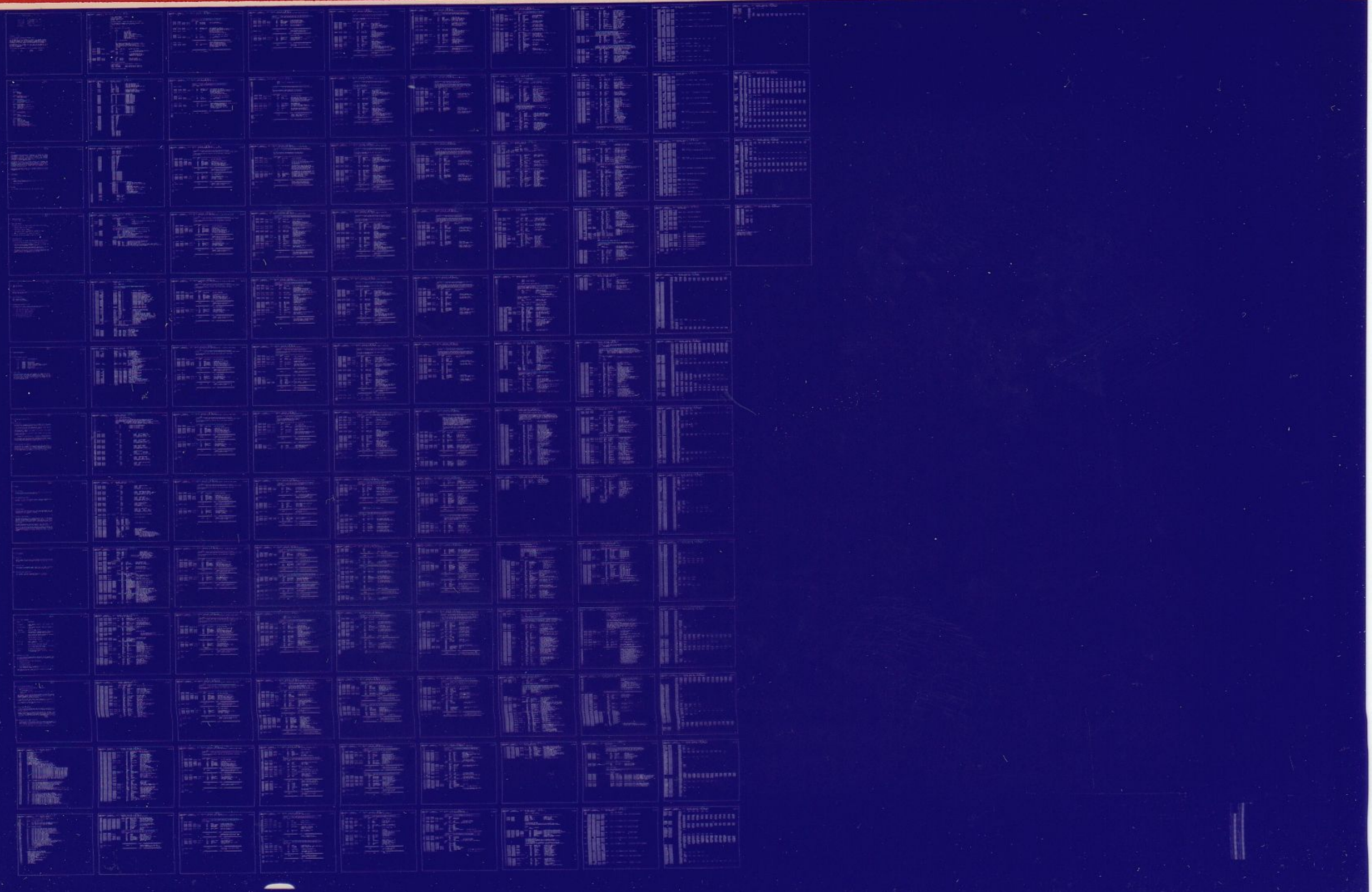
MINC-11

MNCKW DIAGNOSTIC
CVMNCB0

AH-B092B-MC

FICHE 1 OF 1

DEC 1978
digital
MADE IN USA



IDENTIFICATION

PRODUCT CODE: AC-B091B-MC

PRODUCT NAME: CVMNCB0 MNCKW DIAG (CLOCK)

