MOV     @LKS,@BDDAT     ;READ CSR
                                                CLR     @GDDAT          ;EXPECT ZERO ALTHOUGH BIT7
                                                                        ;MAY BE SET, WE WON'T LOOK
                                                                        ;AT IT.
                                                BIT     @BIT6,@BDDAT    ;IS BIT6 SET?
                                                BEQ     PUEND           ;NO - GOOD.
                                                ERROR   3               ;BIT6 OF CLOCK CSR SET AFTER
                                                                        ;POWER UP - INIT FAILED TO
                                                                        ;CLEAR IT.

                                        PUEND:
                                                MOV     (6)+,RADDR      ;RESTORE EVERYTHING THAT WAS SAVED BEFORE POWER FAIL.
                                                MOV     (6)+,@BDDAT
                                                MOV     (6)+,@GDDAT
                                                MOV     (6)+,R5
                                                MOV     (6)+,R4
                                                MOV     (6)+,R3
                                                MOV     (6)+,R2
                                                MOV     (6)+,R1
                                                MOV     (6)+,R0
                                                BEQ     2$              ;IF R0=0 THEN NO MESSAGE BEING TYPED.
                                                DECB    R0              ;IF MESSAGE BEING TYPED,BACK UP FOR LAST CHARACTOR.
                                        2$:
                                                TST     R0              ;IF NO MESSAGE BEING TYPED OUT WE
                                                                        ;WILL TYPE OUT 'PF-OK'.
                                                BNE     1$
                                                TYPE    ,65$            ;;TYPE ASCIZ STRING
                                                BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <200>/PF-OK/
                                        64$:
                                        1$:     RTI

                                        PUPBD:  HALT                    ;POWER DOWN OCCURRED BEFORE
```

```
                004030  000776                          BR      PUPBD           ;WE COULD SUCCESSFULLY START
                                                                                ;A POWER UP
                                                
                                                ,SBTTL  ERROR HANDLER ROUTINE
                                                
                                                ;;**********************************************************************
                                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                                ;*AND GO TO @ERRTYP ON ERROR
                                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                                ;*SW15=1         HALT ON ERROR
                                                ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                                ;*SW10=1         BELL ON ERROR
                                                ;*SW09=1         LOOP ON ERROR
                                                ;*CALL
                                                ;*      ERROR   N       ;;ERROR=ENT AND N=ERROR ITEM NUMBER
                                                
                004032                          $ERROR:
                004032  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
                004034  105237  001103          7$:     INCB    @ERFLG          ;;SET THE ERROR FLAG
                004040  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
                004042  013777  001102  175072          MOV     @TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
                004050  032777  002000  175062          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
                004056  001402                          BEQ     1$              ;;NO - SKIP
                004060  104401  001164                  TYPE    ,@BELL          ;;RING BELL
                004064  005237  001112          1$:     INC     @ERTTL          ;;COUNT THE NUMBER OF ERRORS
                004070  011637  001116                  MOV     (SP),@ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
                004074  162737  000002  001116          SUB     #2,@ERRPC
                004102  117737  175010  001114          MOVB    @@ERRPC,@ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
                004110  032777  020000  175022          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
                004116  001004                          BNE     20$             ;;SKIP TYPEOUTS
                004120  004737  004220                  JSR     PC,@ERRTYP      ;;GO TO USER ERROR ROUTINE
                004124  104401  001171                  TYPE    ,@CRLF
                004130                          20$:
                004130  122737  000001  001214          CMPB    @APTENV,@ENV    ;;RUNNING IN APT MODE
                004136  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
                004140  113737  001114  004152          MOVB    @ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
                004146  004737  006364                  JSR     PC,@ATY4        ;;REPORT FATAL ERROR TO APT
                004152     000                  21$:    .BYTE   0
                004153     000                          .BYTE   0
                004154  000777                  22$:    BR      22$             ;;APT ERROR LOOP
                004156  005777  174756          2$:     TST     @SWR            ;;HALT ON ERROR
                004162  100002                          BPL     3$              ;;SKIP IF CONTINUE
                004164  000000                          HALT                    ;;HALT ON ERROR!
                004166  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
                004170  032777  001000  174742  3$:     BIT     @BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
                004176  001402                          BEQ     4$              ;;BR IF NO
                004200  013716  001110                  MOV     @LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
                004204  005737  001162          4$:     TST     @ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
                004210  001402                          BEQ     5$              ;;BR IF NONE
                004212  013716  001162                  MOV     @ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
                004216                          5$:
                004216  000002                          RTI                     ;;RETURN
                                                ,SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
```

```
                                        ;;**********************************************************************
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;*ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

004220                                  $ERRTYP:
004220  104401  001171                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
004224  010046                                  MOV     R0,-(SP)        ;;SAVE R0
004226  005000                                  CLR     R0              ;;PICKUP THE ITEM INDEX
004230  153700  001114                          BISB    @#$ITEMB,R0
004234  001004                                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                        ;;TYPE THE PC OF THE ERROR
004236  013746  001116                          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
004242  104402                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
004244  000426                                  BR      6$              ;;GET OUT
004246  005300                          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
004250  006300                                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
004252  006300                                  ASL     R0
004254  006300                                  ASL     R0
004256  062700  001256                          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
004262  012037  004272                          MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
004266  001404                                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
004270  104401                                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
004272  000000                          2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
004274  104401  001171                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
004300  012037  004310                  3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
004304  001404                                  BEQ     5$              ;;SKIP TYPEOUT IF 0
004306  104401                                  TYPE                    ;;TYPE THE "DATA HEADER"
004310  000000                          4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
004312  104401  001171                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
004316  011000                          5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
004320  001004                                  BNE     7$              ;;GO TYPE THE DATA
004322  012600                          6$:     MOV     (SP)+,R0        ;;RESTORE R0
004324  104401  001171                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
004330  000207                                  RTS     PC              ;;RETURN
004332                                  7$:
004332  013046                                  MOV     0(R0)+,-(SP)    ;;SAVE 0(R0)+ FOR TYPEOUT
004334  104402                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
004336  005710                                  TST     (R0)            ;;IS THERE ANOTHER NUMBER?
004340  001770                                  BEQ     6$              ;;BR IF NO
004342  104401  004350                          TYPE    ,8$             ;;TYPE TWO(2) SPACES
004346  000771                                  BR      7$              ;;LOOP
004350  020040      000               8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
004354                                  .EVEN
                                        .SBTTL  SCOPE HANDLER ROUTINE

                                        ;;**********************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS, IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
```

```
                                        ;*SW11=1        INHIBIT ITERATIONS
                                        ;*SW09=1        LOOP ON ERROR
                                        ;*SW08=1        LOOP ON TEST IN SWR<7:0>
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT
                                        $SCOPE:
                                                CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
                                                CKSWR
004354  104407                          18:     BIT     #BIT14,@#SWR    ;;LOOP ON PRESENT TEST?
004356  001114                                  BNE     $OVER           ;;YES IF SW14=1
                                        ;@@@@@START OF CODE FOR THE XOR TESTER@@@@@
004360  000416                          $XTSTR: BR      .+40            ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
004362  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
004366  012737  004416  000004                  MOV     #.+50,@#ERRVEC  ;;SET FOR TIMEOUT
004374  005737  177060                          TST     @#177060        ;;TIME OUT ON XOR?
004400  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
004404  000463                                  BR      $VLAD           ;;GO TO THE NEXT TEST
004406  022626                          .+50:   CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
004410  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
004414  000423                                  BR      .-70            ;;LOOP ON THE PRESENT TEST
004416                                  .+52:;@@@@@END OF CODE FOR THE XOR TESTER@@@@@
004416  032777  000400  174504                  BIT     #BIT08,@#SWR    ;;LOOP ON SPEC. TEST?
004424  001404                                  BEQ     .+20            ;;BR IF NO
004426  127737  174476  001102                  CMPB    @#SWR,@#TSTNM   ;;ON THE RIGHT TEST?    SWR<7:0>
004434  001465                                  BEQ     $OVER           ;;BR IF YES
004436  105737  001103                  .+42:   TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
004442  001421                                  BEQ     .+44            ;;BR IF NO
004444  123737  001115  001103                  CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
004452  101015                                  BHI     .+34            ;;BR IF NO
004454  032777  001000  174446                  BIT     #BIT09,@#SWR    ;;LOOP ON ERROR?
004462  001404                                  BEQ     .+10            ;;BR IF NO
004464  013737  001110  001106          .+60:   MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
004472  000446                                  BR      $OVER
004474  105037  001103                  .+50:   CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
004500  005037  001160                          CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
004504  000415                                  BR      .+32            ;;ESCAPE TO THE NEXT TEST
004506  032777  004000  174414          .+40:   BIT     #BIT11,@#SWR    ;;INHIBIT ITERATIONS?
004514  001011                                  BNE     .+24            ;;BR IF YES
004516  005737  001202                          TST     $PASS           ;;IF FIRST PASS OF PROGRAM
004522  001406                                  BEQ     .+16            ;;      INHIBIT ITERATIONS
004524  005237  001104                          INC     $ICNT           ;;INCREMENT ITERATION COUNT
004530  023737  001160  001104                  CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
004536  002024                                  BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
004540  012737  000001  001104  .+16:   MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
004546  013737  004634  001160          MOV     $NXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
004554  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
004560  113737  001102  001200          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
004566  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
004572  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
004576  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
004602  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
004610  013777  001102  174314  $OVER:  MOV     $TSTNM,@$DISPLAY ;;DISPLAY TEST NUMBER
```