DATE CREATED: AUGUST 1978

MAINTAINER: DIAGNOSTIC ENGINEERING

table of contents

1.0 abstract

this program allows the user to check-out or debug the mnckw programmable real-time clock. the logic test is self contained and needs no external maintenance hardware or operator intervention with only one exception: if the customer hardware connected to the mnckw could inject signals on st2, st1, or slave in inputs, it must be disconnected.

even though the mnckw is a q bus 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. it can be used on an lsi-11 or by cpu's that have hardware switch registers, see section 8.6.

every effort was made to make this program conform to lsi-11 programming restrictions, however; the user should read sections 7.2 and 7.3.

2.0 requirements

2.1 equipment

1. pdp-11 family computer with 8k of memory (or more) an i/o terminal (la36, vt100, etc.)
2. mnckw under test.

2.2 storage

this program occupies and uses only the lower 8k of memory.

SEQ 0004

3.0 loading procedure

3.1 paper tape 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 rydp based method

standard procedure for normal xxdp operation should be followed.

1. ensure that the diagnostic disk is installed in drive 0.
2. boot the disk by typing ''173000g'' if in the micro-code odt state or cycling the power ''on-off'' switch.
3. upon successful booting, the diagnostic monitor will identify itself and inform the operator of selectable options.
4. the operator should type ''r mncb'' followed by depressing the ''return'' key. this operation will load the diagnostic into memory and start the program at location 200.

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 program
204 restart of program
210 tester starting address

4.3 program and/or operator action

the operator must type a single test indicator character followed by a "return". the following characters are used:

l - logic test with no dwarf connected.

d logic test with dwarf connected.

b - base or vector address changes.

g get new switch register value.

h - help operator and retype this list.

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
12	010000	inhibit sizing the number of mnckw's
11	004000	inhibit iterations (short pass)
10	002000	bell on error
09	001000	loop on error
08	000400	loop on test in swr <7:0>

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

5.3.1 logic test

the first pass through the program will be made with iterations inhibited. successive passes will enable iterations if swr11=0.

if not inhibited by apt, the program will look for more mnckw's to exercise, one pass will exercise all mnckw's.

the program will report the number of mnckw's found before starting the logic test.

at end of pass when all units have been tested, the following typeout will occur:

```
''endpass 12 - total errors 4    ;bad units 0000000000000100
```

this indicates that the program has completed 10 decimal passes. during that time 4 decimal errors were detected. also we tested 4 units and the third unit was the only unit to fail.

5.4 inhibiting auto-size feature

this program will automatically auto-size and test each mnckw it detects on the system. to inhibit this feature, set switch register bit 12 or set bit 15 of location '$envm''. also, to test an individual mnckw in a group, set this bit and refer to section 3.2 for changing the base address of the mnckw under test.

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 halts

any halt in the trap catcher area locations 000000-001000, indicates time-out or illegal instruction hardware trap.

7.0 restrictions

7.1 external inputs

external inputs such as "slave in", "st1" and "st2" must not be connected to any customer hardware that might generate these signals while the diagnostic is running.

7.2 possible program "bombs"

the first two tests of this program check to see if the mnckw responds to the address the program thinks its at. if the mnckw does not respond, a bus error occurs. also bus errors can occur during the time the program sizes to see how many mnckw's are on your system.

for more information on the next subject, see jan. 1976 lsi-11 engieering bulletin issued by the digital components group.

bus errors may alter the preset contents of location 4 before the trap is executed, thereby transferring program control to area in the program that was not set up to handle the trap. if this happens, the program will "bomb" and possibly rewrite parts of itself.

8.0 miscellaneous

8.1 power fail

after a power failure occurs, the program execution will continue at the point where the power occurred. the program will type "power".

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 (logic test)

0.5 minutes (30 sec) iteration inhibited - no errors
2.5 minutes (150 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 loation r.
/	reopen the last location.
$n/ or rn/	open general register n(0-7) or s(ps register).
r;g or rg	goto location r and start program.
nl	execute bootstrap loader using n as device csr. console device is 177560.
;p 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 kd11f (lsi-11 module) can be entered in five different ways (others are a subset of these) from the run state:

1. execution of a 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 program software switch register is enabled if

1. no hardware swr exists;

2. if you start with all ones (swr=177777) in the switch register.

the software switch register may be changed 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 tester starting address

a special starting address has been provided for manufacturing to use to inform the program that the clock module is cabled to an in-house tester.

manual intervention is needed in this sequence of testing. the program will type out all instructions. a cable should connect j1 on the clock module to the tester. switches 1 and 3 of s2 (on the clock module) should be on, all other switches on s2 should be off.

8.8 logic test with a dwarf connected

more complete testing of the clocks i/o signals can be made if a dwarf module is connected to the clock. if you do this, select "d" to run the logic test with the dwarf tests enabled.

a series of instructions will be typed out for you to follow.

```
                                .TITLE  CVMNC-B MNCKW  DIAGNOSTIC
                                ;*COPYRIGHT (C) 1978
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY EDWARD C. BADGER AND SUBMITTED BY R. SHOOP
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
000001                          $TN=1

                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            INHIBIT SIZING THE # OF MNCKW'S
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            ENABLE LINE FREQUENCY RATE TESTING
                                ;*         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  021316  000200                  .WORD   IOTRD,200       ;HANDLE BUSS 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
                                                                ;IGNORING ITS INTERRUPTS.

000200                                  .=200
000200  000137  001530                  JMP     @#START         ;MAX. OF 8 UNITS
000204  000137  001550                  JMP     @#RESTRT        ;RESTART ADDRESS
000210  000137  001506                  JMP     TESTER          ;TESTER STARTING ADDRESS

                                .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
```

```
(1)  000011  HT=     11          ;;CODE FOR HORIZONTAL TAB
(1)  000012  LF=     12          ;;CODE FOR LINE FEED
(1)  000015  CR=     15          ;;CODE FOR CARRIAGE RETURN
(1)  000200  CRLF=   200         ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)  177776  PS=     177776      ;;PROCESSOR STATUS WORD
(1)          .EQUIV  PS,PSW
(1)  177774  STKLMT= 177774      ;;STACK LIMIT REGISTER
(1)  177772  PIRQ=   177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
(1)  177570  DSWR=   177570      ;;HARDWARE SWITCH REGISTER
(1)  177570  DDISP=  177570      ;;HARDWARE DISPLAY REGISTER
(1)
(1)          ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1)  000000  R0=     %0          ;;GENERAL REGISTER
(1)  000001  R1=     %1          ;;GENERAL REGISTER
(1)  000002  R2=     %2          ;;GENERAL REGISTER
(1)  000003  R3=     %3          ;;GENERAL REGISTER
(1)  000004  R4=     %4          ;;GENERAL REGISTER
(1)  000005  R5=     %5          ;;GENERAL REGISTER
(1)  000006  R6=     %6          ;;GENERAL REGISTER
(1)  000007  R7=     %7          ;;GENERAL REGISTER
(1)  000006  SP=     %6          ;;STACK POINTER
(1)  000007  PC=     %7          ;;PROGRAM COUNTER
(1)
(1)          ;*PRIORITY LEVEL DEFINITIONS
(1)  000000  PR0=    0           ;;PRIORITY LEVEL 0
(1)  000040  PR1=    40          ;;PRIORITY LEVEL 1
(1)  000100  PR2=    100         ;;PRIORITY LEVEL 2
(1)  000140  PR3=    140         ;;PRIORITY LEVEL 3
(1)  000200  PR4=    200         ;;PRIORITY LEVEL 4
(1)  000240  PR5=    240         ;;PRIORITY LEVEL 5
(1)  000300  PR6=    300         ;;PRIORITY LEVEL 6
(1)  000340  PR7=    340         ;;PRIORITY LEVEL 7
(1)
(1)          ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
(1)  100000  SW15=   100000
(1)  040000  SW14=   40000
(1)  020000  SW13=   20000
(1)  010000  SW12=   10000
(1)  004000  SW11=   4000
(1)  002000  SW10=   2000