```
1150  004626  013716  001106            MOV     @LPADR,(SP)     ;;FUDGE RETURN ADDRESS
1151  004632  000002                    RTI                     ;;FIXES PS
1152  004634  000020                    $MXCNT: 20              ;;MAX. NUMBER OF ITERATIONS
1153                                    .SBTTL  TTY INPUT ROUTINE
1154
1155                                    ;;**********************************************************************
1156                                    .ENABL  LSB
1157
1158                                    ;;**********************************************************************
1159                                    ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE,
1160                                    ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1161                                    ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1162                                    ;*WHEN OPERATING IN TTY FLAG MODE,
1163  004636  022737  000176  001140    $CKSWR: CMP     @SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
1164  004644  001074                            BNE     15$             ;;BRANCH IF NO
1165  004646  105777  174272                    TSTB    @$TKS           ;;CHAR THERE?
1166  004652  100071                            BPL     15$             ;;IF NO, DON'T WAIT AROUND
1167  004654  117746  174266                    MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
1168  004660  042716  177600                    BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
1169  004664  022726  000007                    CMP     #7,(SP)+        ;;IS IT A CONTROL G?
1170  004670  001062                            BNE     15$             ;;NO, RETURN TO USER
1171  004672  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
1172  004700  001456                            BEQ     15$             ;;BRANCH IF YES
1173
1174  004702  104401  005363                    TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
1175  004706  104401  005370            $GTSWR: TYPE    ,$NSWR          ;;TYPE CURRENT CONTENTS
1176  004712  013746  000176                    MOV     @SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
1177  004716  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1178  004720  104401  005401                    TYPE    ,$NNEW          ;;PROMPT FOR NEW SWR
1179  004724  005046                    19$:    CLR     -(SP)           ;;CLEAR COUNTER
1180  004726  005046                            CLR     -(SP)           ;;THE NEW SWR
1181  004730  105777  174210            7$:     TSTB    @$TKS           ;;CHAR THERE?
1182  004734  100375                            BPL     7$              ;;IF NOT TRY AGAIN
1183
1184  004736  117746  174204                    MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
1185  004742  042716  177600                    BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1186
1187
1188
1189  004746  021627  000025            9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
1190  004752  001005                            BNE     10$             ;;BRANCK IF NOT
1191  004754  104401  005356                    TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
1192  004760  062706  000006            20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
1193  004764  000757                            BR      19$             ;;LET'S TRY IT AGAIN
1194
1195
1196  004766  021627  000015            10$:    CMP     (SP),#15        ;;IS IT A <CR>?
1197  004772  001022                            BNE     16$             ;;BRANCH IF NO
1198  004774  005766  000004                    TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
1199  005000  001403                            BEQ     11$             ;;BRANCH IF YES
1200  005002  016677  000002  174130            MOV     2(SP),@SWR      ;;SAVE NEW SWR
1201  005010  062706  000006            11$:    ADD     #6,SP           ;;CLEAR UP STACK
1202  005014  104401  001171            14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
1203  005020  123727  001135  000001          CMPB    @$INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
```

```
1204  005026  001003                        BNE     15$             ;;BRANCH IF NOT
1205  005030  012777  000100  174106        MOV     #100,@#TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
1206  005036  000002                10$:    RTI                     ;;RETURN
1207  005040  004737  006276        15$:    JSR     PC,$TYPEC       ;;ECHO CHAR
1208  005044  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
1209  005050  002420                        BLT     18$             ;;BRANCH IF YES
1210  005052  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
1211  005056  003015                        BGT     18$             ;;BRANCH IF YES
1212  005060  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
1213  005064  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
1214  005070  001403                        BEQ     17$             ;;BRANCH IF YES
1215  005072  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
1216  005074  006316                        ASL     (SP)            ;;   CHAR OVER TO MAKE
1217  005076  006316                        ASL     (SP)            ;;   ROOM FOR NEW ONE.
1218  005100  005266  000002        17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
1219  005104  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
1220  005110  000707                        BR      7$              ;;GET THE NEXT ONE
1221  005112  104401  001170        18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
1222  005116  000720                        BR      20$             ;;SIMULATE CONTROL-U
1223                                .DSABL  LSB
1224
1225
1226                                ;;**********************************************************************
1227                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1228                                ;*CALL:
1229                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
1230                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
1231                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
1232                                ;
1233
1234  005120  011646                $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
1235  005122  016666  000004  000002        MOV     4(SP),2(SP)     ;;SAVE THE PS
1236  005130  105777  176010        1$:     TSTB    @#TKS           ;;WAIT FOR
1237  005134  100375                        BPL     1$              ;;A CHARACTER
1238  005136  117766  176004  000004        MOVB    @#TKB,4(SP)     ;;READ THE TTY
1239  005144  042766  177600  000004        BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
1240  005152  026627  000004  000023        CMP     4(SP),#23       ;;IS IT A CONTROL-S?
1241  005160  001013                        BNE     3$              ;;BRANCH IF NO
1242  005162  105777  173756        2$:     TSTB    @#TKS           ;;WAIT FOR A CHARACTER
1243  005166  100375                        BPL     2$              ;;LOOP UNTIL IT'S THERE
1244  005170  117746  173752                MOVB    @#TKB,-(SP)     ;;GET CHARACTER
1245  005174  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1246  005200  022627  000021                CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
1247  005204  001366                        BNE     2$              ;;IF NOT DISCARD IT
1248  005206  000750                        BR      1$              ;;YES, RESUME
1249  005210  026627  000004  000140  3$:    CMP     4(SP),#140      ;;IS IT UPPER CASE?
1250  005216  002407                        BLT     4$              ;;BRANCH IF YES
1251  005220  026627  000004  000175        CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
1252  005226  003003                        BGT     4$              ;;BRANCH IF YES
1253  005230  042766  000040  000004        BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
1254  005236  000002                4$:     RTI                     ;;GO BACK TO USER
1255                                ;;**********************************************************************
1256                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1257                                ;*CALL:
```

```
                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

005240  010346                 $RDLIN: MOV     R3,-(SP)                ;;SAVE R3
005242  012703  005346          1$:     MOV     #$TTYIN,R3              ;;GET ADDRESS
005246  022703  005356          2$:     CMP     #$TTYIN+8.,R3           ;;BUFFER FULL?
005252  101405                          BLOS    4$                      ;;BR IF YES
005254  104410                          RDCHR                           ;;GO READ ONE CHARACTER FROM THE TTY
005256  112613                          MOVB    (SP)+,(R3)              ;;GET CHARACTER
005260  122713  000177          10$:    CMPB    #177,(R3)               ;;IS IT A RUBOUT
005264  001003                          BNE     3$                      ;;SKIP IF NOT
005266  104401  001170          4$:     TYPE    ,$QUES                  ;;TYPE A '?'
005272  000763                          BR      1$                      ;;CLEAR THE BUFFER AND LOOP
005274  111337  005344          3$:     MOVB    (R3),9$                 ;;ECHO THE CHARACTER
005300  104401  005344                  TYPE    ,9$
005304  122723  000015                  CMPB    #15,(R3)+               ;;CHECK FOR RETURN
005310  001356                          BNE     2$                      ;;LOOP IF NOT RETURN
005312  105063  177777                  CLRB    -1(R3)                  ;;CLEAR RETURN (THE 15)
005316  104401  001172                  TYPE    ,$LF                    ;;TYPE A LINE FEED
005322  012603                          MOV     (SP)+,R3                ;;RESTORE R3
005324  011646                          MOV     (SP),-(SP)              ;;ADJUST THE STACK AND PUT ADDRESS OF THE
005326  016666  000004  000002          MOV     4(SP),2(SP)             ;;      FIRST ASCII CHARACTER ON IT
005334  012766  005346  000004          MOV     #$TTYIN,4(SP)
005342  000002                          RTI                             ;;RETURN
005344     000          9$:     .BYTE   0                               ;;STORAGE FOR ASCII CHAR, TO TYPE
005345     000                  .BYTE   0                               ;;TERMINATOR
005346  000010          $TTYIN: .BLKB   8.                              ;;RESERVE 8 BYTES FOR TTY INPUT
005356  052536  005015     000 $CNTLU: .ASCIZ  /^U/<15><12>            ;;CONTROL "U"
005363     136  006507  000012 $CNTLG: .ASCIZ  /^G/<15><12>            ;;CONTROL "G"
005370  005015  053523  020122 $RSWR:  .ASCIZ  <15><12>/SWR = /
005376  020075     000
005401     040  047040  053505 $RNEW:  .ASCIZ  /  NEW = /
005406  036440  000040
                                        .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                        ;;**************************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
```

```
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
                                        
005412  017646  000000          $TYPOS: MOV     0(SP),-(SP)     ;;PICKUP THE MODE
005416  116637  000001  005635          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
005424  112637  005637                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
005430  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
005434  000406                          BR      $TYPON
005436  112737  000001  005635  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
005444  112737  000006  005637          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
005452  112737  000005  005634  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
005460  010346                          MOV     R3,-(SP)        ;;SAVE R3
005462  010446                          MOV     R4,-(SP)        ;;SAVE R4
005464  010546                          MOV     R5,-(SP)        ;;SAVE R5
005466  113704  005637                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
005472  005404                          NEG     R4
005474  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
005500  110437  005636                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
005504  113704  005635                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
005510  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
005514  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
005516  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
005520  000404                          BR      3$              ;;GO DO MSB
005522  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
005524  006105                          ROL     R5
005526  006105                          ROL     R5
005530  010503                          MOV     R5,R3
005532  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
005534  105337  005636                  DECB    $OMODE          ;;TYPE THIS DIGIT?
005540  100016                          BPL     7$              ;;BR IF NO
005542  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
005546  001002                          BNE     4$              ;;TEST FOR 0
005550  005704                          TST     R4              ;;SUPPRESS THIS 0?
005552  001403                          BEQ     5$              ;;BR IF YES
005554  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
005556  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
005562  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
005566  110337  005632                  MOVB    R3,$8           ;;SAVE FOR TYPING
005572  104401  005632                  TYPE    ,$8             ;;GO TYPE THIS DIGIT
005576  105337  005634          7$:     DECB    $OCNT           ;;COUNT BY 1
005602  003347                          BGT     2$              ;;BR IF MORE TO DO
005604  002402                          BLT     6$              ;;BR IF DONE
005606  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
005610  000744                          BR      2$              ;;GO DO THE LAST DIGIT
005612  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
005614  012604                          MOV     (SP)+,R4        ;;RESTORE R4
005616  012603                          MOV     (SP)+,R3        ;;RESTORE R3
005620  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
005626  012616                          MOV     (SP)+,(SP)
005630  000002                          RTI                     ;;RETURN
005632     000                  $8:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
005633     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
```

```
1366  005634  000                 $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
1367  005635  000                 $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
1368  005636  000000              $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
1369                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1370
1371                                ;;**********************************************************************
1372                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1373                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT, DEPENDING ON WHETHER THE
1374                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1375                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER, LEADING ZEROS WILL ALWAYS BE
1376                                ;*REPLACED WITH SPACES.
1377                                ;*CALL:
1378                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
1379                                ;*      TYPDS                   ;;GO TO THE ROUTINE
1380
1381  005640                        $TYPDS:
1382  005640  010046                       MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1383  005642  010146                       MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1384  005644  010246                       MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1385  005646  010346                       MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1386  005650  010546                       MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1387  005652  012746  020200               MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
1388  005656  016605  000020               MOV     20(SP),R5       ;;GET THE INPUT NUMBER
1389  005662  100004                       BPL     1$              ;;BR IF INPUT IS POS.
1390  005664  005405                       NEG     R5              ;;MAKE THE BINARY NUMBER POS.
1391  005666  112766  000055  000001       MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
1392  005674  005000                1$:    CLR     R0              ;;ZERO THE CONSTANTS INDEX
1393  005676  012703  006054               MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
1394  005702  112723  000040               MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
1395  005706  005002                2$:    CLR     R2              ;;CLEAR THE BCD NUMBER
1396  005710  016001  006044               MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
1397  005714  160105                3$:    SUB     R1,R5           ;;FORM THIS BCD DIGIT
1398  005716  002402                       BLT     4$              ;;BR IF DONE
1399  005720  005202                       INC     R2              ;;INCREASE THE BCD DIGIT BY 1
1400  005722  000774                       BR      3$
1401  005724  060105                4$:    ADD     R1,R5           ;;ADD BACK THE CONSTANT
1402  005726  005702                       TST     R2              ;;CHECK IF BCD DIGIT=0
1403  005730  001002                       BNE     5$              ;;FALL THROUGH IF 0
1404  005732  105716                       TSTB    (SP)            ;;STILL DOING LEADING 0'S?
1405  005734  100407                       BMI     7$              ;;BR IF YES
1406  005736  106316                5$:    ASLB    (SP)            ;;MSD?
1407  005740  103003                       BCC     6$              ;;BR IF NO
1408  005742  116663  000001  177777       MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
1409  005750  052702  000060         6$:   BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
1410  005754  052702  000040         7$:   BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1411  005760  110223                       MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1412  005762  005720                       TST     (R0)+           ;;JUST INCREMENTING
1413  005764  020027  000010               CMP     R0,#10          ;;CHECK THE TABLE INDEX
1414  005770  002746                       BLT     2$              ;;GO DO THE NEXT DIGIT
1415  005772  003002                       BGT     8$              ;;GO TO EXIT
1416  005774  010502                       MOV     R5,R2           ;;GET THE LSD
1417  005776  000764                       BR      6$              ;;GO CHANGE TO ASCII
1418  006000  105726                8$:    TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
1419  006002  100003                       BPL     9$              ;;BR IF NO
```

```
1420  006004  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
1421  006012  105013                  98:     CLRB    (R3)            ;;SET THE TERMINATOR
1422  006014  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
1423  006016  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
1424  006020  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
1425  006022  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
1426  006024  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
1427  006026  104401  006054                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
1428  006032  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
1429  006040  012616                          MOV     (SP)+,(SP)
1430  006042  000002                          RTI                     ;;RETURN TO USER
1431  006044  023420                  $DTBL:  10000.
1432  006046  001750                          1000.
1433  006050  000144                          100.
1434  006052  000012                          10.
1435  006054  000004                  $DBLK:  .BLKW   4
1436                                   .$DTTL  TYPE ROUTINE
1437
1438                                   ;;**********************************************************************
1439                                   ;*ROUTINE TO TYPE ASCIZ MESSAGE, MESSAGE MUST TERMINATE WITH A 0 BYTE.
1440                                   ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1441                                   ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1442                                   ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1443                                   ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1444                                   ;*
1445                                   ;*CALL:
1446                                   ;*1) USING A TRAP INSTRUCTION
1447                                   ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1448                                   ;*OR
1449                                   ;*      TYPE
1450                                   ;*      MESADR
1451                                   ;*
1452
1453  006064  105737  001157  $TYPE:   TSTB    $TPFLG          ;;IS THERE A TERMINAL?
1454  006070  100002                   BPL     1$              ;;BR IF YES
1455  006072  000000                   HALT                    ;;HALT HERE IF NO TERMINAL
1456  006074  000430                   BR      3$              ;;LEAVE
1457  006076  010046                   MOV     R0,-(SP)        ;;SAVE R0
1458  006100  017600  000002          1$:     MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
1459  006104  122737  000001  001214          CMPB    $APTENV,$ENV    ;;RUNNING IN APT MODE
1460  006112  001011                   BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
1461  006114  132737  000100  001215          BITB    $APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
1462  006122  001405                   BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
1463  006124  010037  006134                   MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
1464  006130  004737  006354                   JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
1465  006134  000000                   61$:    .WORD   0               ;;MESSAGE ADDRESS
1466  006136  132737  000040  001215  62$:    BITB    $APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
1467  006144  001003                   BNE     60$             ;;YES,SKIP TYPE OUT
1468  006146  112046                   2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1469  006150  001005                   BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1470  006152  005726                   TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1471  006154  012600                   60$:    MOV     (SP)+,R0        ;;RESTORE R0
1472  006156  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1473  006162  000002                   RTI                     ;;RETURN
```

```
1474   006164   122716   000011            48:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1475   006170   001430                      .       BEQ     8$
1476   006172   122716   000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
1477   006176   001006                            BNE     5$
1478   006200   005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
1479   006202   104401                            TYPE                    ;;TYPE A CR AND LF
1480   006204   001171                            #CRLF
1481   006206   105037   006342                   CLRB    #CHARCNT        ;;CLEAR CHARACTER COUNT
1482   006212   000755                            BR      2$              ;;GET NEXT CHARACTER
1483   006214   004737   006276            5$:     JSR     PC,#TYPEC       ;;GO TYPE THIS CHARACTER
1484   006220   123726   001156            6$:     CMPB    #FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1485   006224   001350                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
1486   006226   013746   001154                    MOV     #NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1487                                               ;;AND THE NULL CHAR.
1488   006232   105366   000001            7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1489   006236   002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1490   006240   004737   006276                    JSR     PC,#TYPEC       ;;GO TYPE A NULL
1491   006244   105337   006342                    DECB    #CHARCNT        ;;DO NOT COUNT AS A COUNT
1492   006250   000770                            BR      7$              ;;LOOP
1493
1494                                          ;HORIZONTAL TAB PROCESSOR
1495
1496   006252   112716   000040            8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1497   006256   004737   006276            9$:     JSR     PC,#TYPEC       ;;TYPE A SPACE
1498   006262   132737   000007   006342           BITB    #7,#CHARCNT     ;;BRANCH IF NOT AT
1499   006270   001372                            BNE     9$              ;;TAB STOP
1500   006272   005726                            TST     (SP)+           ;;POP SPACE OFF STACK
1501   006274   000724                            BR      2$              ;;GET NEXT CHARACTER
1502   006276   105777   172646           #TYPEC: TSTB    @#TPS           ;;WAIT UNTIL PRINTER IS READY
1503   006302   100375                            BPL     #TYPEC
1504   006304   116677   000002   172640           MOVB    2(SP),@#TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1505   006312   122766   000015   000002           CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1506   006320   001003                            BNE     1$              ;;BRANCH IF NO
1507   006322   105037   006342                   CLRB    #CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1508   006326   000406                            BR      #TYPEX          ;;EXIT
1509   006330   122766   000012   000002   1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1510   006336   001402                            BEQ     #TYPEX          ;;BRANCH IF YES
1511   006340   105227                            INCB    (PC)+           ;;COUNT THE CHARACTER
1512   006342   000000                   #CHARCNT:.WORD   0               ;;CHARACTER COUNT STORAGE
1513   006344   000207                   #TYPEX: RTS     PC
1514
1515                                          .SBTTL  APT COMMUNICATIONS ROUTINE
1516
1517                                          ;;**************************************************************************
1518   006346   112737   000001   006612   #ATY1:  MOVB    #1,#FFLG        ;;TO REPORT FATAL ERROR
1519   006354   112737   000001   006610   #ATY3:  MOVB    #1,#MFLG        ;;TO TYPE A MESSAGE
1520   006362   000403                            BR      #ATYC
1521   006364   112737   000001   006612   #ATY4:  MOVB    #1,#FFLG        ;;TO ONLY REPORT FATAL ERROR
1522   006372                            #ATYC:
1523   006372   010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1524   006374   010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1525   006376   105737   006610                   TSTB    #MFLG           ;;SHOULD TYPE A MESSAGE?
1526   006402   001450                            BEQ     5$              ;;IF NOT:  BR
1527   006404   122737   000001   001214           CMPB    #APTENV,#ENV    ;;OPERATING UNDER APT?
```