(1)  001000  SW09=   1000
(1)  000400  SW08=   400
(1)  000200  SW07=   200
(1)  000100  SW06=   100
(1)  000040  SW05=   40
(1)  000020  SW04=   20
(1)  000010  SW03=   10
(1)  000004  SW02=   4
(1)  000002  SW01=   2
(1)  000001  SW00=   1
(1)          .EQUIV  SW09,SW9
(1)          .EQUIV  SW08,SW8
(1)          .EQUIV  SW07,SW7
(1)          .EQUIV  SW06,SW6
(1)          .EQUIV  SW05,SW5
(1)          .EQUIV  SW04,SW4
```

```
   (1)                          .EQUIV  SW03,SW3
   (1)                          .EQUIV  SW02,SW2
   (1)                          .EQUIV  SW01,SW1
   (1)                          .EQUIV  SW00,SW0
   (1)
   (1)                          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
   (1)          100000          BIT15=  100000
   (1)          040000          BIT14=  40000
   (1)          020000          BIT13=  20000
   (1)          010000          BIT12=  10000
   (1)          004000          BIT11=  4000
   (1)          002000          BIT10=  2000
   (1)          001000          BIT09=  1000
   (1)          000400          BIT08=  400
   (1)          000200          BIT07=  200
   (1)          000100          BIT06=  100
   (1)          000040          BIT05=  40
   (1)          000020          BIT04=  20
   (1)          000010          BIT03=  10
   (1)          000004          BIT02=  4
   (1)          000002          BIT01=  2
   (1)          000001          BIT00=  1
   (1)                          .EQUIV  BIT09,BIT9
   (1)                          .EQUIV  BIT08,BIT8
   (1)                          .EQUIV  BIT07,BIT7
   (1)                          .EQUIV  BIT06,BIT6
   (1)                          .EQUIV  BIT05,BIT5
   (1)                          .EQUIV  BIT04,BIT4
   (1)                          .EQUIV  BIT03,BIT3
   (1)                          .EQUIV  BIT02,BIT2
   (1)                          .EQUIV  BIT01,BIT1
   (1)                          .EQUIV  BIT00,BIT0
   (1)
   (1)                          ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
   (1)          000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
   (1)          000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
   (1)          000014          TBITVEC=14              ;;''T'' BIT
   (1)          000014          TRTVEC= 14              ;;TRACE TRAP
   (1)          000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
   (1)          000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
   (1)          000024          PWRVEC= 24              ;;POWER FAIL
   (1)          000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
   (1)          000034          TRAPVEC=34              ;;''TRAP'' TRAP
   (1)          000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
   (1)          000064          TPVEC=  64              ;;TTY PRINTER VECTOR
   (1)          000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
5653
5654            171020                  ABASE=  171020
5655            000440                  AVECT1= 440
5656            000200                  APRIOR= 200
5657
5658            167400                  $SWR=   167400
5659            000001                  $TN=    1
5660
5661                         .SBTTL  ACT11 HOOKS
 (1)
```

```
(2)                                ;;*****************************************************************
(1)                                ;HOOKS REQUIRED BY ACT11
(1)           000214                       $SVPC=.                    ;SAVE PC
(1)           000046                       .=46
(1)  000046   014704                       $ENDAD                     ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)           000052                       .=52
(1)  000052   000000                       .WORD   0                  ;;2)SET LOC.52 TO ZERO
(1)           000214                       .=$SVPC                    ;; RESTORE PC
5662          001000                       .=1000
5663                               .SBTTL  APT PARAMETER BLOCK
(1)
(2)                                ;;*****************************************************************
(1)                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                ;;*****************************************************************
(1)           001000                       .$X=.    ;;SAVE CURRENT LOCATION
(1)           000024                       .=24     ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1)  000024   000200                       200      ;;FOR APT START UP
(1)           000044                       .=44     ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1)  000044   001000                       $APTHDR  ;;POINT TO APT HEADER BLOCK
(1)           001000                       .=.$X    ;;RESET LOCATION COUNTER
(2)                                ;;*****************************************************************
(1)                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)                                ;INTERFACE SPEC.
(1)
(1)  001000                        $APTHD:
(1)  001000   000000               $HIBTS: .WORD   0        ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1)  001002   001174               $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1)  001004   000012               $TSTM:  .WORD   10.      ;;RUN TIM OF LONGEST TEST
(1)  001006   000170               $PASTM: .WORD   120.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1)  001010   000170               $UNITM: .WORD   120.     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1)  001012   000031                       .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
 5664                                   .SBTTL  COMMON TAGS
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
  (1)                                   ;*USED IN THE PROGRAM.
  (1)
  (1)           001100                          .=1100
  (1)  001100                           $CMTAG:                         ;;START OF COMMON TAGS
  (1)  001100  000000                           .WORD   0
  (1)  001102     000                   $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
  (1)  001103     000                   $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
  (1)  001104  000000                   $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
  (1)  001106  000000                   $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
  (1)  001110  000000                   $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
  (1)  001112  000000                   $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
  (1)  001114     000                   $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
  (1)  001115     001                   $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
  (1)  001116  000000                   $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
  (1)  001120  000000                   $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
  (1)  001122  000000                   $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
  (1)  001124  000000                   $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
  (1)  001126  000000                   $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
  (1)  001130  000000                           .WORD   0               ;;RESERVED--NOT TO BE USED
  (1)  001132  000000                           .WORD   0
  (1)  001134     000                   $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
  (1)  001135     000                   $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
  (1)  001136  000000                           .WORD   0
  (1)  001140  177570                   SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
  (1)  001142  177570                   DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
  (1)  001144  177560                   $TKS:   177560                  ;;TTY KBD STATUS
  (1)  001146  177562                   $TKB:   177562                  ;;TTY KBD BUFFER
  (1)  001150  177564                   $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
  (1)  001152  177566                   $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
  (1)  001154     000                   $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
  (1)  001155     002                   $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
  (1)  001156     012                   $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
  (1)  001157     000                   $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
  (1)  001160  000000                   $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
  (1)  001162  000000                   $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
  (1)  001164  177607  000377           $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
  (1)  001170     077                   $QUES:  .ASCII  /?/             ;;QUESTION MARK
  (1)  001171     015                   $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
  (1)  001172  000012                   $LF:    .ASCIZ  <12>            ;;LINE FEED
  (2)                                   ;;**********************************************************************
  (2)                                   .SBTTL  APT MAILBOX-ETABLE
  (2)
  (3)                                   ;;**********************************************************************
  (2)                                   .EVEN
  (2)  001174                           $MAIL:                          ;;APT MAILBOX
  (2)  001174  000000                   $MSGTY: .WORD   AMSGTY           ;;MESSAGE TYPE CODE
  (2)  001176  000000                   $FATAL: .WORD   AFATAL           ;;FATAL ERROR NUMBER
  (2)  001200  000000                   $TESTN: .WORD   ATESTN           ;;TEST NUMBER