```
1528   006412  001031                          BNE     3$                      ;;IF NOT:  BR
1529   006414  132737  000100  001215          BITB    #APTSPOOL,@#ENVM        ;;SHOULD SPOOL MESSAGES?
1530   006422  001425                          BEQ     3$                      ;;IF NOT:  BR
1531   006424  017600  000004                  MOV     @4(SP),R0               ;;GET MESSAGE ADDR,
1532   006430  062766  000002  000004          ADD     #2,4(SP)                        ;;BUMP RETURN ADDR,
1533   006436  005737  001174          1$:     TST     @#MSGTYPE               ;;SEE IF DONE W/ LAST XMISSION?
1534   006442  001375                          BNE     1$                      ;;IF NOT:  WAIT
1535   006444  010037  001210                  MOV     R0,@#MSGAD              ;;PUT ADDR IN MAILBOX
1536   006450  105720                  2$:     TSTB    (R0)+                   ;;FIND END OF MESSAGE
1537   006452  001376                          BNE     2$
1538   006454  163700  001210                  SUB     @#MSGAD,R0              ;;SUB START OF MESSAGE
1539   006460  006200                          ASR     R0                      ;;GET MESSAGE LNGTH IN WORDS
1540   006462  010037  001212                  MOV     R0,@#MSGLGT             ;;PUT LENGTH IN MAILBOX
1541   006466  012737  000004  001174          MOV     #4,@#MSGTYPE            ;;TELL APT TO TAKE MSG,
1542   006474  000413                          BR      5$
1543   006476  017637  000004  006522  3$:     MOV     @4(SP),4$               ;;PUT MSG ADDR IN JSR LINKAGE
1544   006504  062766  000002  000004          ADD     #2,4(SP)                        ;;BUMP RETURN ADDRESS
1545   006512  013746  177776                  MOV     #177776,-(SP)           ;;PUSH 177776 ON STACK
1546   006516  004737  006064                  JSR     PC,@#TYPE               ;;CALL TYPE MACRO
1547   006522  000000                  4$:     .WORD   0
1548   006524                          5$:
1549   006524  105737  006612          10$:    TSTB    @#FFLG                  ;;SHOULD REPORT FATAL ERROR?
1550   006530  001416                          BEQ     12$                     ;;IF NOT:  BR
1551   006532  005737  001214                  TST     @#ENV                   ;;RUNNING UNDER APT?
1552   006536  001413                          BEQ     12$                     ;;IF NOT:  BR
1553   006540  005737  001174          11$:    TST     @#MSGTYPE               ;;FINISHED LAST MESSAGE?
1554   006544  001375                          BNE     11$                     ;;IF NOT:  WAIT
1555   006546  017637  000004  001176          MOV     @4(SP),@#FATAL          ;;GET ERROR #
1556   006554  062766  000002  000004          ADD     #2,4(SP)                        ;;BUMP RETURN ADDR,
1557   006562  005237  001174                  INC     @#MSGTYPE               ;;TELL APT TO TAKE ERROR
1558   006566  105037  006612          12$:    CLRB    @#FFLG                  ;;CLEAR FATAL FLAG
1559   006572  105037  006611                  CLRB    @#LFLG                  ;;CLEAR LOG FLAG
1560   006576  105037  006610                  CLRB    @#MFLG                  ;;CLEAR MESSAGE FLAG
1561   006602  012601                          MOV     (SP)+,R1                ;;POP STACK INTO R1
1562   006604  012600                          MOV     (SP)+,R0                ;;POP STACK INTO R0
1563   006606  000207                          RTS     PC                      ;;RETURN
1564   006610     000          @#MFLG:  .BYTE  0                       ;;MESSG, FLAG
1565   006611     000          @#LFLG:  .BYTE  0                       ;;LOG FLAG
1566   006612     000          @#FFLG:  .BYTE  0                       ;;FATAL FLAG
1567                                   .EVEN
1568           000200          APTSIZE=200
1569           000001          APTENV=001
1570           000100          APTSPOOL=100
1571           000040          APTCSUP=040
1572
1573                           ;*
1574                           ;*THIS ROUTINE WILL PROTECT THE PROGRAM
1575                           ;*FROM INTERRUPTS,
1576                           ;*
1577                           ;*THE TRAP CATCHER IS SET UP FOR
1578                           ;*      .WORD   .+2
1579                           ;*      .WORD   JSR     PC,R0
1580                           ;*
1581                           ;*ILLEGAL INTERRUPTS OR INTERRUPTS TO THE WRONG VECTOR
```

```
                                        ;*GOTO THE VECTOR AND PICK UP THE ",+2" AS AN ADDRESS
                                        ;*AND "4700" AS NEW STATUS.
                                        ;*THE ,+2 AS A PC WILL CAUSE EXECUTION OF THE "JSR PC,R0" (AN ILLEGAL INSTR),
                                        ;*AND TRAP TO LOCATION "4". IN LOCATION 10 WE HAVE A
                                        ;*POINTER HERE. IF THIS CONDITION CAUSES A TRAP TO LOC.4
                                        ;*WE WILL REPORT IT IN THE SAME MANNER THAT WE WOULD
                                        ;*REPORT ANY OTHER ERROR.
                                        ;*IF A BUSS ERROR TRAP DID OCCUR AND CAUSE A TRAP TO 4,
                                        ;*WE WILL HALT.
006614  011637  006666        IOTRD:  MOV     (6),TRTO        ;GET WHERE WE CAME TO.
006620  162737  000004  006666        SUB     #4,TRTO         ;FORM REAL ADDR.
006626  023727  006666  001000        CMP     TRTO,#1000      ;DID TRAP COME FROM LESS THAN ADDR, 1000?
006634  003402                        BLE     2$
006636  000000                1$:     HALT                    ;NO! MUST BE A BUSS ILLEGAL ADDR, TIME OUT.
                                                                ;ADDRESS CONTAINED IN TRTO.
006640  000776                        BR      1$              ;DON'T ALLOW A CONTINUE
006642                          2$:
006642  113737  001102  006672        MOVB    @TSTNM,TSTNM    ;SAVE TEST NUMBER.
006650  016637  000004  006670        MOV     4(6),TRFRO      ;GET TRAPPED FROM ADDR.
006656  062706  000004                ADD     #4,R6
006662  104006                        ERROR   6               ;ERROR! ILLEGAL INTERRUPT
                                        ;OR INTERRUPT TO WRONG
                                        ;VECTOR - IF TEST NUMBER
                                        ;IS LESS THAN 2, ITS LIKELY
                                        ;(BUT NOT EXCLUSIVELY) TO BE A
                                        ;DEVICE OTHER THAN THE DEVICE
                                        ;TO BLAME.
                                        ;IF THE INTERRUPT OCCURRED
                                        ;DURING AN INTERRUPT TEST, I'D
                                        ;SUSPECT A PROBLEM WITH THE
                                        ;DEVICE.
                                        ;IF THE ADDRESS THE INTERRUPT
                                        ;VECTOR TO IS WITHIN THE RANGE
                                        ;OF VECTORS ASSIGNED TO THE DEVICE,
                                        ;THEN I'D SUSPECT THE DEVICE
                                        ;INTERRUPTED ILLEGALLY.
                                        ;IF THE ADDRESS THE INTERRUPT
                                        ;VECTORED TO IS OUTSIDE OF THE
                                        ;RANGE ASSIGNED TO THE DEVICE,
                                        ;I'D SUSPECT THAT THE
                                        ;DEVICE PUT THE WRONG VECTOR ON
                                        ;THE BUSS DURING THE INTERRUPT
                                        ;PROCESS.
                                        ;FOR THIS ERROR - DON'T
                                        ;USE "LOOP ON ERROR" OPTION.
                                        ;ALSO EXPECT THE INTERRUPT TEST TO
                                        ;REPORT THAT THE DEVICE DIDN'T
```

```
                                                        ;INTERRUPT,
                                                        ;FOLLOW RECOMMENDED PROCEDURE
                                                        ;IN THE DOCUMENT (ON THIS DIAGNOSTIC)
                                                        ;FOR LOOPING ON ERROR

006664  000002                  RTI

006666  000000          TRTO:   .WORD   0               ;ADDR THAT WE INTERRUPTED TO
006670  000000          TRFRO:  .WORD   0               ;ADDR THAT WE INTERRUPTED FROM,
006672  000000          TSTNM:  .WORD   0               ;TEST NUMBER FROM WHICH WE CAME,

                        .SBTTL  TRAP DECODER

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

006674  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
006676  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
006702  005740                  TST     -(R0)           ;;BACKUP BY 2
006704  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
006706  006300                  ASL     R0              ;;POSITION FOR INDEXING
006710  016000  006730          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
006714  000200                  RTS     R0              ;;GO TO ROUTINE


                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

006716  011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
006720  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
006726  000002                  RTI                     ;;RESTORE THE PSW


                        .SBTTL  TRAP TABLE

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

                        ;       ROUTINE
                        ;       -------
006730  006716          $TRPAD: .WORD   $TRAP2
006732  006064                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
006734  005436                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
006736  005412                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
006740  005452                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
006742  005640                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

006744  004706                  $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

006746  004636                  $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
006750  005120                  $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
006752  005240                  $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
```

```
                                                    .SBTTL  ASCII MESSAGES
                                                    
006754  046600  046505  051117  EM1:    .ASCIZ  <200>@MEMORY READ/WRITE ERROR@
006762  020131  042522  042101
006770  053457  044522  042524
006776  042440  051122  051117
007004     000
                                                    
007005     200  042515  047515  EM2:    .ASCIZ  <200>@MEMORY DATA ERROR AFTER POWER FAIL@
007012  054522  042040  052101
007020  020101  051105  047522
007026  020122  043101  042524
007034  020122  047520  042527
007042  020122  040506  046111
007050     000
                                                    
007051     200  046103  041517  EM3:    .ASCIZ  <200>@CLOCK CSR ERROR@
007056  020113  051503  020122
007064  051105  047522  000122
                                                    
007072  041600  047514  045503  EM4:    .ASCIZ  <200>@CLOCK ADDR, ERROR@
007100  040440  042104  027122
007106  042440  051122  051117
007114     000
                                                    
007115     200  046103  041517  EM5:    .ASCIZ  <200>@CLOCK INTERRUPT ERROR@
007122  020113  047111  042524
007130  051122  050125  020124
007136  051105  047522  000122
                                                    
007144  044600  052116  051105  EM6:    .ASCIZ  <200>@INTERRUPT ERROR@
007152  052522  052120  042440
007160  051122  051117     000
                                                    
007165     200  051105  050122  DH1:    .ASCIZ  <200>@ERRPC    ADDR    WAS     S/B@
007172  020103  020040  042101
007200  051104  020040  020040
007206  040527  020123  020040
007214  020040  027523  000102
                                                    
007222  042600  051122  041520  DH3:    .ASCIZ  <200>@ERRPC    WAS     S/B@
007230  020040  053440  051501
007236  020040  020040  051440
007244  041057     000
                                                    
007247     200  051105  050122  DH4:    .ASCIZ  <200>@ERRPC    ADDR@
007254  020103  020040  042101
007262  051104     000
                                                    
007265     200  042524  052123  DH6:    .ASCIZ  <200>@TEST     ERRPC   TO      FROM@
007272  020040  020040  051105
007300  050122  020103  020040
007306  047524  020040  020040
007314  020040  051106  046517
```

```
1744   007322   000
1745
1746            007324                          .EVEN
1747   007324   001116  001362  001126  DT1:    .WORD   $ERRPC,RADDR,$BDDAT,$GDDAT,0
1748   007332   001124  000000
1749
1750   007336   001116  001126  001124  DT3:    .WORD   $ERRPC,$BDDAT,$GDDAT,0
1751   007344   000000
1752
1753   007346   001116  001336  000000  DT4:    .WORD   $ERRPC,LKS,0
1754
1755   007354   006672  001116  006666  DT6:    .WORD   TSTNM,$ERRPC,TRTO,TRFRO
1756   007362   006670
1757
1758   007364   000000  000000          DF0:    .WORD   0,0
1759
1760   007370   000000                  MEMST:  .WORD   0               ;MEMORY WILL BE WRITTEN
1761                                                                   ;FROM HERE UP
1762
1763            000001                          .END
```

```
ABASE = 177546          1558    240     281     347
ACDW1 = 000000          240     283
ACDW2 = 000000          240
ACPUOP= 000000          240     255
ADCK    001356          358#    417#    536#    557     953
ADDW0 = 000000          240
ADDW1 = 000000          240
ADDW10= 000000          240
ADDW11= 000000          240
ADDW12= 000000          240
ADDW13= 000000          240
ADDW14= 000000          240
ADDW15= 000000          240
ADDW2 = 000000          240
ADDW3 = 000000          240
ADDW4 = 000000          240
ADDW5 = 000000          240
ADDW6 = 000000          240
ADDW7 = 000000          240
ADDW8 = 000000          240
ADDW9 = 000000          240
ADEVCT= 000000          240     246
ADEVM = 000000          240     282
AENV  = 000000          240     251
AENVM = 000000          240     252
AFATAL= 000000          240     243
AMADR1= 000000          240     268
AMADR2= 000000          240     272
AMADR3= 000000          240     275
AMADR4= 000000          240     278
AMAMS1= 000000          240     262
AMAMS2= 000000          240     270
AMAMS3= 000000          240     273
AMAMS4= 000000          240     276
AMSGAD= 000000          240     248
AMSGLG= 000000          240     249
AMSGTY= 000000          240     242
AMTYP1= 000000          240     263
AMTYP2= 000000          240     271
AMTYP3= 000000          240     274
AMTYP4= 000000          240     277
APASS = 000000          240     245
APRIOR= 000000          240
APTCSU= 000040          1466    15710
APTENV= 000001          1022    1459    1527    15690
APTSIZ= 000200          408     1568#
APTSPO= 000100          1461    1529    15700
ASWREG= 000000          240     253
ATESTN= 000000          240     244
AUNIT = 000000          240     247
AUSWR = 000000          240     254
AVECT1= 000100          1560    240     279     349     350
AVECT2= 000000          240     280
BAD     001364          361#    418#    851     857#    888#    903
```

```
BIT0  = 000001          1398    770
BIT00 = 000001          1298    139
BIT01 = 000002          1288    138
BIT02 = 000004          1278    137
BIT03 = 000010          1268    136
BIT04 = 000020          1258    135
BIT05 = 000040          1248    134
BIT06 = 000100          1238    133
BIT07 = 000200          1228    132
BIT08 = 000400          1218    131    1119
BIT09 = 001000          1208    130    1033    1127
BIT1  = 000002          1388
BIT10 = 002000          1198   1010
BIT11 = 004000          1188   1134
BIT12 = 010000          1178
BIT13 = 020000          1168   1017
BIT14 = 040000          1158   1105
BIT15 = 100000          1148
BIT2  = 000004          1378
BIT3  = 000010          1368
BIT4  = 000020          1358
BIT5  = 000040          1348
BIT6  = 000100          1338    574     576     595     626     960
BIT7  = 000200          1328    694
BIT8  = 000400          1318
BIT9  = 001000          1308
BLKCK   001366          3628    479*    802*    859     864*    925
BPTVEC* 000014          1468
CKEND   003710           937    9498
CKSWR = 104407          1006    1032    1103    1104   16868
CK1     003652           930     934    9368     940     947
CR    = 000015           548    1505    1515
CRLF  = 000200           558     447    1476    1515
DDISP = 177570           618     221     396
DFC     007364           308     315     322     329     336     343   17508
DH1     007165           306     313   17248
DH3     007222           320   17308
DH4     007247           327     334   17358
DH6     007265           341   17398
DISPLA  001142          2218    396*    404*   1009*   1149*
DISPRE  000174            36#    404
DSWR  = 177570            60#    220     395
DT1     007324           307     314   17478
DT3     007336           321   17508
DT4     007346           328     335   17538
DT6     007354           342   17558
EMTVEC* 000030          1498    382*    383*
EM1     006754           305   16928
EM2     007005           312   16988
EM3     007051           319   17068
EM4     007072           326   17108
EM5     007115           333   17158
EM6     007144           340   17208
ERRVEC* 000004          1428    393     394*    405*    526     527*    533*    550*   1110    1111*   1113*   1116*
```