  (2)  001202  000000                   $PASS:  .WORD   APASS            ;;PASS COUNT
  (2)  001204  000000                   $DEVCT: .WORD   ADEVCT           ;;DEVICE COUNT
  (2)  001206  000000                   $UNIT:  .WORD   AUNIT            ;;I/O UNIT NUMBER
  (2)  001210  000000                   $MSGAD: .WORD   AMSGAD           ;;MESSAGE ADDRESS
```

```
(2)  001212  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001214                         $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001214     000                 $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001215     000                 $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001216  000000                 $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001220  000000                 $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001222  000000                 $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                 ;*                      BITS 15-11=CPU TYPE
(2)                                 ;*                              11/04=01,11/05-02,11/20=03,11/40=04,11/45=05
(2)                                 ;*                              11/70=06,PDQ=07,Q=10
(2)                                 ;*                      BIT 10=REAL TIME CLOCK
(2)                                 ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                 ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001224     000                 $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001225     000                 $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                 ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                 ;*                              900 NSEC CORE=001
(2)                                 ;*                              300 NSEC BIPOLAR=002
(2)                                 ;*                              500 NSEC MOS=003
(2)  001226  000000                 $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                 ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
(2)  001230     000                 $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001231     000                 $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001232  000000                 $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001234     000                 $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001235     000                 $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001236  000000                 $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001240     000                 $MAMS4: .BYTE   AMAMS4  ;;H ADDRESS,M.S.BYTE
(2)  001241     000                 $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001242  000000                 $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001244  000440                 $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001246  000000                 $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001250  171020                 $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001252  000000                 $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001254  000000                 $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001256                         $ETEND:
(2)                                 .MEXIT
```

```
  (1)                                   .SBTTL  ERROR POINTER TABLE
  (1)
  (1)                                   ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
  (1)                                   ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
  (1)                                   ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
  (1)                                   ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
  (1)                                   ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
  (1)
  (1)                                   ;*      EM              ;;POINTS TO THE ERROR MESSAGE
  (1)                                   ;*      DH              ;;POINTS TO THE DATA HEADER
  (1)                                   ;*      DT              ;;POINTS TO THE DATA
  (1)                                   ;*      DF              ;;POINTS TO THE DATA FORMAT
  (1)
  (1)
  (1)  001256                           $ERRTB:
 5665
 5670                                   ;ITEM   1
 5671  001256  021662                          EM1                     ;CLOCK   SR FUNCTION ERROR
 5672  001260  022604                          DH1                     ;ERRPC   ASR     WAS     S/B
 5673  001262  024262                          DT1                     ;$ERRPC,ASR,$BDDAT,$GDDAT
 5674  001264  024402                          DF0                     ;ALL NUMBERS ARE IN OCTAL FORM
  (1)
 5675                                   ;ITEM   2