```
GNS     = ******  U      446     816     822     831     983    1678    1679    1680    1681    1682    1684    1686    1687
                      1688
GTSWR   = 104406         441    1684#
HILIM     001344         352#    484     502     775     928     932
HT      = 000011          52#   1474    1515
IOTRD     006614          34    1593#
IOTVEC= 000020         1478#    380#    381#
KVECT     001340         349#    643#    666#    667#    726#    749#    750#
KVECTP    001342         350#
LF      = 000012          53#   1509    1515
LKS       001336         347#    529     534#    574#    575     576     593#    594     626#    629     636     668#    683#
                         686     693     700#    702     867#    956    1753
LNWR      001370         363#    370#    472#    478#    484     491#    492#    935
LOLIM     001346         353#    360     472     497     773     862     929     931
LOOP      002002         450#    843
LOW2      001350         354#    862
MCLK      003020         523    1752#
MEMST     007370         353     354    1760#
MEMWR     001360         359#    471#    495#    865#    933
NOCLK     001372         364#    521     826
PC      =%000007         73#    807#    810#    837#    842    1019#   1025#   1078#   1207#   1464#   1483#   1490#   1497#
                      1511#   1513#   1546#   1563#
PCOUNT    001352         356#    416#    824     949#
PDBAD     003352         853#    904
PDOWN     003344         413     851#    854     917
PD2       003356         852     857#    912
PD3       003554         911     917#
PFLAG     001354         357#    563#    578     597     615#    633     655     681#    690     704     711     724#    741
                         768#    907#    910
PIRQ    = 177772          59#
PIRQVE= 000240         1539#
PR0     = 000000          76#
PR1     = 000040          77#
PR2     = 000100          78#
PR3     = 000140          79#
PR4     = 000200          80#
PR5     = 000240          81#
PR6     = 000300          82#
PR7     = 000340          83#
PS      = 177776          56#     57
PSW     = 177776          57#
PUEND     003746         954     961    9658
PUP       003506         807    9028
PUPBD     004026         905    9078    908
PU2       003532        9108    915
PWRVEC= 000024         148#    413#    414#    887#    905#    912#    917#
RADDR     001362         360#    508#    509#    510     780#    781#    782     876     942#    943#    966#   1747
RDCHR   = 104410      1266    1687#
RDLIN   = 104411      1688#
RESVEC= 000010         143#
R0      =%000000         64#    371#    836#    837     868     878#    880#    974#    976#    978    1050    1051#   1052#
                      1059#   1060#   1061#   1062#   1063#   1064    1069    1074#   1076#   1080    1082    1382    1392#
                      1396    1412    1413    1426#   1457    1458#   1463    1468    1471#   1523    1531#   1535    1536
                      1538#   1539#   1540    1562#   1655    1656#   1657    1658#   1659#   1660#   1661#
```

```
R1      =%000001        680    473*   476*   478    491    505    511    769*   772*   778    783    869    973*
                       1383   1396*  1397   1401   1425*  1524   1561*
R2      =%000002        660    498*   502    508    627*   631*   684*   688*   738*   739*   773*   775    780
                        870    972*  1384   1395*  1399*  1402   1409*  1410*  1411   1416*  1424*
R3      =%000003        670    368*   370    497*   498    871    971*  1262   1263*  1264   1267*  1268   1272
                       1274   1276*  1278*  1325   1334*  1340*  1341*  1344*  1349*  1350*  1351   1360*  1385
                       1393*  1394*  1408*  1411*  1420*  1421*  1423*
R4      =%000004        680    872    929*   931*   936    942    970*  1326   1328*  1329*  1330*  1331   1332*
                       1346   1348*  1356*  1389*
R5      =%000005        690    873    928*   932*   935*   936    969*  1327   1333*  1335*  1337*  1338*  1339*
                       1340   1350*  1386   1388*  1390*  1397*  1401*  1416   1422*
R6      =%000006        700    374*   375*   376    664*   745*   877    902*  1607*
R7      =%000007        710
SP      =%000006        720    378*   393*   401*   405    818*   824*  1014   1035*  1038*  1050*  1055*  1076
                       1080*  1110*  1113   1115   1116   1145   1146   1150*  1167*  1168*  1169   1176*  1179*
                       1180*  1184*  1185*  1189   1192*  1196   1198   1200   1201*  1208   1210   1212*  1213
                       1215*  1216*  1217*  1218*  1219*  1234*  1235*  1238*  1239*  1240   1244*  1245*  1246
                       1249   1251   1253*  1262*  1267   1278   1279*  1280*  1281*  1317*  1318   1319   1320*
                       1325*  1326*  1327*  1333   1358   1359   1360   1361*  1362*  1382*  1383*  1384*  1385*
                       1386*  1387*  1388   1391*  1404   1406*  1408   1418   1420   1422   1423   1424   1425
                       1426   1428*  1429*  1457*  1458   1460*  1470   1471   1472*  1474   1476   1478   1484
                       1486*  1488*  1496*  1500   1504   1505   1509   1523*  1524*  1531   1532*  1543   1544*
                       1548*  1555   1556*  1561   1562   1655*  1656   1666*  1667*
SR6     003504          777*   0978   902
STACK = 001100          478    378
START   001374           42   3688
STKLMT= 177774          588
SWR     001140         2208    376    395*   397    403*   410*   439   1010   1017   1029   1033   1105   1119
                       1121   1127   1134   1163   1200*
SWREG   000176          378    403    439   1163   1176
SW0   = 000001         1118
SW00  = 000001         1018    111
SW01  = 000002         1008    110
SW02  = 000004          998    109
SW03  = 000010          988    108
SW04  = 000020          978    107
SW05  = 000040          968    106
SW06  = 000100          958    105
SW07  = 000200          948    104
SW08  = 000400          938    103
SW09  = 001000          928    102
SW1   = 000002         1108
SW10  = 002000          918
SW11  = 004000          908
SW12  = 010000          898
SW13  = 020000          888
SW14  = 040000          878
SW15  = 100000          868
SW2   = 000004         1098
SW3   = 000010         1088
SW4   = 000020         1078
SW5   = 000040         1068
SW6   = 000100         1058
SW7   = 000200         1048
```

```
SW8    = 000400          1038
SW9    = 001000          1028
TBITVE= 000014          1448
TKVEC  = 000060          1518
TPVEC  = 000064          1528
TRAPVE= 000034          1508    384*    385*
TRFRO   006670         1606*   1644#   1755
TRTO    006666         1593*   1594*   1596    1643#   1755
TRTVEC= 000014          1458
TSTNM   006672         1604*   1645#   1755
TST1    002002          4648
TST10   003124           776     7918
TST2    002210           503     5198
TST3    002302          5558
TST4    002440           590     599     6138
TST5    002610           641     6798
TST6    002724           698     709     7218
TST7    003026          7648
TYPDS  = 104405         1682#
TYPE   = 104401           433     814     820     829     981    1012    1020    1049    1066    1068    1071    1073    1077
                         1084    1174    1175    1178    1191    1202    1221    1270    1273    1277    1352    1427    1479
                        1678#
TYPOC  = 104402           819     825    1057    1081    1177    1679#
TYPON  = 104404         1681#
TYPOS  = 104403         1680#
$APTHD  001000           180    1868
$ASTAT= ****** U        1549    1564
$ATYC   006372          1520   1522#
$ATY1   006346          1518#
$ATY3   006354          1464   1519#
$ATY4   006364          1028   1521#
$AUTOB  001134          2178    443*   1171    1292
$BASE   001250          2818
$BDADR  001122          2128
$BDDAT  001126          2148    510*    529*    575*    594*    595     636*    693*    702*    782*    875     941*    956*
                         960    967*   1747    1750
$BELL   001164          2328   1012    1041
$CDW1   001254          2838
$CHARC  006342         1481*   1491*   1498   1507*   1512#
$CKSWR  004636         1163#   1686
$CMTAG  001100          2008    373     374     382     386     387
$CM3   = 000000          2308
$CNTLG  005363         1174    1287#
$CNTLU  005356         1191    1286#
$CPUDP  001222          2558
$CRLF   001171          2348   1020    1041    1049    1068    1073    1077    1202    1286    1480    1515
$DBLK   006054         1393    1427   1435#
$DEVCT  001204          2468
$DEVM   001252          2828
$DOAGN  003334           809     835     8418
$DTBL   006044         1396   1431#
$ENDAD  003324           166     837#
$ENDCT  003164          8118
$ENULL  003340          8448
```

```
$ENV    001214          2518#   437    1022    1459    1527    1551
$ENVM   001215          2528#   408     1461    1466    1529
$EOP    003126          3018
$EOPCT  003156          3088    812
$ERFLG  001103          2038    1007*   1041    1093    1123    1125    1131*   1153
$ERMAX  001115          2098    388*    1125    1148*   1153
$ERROR  004032          382     1005#
$ERRPC  001116          2108    1014*   1015*   1016    1041    1055    1747    1750    1753    1755
$ERRTB  001256          3008    1063
$ERRTY  004220          1019    1040#
$ERTTL  001112          2078    1013*   1041
$ESCAP  001162          2318    387*    1036    1038    1041    1147*
$ETABL  001214          2508
$ETEND  001256          192     2848
$FATAL  001176          2438    1555*
$FFLG   006612          1518*   1521*   1549    1558*   1566#
$FILLC  001156          2288    1484    1515
$FILLS  001155          2278    1515
$GDADR  001120          2118
$GDDAT  001124          2138    511*    581*    592*    637*    694*    701*    783*    874     879*    921*    924*    927*
                        939     957*    968*    1747    1750
$GET42  003314          8348
$GTSWR  004706          11758   1684
$HD   = 000001          11      12
$HIBTS  001000          1878
$ICNT   001104          2048    1138*   1139    1141*   1152
$INTAG  001135          2188    1203    1292
$ITEMB  001114          2088    1016*   1024    1041    1052
$LF     001172          2358    1041    1277    1286    1515
$LFLG   006611          1559*   1565#
$LPADR  001106          2058    389*    469*    1129*   1145*   1150    1152
$LPERR  001110          2068    390*    470*    1035    1129    1146*   1152
$MADR1  001226          2688
$MADR2  001232          2728
$MADR3  001236          2758
$MADR4  001242          2788
$MAIL   001174          188     192     2418    407     437     1022    1144    1459
$MAMS1  001224          2628
$MAMS2  001230          2708
$MAMS3  001234          2738
$MAMS4  001240          2768
$MBADR  001002          1888
$MFLG   006610          1519*   1525    1560*   1564#
$MNEW   005401          1178    1290#
$MSGAD  001210          2488    1535*   1538
$MSGLG  001212          2498    1540*
$MSGTY  001174          2428    1533    1541*   1553    1557*
$MSWR   005370          1175    1288#
$MTYP1  001225          2638
$MTYP2  001231          2718
$MTYP3  001235          2748
$MTYP4  001241          2778
$MXCNT  004634          1142    1152#
$NULL   001154          2268    1486    1515
```

```
$NWTST= 000001          453#    455     516#    552#    610#    670#    672     718#    755#    757     788#
$OCNT   005634         1324*   1353*   1366#
$OMODE  005636         1319*   1323*   1328    1331*   1342*   1368#
$OVER   004620         1106    1122    1130    1140    1149#
$PASS   001202          245#    407*    415*    474     770     805*    806*    818     844     922     1136    1153
$PASTM  001006          190#
$QUES   001170          233#    1041    1221    1270    1286    1515
$RDCHR  005120         1234#    1687
$RDDEC= ****** U       1689
$RDLIN  005240         1262#    1688
$RDOCT= ****** U       1689
$RDSZ = 000010         1255#
$RTNAD  003336          843#
$R2A  = ****** U       1689
$SAVRE= ****** U       1689
$SCOPE  004354          380    1102#
$SETUP= 000107          367#     379     380     382     384     386     387     389     433     434     803     1006    1032
                       1040    1103    1158    1292
$STUP = 177777          367#
$SVLAD  004564         1114    1143#
$SVPC = 000204         1643     169
$SWR  = 167400           1#      11      17      18      19      20      21      22      23     230     231     232     386
                        387     389     390     465     520     556     614     680     722     765     792     798     804
                        836     842     844     997     998     999    1000    1001    1010    1017    1029    1033    1041
                       1094    1095    1096    1097    1098    1105    1117    1119    1120    1123    1124    1125    1132
                       1133    1134    1146    1149    1152
$SWREG  001216          253#     410
$SWRMK= 000000           23      24    1098    1099    1121
$TESTN  001200          244#     468*   1144*
$TIMES  001160          230#     386*    465*    722*    765*    804*   1132*   1139    1142*   1152
$TKB    001146          223#    1156    1167    1184    1238    1244
$TKS    001144          222#    1156    1165    1181    1205*   1236    1242
$TN   = 000011           11#     157#     453     465#     503     516     520#     552     556#     590     599     610     614#
                        641     670     680#     698     709     718     722#     755     765#     776     788     792#
$TPB    001152          225#    1504*   1515
$TPFLG  001157          229#    1453    1515
$TPS    001150          224#    1502    1515
$TRAP   006674           384    1658#
$TRAP2  006716         1666#    1677
$TRP  = 000012         1670#    1679#    1680#    1681#    1682#    1683#    1684     1685#    1686     1687#    1688#    1689#
$TRPAD  006730         1660     1677#
$TSTN   001004          189#
$TSTNM  001102          202#     467*    752*    803*    1009    1041    1093    1121    1143*   1144    1149    1153    1604
$TTYIN  005346         1263     1264    1281    1285#
$TYPBN= ****** U       1683
$TYPDS  005640         1381#    1682
$TYPE   006064         1453#    1546    1670    1678
$TYPEC  006276         1207     1483    1490    1497    1502#    1503
$TYPEX  006344         1508     1510    1513#
$TYPOC  005436         1322#    1679
$TYPON  005452         1321     1324#    1681
$TYPOS  005412         1317#    1680
$UNIT   001206          247#
$UNITM  001010          191#
```

```
$USWR   001220            2540
$VECT1  001244            2790
$VECT2  001246            2800
$XTSTR  004370           11080
$$GET4= 000000            8360
$OFILL  005635           1318*   1322*   1332    13670
$40CAT= ****** U          1019    1105
.     = 007372             270     330     350     380     410     164     1650    1670    1690    1700    176     1770    1790
                          1810    1990    236     377     389     390     4470    8320    844     8450    9840    1041    10870
                          1152    1153    1156    12850   1286    1292    14350   1515    15670   17460
,$ASTA= ****** U          1519    1522
,$X   = 001000            1760    181
```

```
COMMEN   1540
DFC      3020   308    315    322    329    336    343
ENDCOM   1540
ENDPAS   7930   813
ERROR     480   513    539    545    583    601    639    659    696    715    747    785    945    962   1609
ESCAPE   1540
GETPRI   1540
GETSWR   1540  4340
MULT     1540
NEWTST   1540   453    516    552    610    670    718    755    788
POP      1540  1422   1561   1562
PR       1590   419    564    616    644    728
PUSH     1540  1381   1522   1524   1545
REPORT   1540
SCOPE     490   519    555    613    679    721    764    791
SETPRI   1540
SETTRA  16700  1679   1680   1681   1682   1684   1686   1687   1688
SETUP    1540   372
SKIP     1540   503    590    599    641    698    709    776
SLASH    1540
SPACE    1540
STARS    1540   162    173    175    182    195    236    239    453    463    516    518    552    554    610
          612   670    678    710    720    755    763    788    790    796    993   1043   1090   1155   1158
         1226  1255   1294   1371   1438   1517   1649
SWRSU    1540  3910
TRMTRP  16700
TYPBIN   1540
TYPDEC   1540
TYPNAM   1540   429
TYPNUM   1540
TYPOCS     10  1540
TYPOCT   1540   810    824   1055   1079   1176
TYPTXT   1540   814    820    829    981
ZZ1      4520   455   6690    672   7540    787
$$CMRE   1930
$$CMTM   1930
$$ESCA   1540
$$NEWT   1540   453    516    552    610    670    718    755    788
$$SET   16700  1679   1680   1681   1682   1684   1686   1687   1688
$$SETM   4070
$$SKIP   1540   503    590    599    641    698    709    776
.EQUAT     10    44
.HEADE     10
.SETTR     10
.SETUP     10   367
.SWRHI     10    13
.SWRLO    240
.TRMTR     10
.$ACT1     10   160
.$APTB   2370
.$APTH     10   171
.$APTY     10  1515
.$CMTA     10   193
.$EOP      10   794
```

```
.$ERRO      10    991
.$ERRT      10   1041
.$READ      10   1153
.$SCOP      10   1080
.$TRAP      10   1647
.$TYPD      10   1369
.$TYPE      10   1436
.$TYPO      10   1292
```

```
ADD      492   664   667   745   750  1063  1192  1201  1320  1330  1401  1472  1532  1544  1556
        1607
ASL     1060  1061  1062  1215  1216  1217  1659
ASLB    1406
ASR     1539
BCC     1407
BEQ      409   438   475   485   503   506   522   771   776   779   827   835   852   860   863
         923   926   934   937   940   954   961   975  1008  1011  1034  1037  1065  1070  1083
        1120  1122  1124  1128  1137  1172  1199  1214  1347  1462  1475  1510  1526  1530  1550
        1552
BGE     1140
BGT      809  1211  1252  1354  1415
BHI     1126
BIC      806  1168  1185  1212  1239  1245  1253  1344
BIS      574  1219  1349  1350  1409  1410
BISB    1052
BIT      576   595   960  1010  1017  1033  1105  1119  1127  1134
BITB     408   474   770   922  1461  1466  1498  1529
BLE     1597
BLO      486
BLOS    1265
BLT     1209  1250  1355  1398  1414  1489
BMI      630   687  1405
BNE      377   398   432   436   440   558   577   579   596   598   632   634   656   689   691
         703   708   712   740   742   881   904   911   980  1018  1023  1053  1075  1106  1135
        1164  1170  1190  1197  1204  1241  1247  1269  1275  1345  1403  1460  1467  1469  1477
        1485  1499  1506  1528  1534  1537  1554
BPL     1030  1166  1182  1237  1243  1343  1389  1419  1484  1503
BR       400   442   445   490   493   537   543   560   590   599   641   662   698   709   743
         815   821   830   854   891   915   930   947   982   988  1020  1058  1085  1108  1114
        1117  1130  1133  1193  1220  1222  1248  1271  1321  1336  1357  1400  1417  1486  1492
        1492  1501  1508  1520  1542  1602
CLR      371   375   386   387   407   415   416   417   418   473   479   495   563   592   593
         615   627   660   681   683   684   700   701   724   730   768   769   803   804   864
         867   888   921   957  1051  1132  1147  1179  1180  1334  1392  1395
CLRB    1131  1276  1421  1481  1507  1558  1559  1560
CMP      376   397   439   484   502   505   775   778   862   910   936   939  1115  1139  1163
        1169  1189  1194  1208  1210  1240  1246  1249  1251  1264  1413  1596
CMPB     437  1022  1121  1125  1171  1203  1268  1274  1459  1474  1476  1484  1505  1509  1527
COM      927
DEC      688   807   880  1059
DECB     976  1342  1353  1488  1491
EMT       40
HALT     487   559   853   890   914   987  1031  1485  1599
INC      431   631   802   805   857   868   907   949  1013  1138  1218  1348  1356  1399  1587
INCB     739  1007  1143  1511
IOT       49
JMP       42   523   842
JSR      837  1019  1025  1207  1464  1483  1490  1497  1546
MOV      368   369   370   374   378   380   381   382   383   384   385   389   390   393   394
         395   396   401   403   404   405   410   413   414   421   422   465   468   469   470
         471   472   476   478   491   497   498   508   510   511   526   527   529   533   534
         536   550   566   567   575   581   594   619   620   626   636   637   643   646   647
         666   693   694   702   722   726   730   731   749   752   768   772   773   780   782
```

```
            703   810   818   824   834   868   869   870   871   872   873   874   875   876   877
            878   879   887   902   905   912   917   924   928   929   931   932   935   941   942
            956   966   967   968   969   970   971   972   973   974  1009  1014  1035  1038  1050
           1055  1064  1069  1074  1076  1080  1110  1111  1113  1116  1129  1141  1142  1145  1146
           1149  1150  1176  1200  1205  1234  1235  1262  1263  1278  1279  1280  1281  1317  1325
           1326  1327  1333  1340  1358  1359  1360  1361  1362  1382  1383  1384  1385  1386  1387
           1388  1393  1396  1416  1422  1423  1424  1425  1426  1428  1429  1457  1458  1463  1471
           1486  1523  1524  1531  1535  1540  1541  1543  1545  1555  1561  1562  1593  1606  1655
           1656  1660  1666  1667
MOVB        388   443   467  1016  1024  1144  1148  1167  1184  1238  1244  1267  1272  1318  1319
           1322  1323  1324  1328  1331  1332  1351  1391  1394  1408  1411  1420  1468  1496  1504
           1518  1519  1521  1604  1658
NEG        1329  1390
NOP         464   654   838   839   840
RESET       836
ROL        1335  1337  1338  1339  1341
RTI         402   423   860   621   640   732   905  1040  1151  1206  1254  1282  1363  1430  1473
           1641  1660
RTS        1078  1513  1563  1661
SUB         509   701   943  1015  1397  1538  1594
TRAP       1670  1679  1680  1681  1682  1684  1686  1687  1688
TST         435   521   557   570   597   633   655   690   704   711   741   826   851   859   903
            925   933   953   978  1029  1036  1082  1112  1136  1198  1213  1346  1402  1412  1470
           1478  1500  1533  1551  1553  1667
TSTB        629   686  1123  1165  1181  1236  1242  1404  1418  1483  1502  1525  1536  1549
.ASCII      233   234
.ASCIZ      232   238   447   817   823   832   904  1006  1206  1287  1288  1290  1692  1698  1706
           1710  1718  1720  1724  1730  1735  1749
.BLKB      1285
.BLKW      1435
.BYTE       202   203   208   209   217   218   226   227   228   229   251   252   262   263   270
            271   273   274   276   277   844  1026  1027  1203  1204  1364  1365  1366  1367  1564
           1565  1566
.DSABL     1223
.ENABL        1  1156
.END       1763
.ENDC         6    20    22    23    24    48   140   154   163   167   169   174   176   183   196
            200   202   230   231   232   233   237   240   262   270   273   276   279   280   281
            282   283   286   367   378   379   382   384   386   387   389   391   412   433   439
            445   447   454   455   463   464   465   466   504   517   518   519   520   553   554
            555   556   591   600   611   612   613   614   642   671   672   678   679   680   699
            710   719   720   721   722   723   756   757   763   764   765   766   777   789   790
            791   792   797   798   800   803   809   812   813   817   823   832   834   836   842
            844   845   984   994   997  1007  1014  1019  1020  1021  1029  1040  1041  1044  1069
           1088  1091  1094  1099  1105  1107  1110  1121  1122  1123  1125  1127  1134  1138  1143
           1145  1149  1152  1153  1156  1157  1159  1187  1223  1227  1255  1256  1263  1265  1268
           1270  1286  1292  1295  1372  1439  1468  1518  1519  1522  1549  1564  1680  1686  1689
           1678  1679  1680  1681  1682  1683  1684  1685  1686  1687  1688  1689
.EQUIV       48    49    57   102   103   104   105   106   107   108   109   110   111   130   131
            132   133   134   135   136   137   138   139
.EVEN       240   447   817   823   832   848   904  1007  1567  1746
.IF           2    20    21    22    23    24    46   112   140   162   165   167   173   175   182
            195   199   201   230   231   232   236   237   239   262   270   273   276   279   286
            281   282   283   284   286   367   373   378   380   382   384   386   387   389   407
```

```
                 432     433     434     437     446     453     455     463     465     466     503     516     518     520     552
                 554     556     590     599     610     612     614     641     670     672     678     680     698     709     710
                 720     722     723     755     757     763     765     766     776     788     790     792     796     797     798
                 799     800     802     808     811     813     816     822     831     834     836     842     844     845     903
                 993     996    1007    1010    1017    1019    1020    1022    1029    1033    1040    1041    1043    1058    1074
                1090    1093    1098    1104    1105    1117    1119    1120    1121    1123    1124    1125    1134    1136    1144
                1146    1151    1152    1153    1155    1157    1158    1159    1187    1226    1227    1255    1263    1264    1268
                1269    1285    1286    1292    1294    1371    1438    1459    1517    1519    1522    1549    1564    1649    1655
                1659    1670    1679    1680    1681    1682    1683    1684    1686    1687    1688    1689
.IFF              20      22      23      24      46     163     167     169     174     176     183     196     199     202     230
                 237     240     378     432     433     453     454     455     464     466     504     517     518     519     520
                 553     554     555     556     591     600     611     612     613     614     642     671     672     679     680
                 699     710     719     720     721     722     756     757     764     765     777     789     790     791     792
                 797     799     802     809     812     844     994     996    1010    1040    1041    1044    1059    1088    1091
                1118    1121    1122    1125    1152    1156    1159    1227    1229    1234    1255    1256    1268    1269    1286
                1295    1372    1439    1518    1650    1656
.IFT              447     817     823     832     984    1020    1133    1229    1234
.IFTF             447     817     823     832     984    1019    1131    1174    1227    1230
.IIF                1       6      11      12      17      18      19      20      23      24     236     240     379     382     386
                 387     389     390     433     798     803     804     819     825     844     845     997     998     999    1000
                1001    1006    1032    1040    1041    1056    1081    1094    1095    1096    1097    1098    1099    1103    1132
                1133    1149    1152    1153    1156    1177    1278    1286    1292    1515    1670    1679    1680    1681    1682
                1684    1686    1687    1688
.IRP              367     483     516     552     610     670     718     755     788     802    1104    1302    1422    1523    1524
                1545    1561    1562
.LIST               1      23      33     154     230     237     240     367     391     433     434     447     453     465     516
                 520     552     556     610     614     670     680     718     722     755     765     788     792     803     817
                 823     832     836     984    1040    1098    1255    1670    1678    1679    1680    1681    1682    1683    1684
                1685    1686    1687    1688    1689
.MACRO            24     159     193     302     407     482     669     754     793    1670
.MCALL             1     154     237     391     434
.MEXIT           285
.NLIST             1      23      33     154     230     237     240     367     391     433     434     447     453     465     516
                 520     552     556     610     614     670     680     718     722     755     765     788     792     803     817
                 823     832     836     984    1040    1098    1255    1670    1678    1679    1680    1681    1682    1683    1684
                1685    1686    1687    1688    1689
.PAGE            193     286
.REPT             33
.SBTTL            13      25      44     160     171     193     237     286     372     429     434     453     516     552     610
                 670     718     755     788     794     919     951     991    1041    1088    1153    1293    1369    1436    1515
                1647    1670    1690
.TITLE             1
.WORD             33      34      36      37      39     168     187     188     189     190     191     192     201     204     205
                 206     207     210     211     212     213     214     215     216     219     220     221     242     243     244
                 245     246     247     248     249     253     254     255     268     272     275     276     279     280     281
                 282     283     347     349     350     352     353     354     356     357     358     359     360     361     362
                 363     364     808     811     843    1067    1072    1368    1468    1812    1547    1643    1644    1648    1677
                1747    1750    1753    1755    1758    1760

ERRORS DETECTED:  0
```

```
*DVKPA,DVKPA/SOL/CRF=DVKPA
RUN-TIME:  73 22  4 SECONDS
CORE USED:  25K
```