 5676  001266  021730                          EM2                     ;CLOCK   SR DATA ERROR
 5677  001270  022604                          DH1                     ;ERRPC   ASR     WAS     S/B
 5678  001272  024262                          DT1                     ;$ERRPC,ASR,$BDDAT,$GDDAT
 5679  001274  024402                          DF0                     ;ALL NUMBERS ARE IN OCTAL FORM
  (1)
 5680                                   ;ITEM   3
 5681  001276  021772                          EM3                     ;CLOCK  BR DATA ERROR
 5682  001300  022651                          DH3                     ;ERRPC   ABR     WAS
 5683  001302  024304                          DT3                     ;$ERRPC,ABR,$BDDAT,$GDDAT
 5684  001304  024402                          DF0                     ;ALL NUMBERS ARE IN OCTAL FORM
  (1)
 5685                                   ;ITEM   4
 5686  001306  022034                          EM4                     ;INTERRUPT ERROR.
 5687  001310  022702                          DH4A                    ;ERRPC   TO      ROM ADDR.
 5688  001312  024320                          DT4                     ;$ERRPC,        TRTO,TRFRO
 5689  001314  024402                          DF0                     ;ALL NUMBERS ARE IN OCTAL FORM
  (1)
 5690                                   ;ITEM   5
 5691  001316  022073                          EM5                     ;CLOCK   COUNT REG ERROR
 5692  001320  022604                          DH1                     ;ERRPC   ASR   WAS   S/B
 5693  001322  024262                          DT1                     ;$ERRPC, ACR, $BDDAT, $GDDAT
 5694  001324  024402                          DF0                     ;ALL NUMBERS ARE IN OCTAL FORM
  (1)
 5695                                   ;ITEM   6
 5696  001326  022250                          EM12                    ;CLOCK   COUNT FUNCTION ERROR
 5697  001330  022743                          DH12                    ;ERRPC   ASR
 5698  001332  024332                          DT12                    ;ERRPC, ASR
 5699  001334  024402                          DF0                     ;ALL NUMBERS ARE IN OCTAL FORM
  (1)
```

```
                                                                I  2
5701                                    ;ITEM   7
5702  001336  022361                            EM16            ;CLOCK   INTERRUPT ERROR
5703  001340  022743                            DH12            ;ERRPC   ASR
5704  001342  024332                            DT12            ;$ERRPC, ASR
5705  001344  024402                            DF0             ;ALL NUMBERS ARE IN OCTAL FORM
 (1)
5706                                    ;ITEM   10
5707  001346  022430                            EM20            ;CLOCK   REPEATABILITY ERROR
5708  001350  022765                            DH20            ;ERROR   ASR   2ND CNT 1ST CNT  3RD CNT
5709  001352  024342                            DT20            ;$ERRPC, ASR, $BDDAT, $GDDAT, $TMP0
5710  001354  024402                            DF0             ;ALL NUMBERS ARE IN OCTAL FORM
 (1)
5711                                    ;ITEM   11
5712  001356  022213                            EM11            ;CLOCK   COUNT ERROR
5713  001360  022604                            DH1             ;ERRPC   ASR   WAS   S/B
5714  001362  024356                            DT22            ;$ERRPC, ASR, $BDDAT, $TMP0
5715  001364  024402                            DF0             ;ALL NUMBERS ARE IN OCTAL FORM
 (1)
5716                                    ;ITEM   12
5717  001366  022475                            EM26            ;CLOCK ADDRESSING ERROR
5718  001370  023025                            DH26            ;ERRPC   CLOCK ADDR.
5719  001372  024372                            DT26            ;$ERRPC,$TMP0
5720  001374  024402                            DF0             ;ALL NUMBERS ARE IN OCTAL FORM
 (1)
5721                                    ;ITEM   13
5722  001376  022315                            EM13            ;CLOCK I.D. LINES ERROR
5723  001400  022604                            DH1             ;ERRPC  ASR     WAS     S/B
5724  001402  024262                            DT1             ;$ERRPC ASR     $BDDAT  $GDDAT
5725  001404  024402                            DF0             ;ALL NUMBERS ARE IN OCTAL FORM
 (1)
5726                                    ;ITEM   14
5727  001406  022134  022635  024276            EM6,DH2,DT2,DF0 ;EXISTING UNIT FAILS TO RESPOND
      001414  024402
5728
5729  001416  177546                  KWL:      177546          ;LINE CLOCK ADDRESS
5730
5731  001420  171020                  ASR:      .WORD   ABASE
5732  001422  171022                  ABR:      .WORD   ABASE+2
5733  001424  000440                  VECT1:    .WORD   AVECT1
5734  001426  000442                  VECTP:    .WORD   AVECT1+2
5735  001430  000444                  VECT2:    .WORD   AVECT1+4  ;VECTOR ADDR. OF ST2 INTRS.
5736  001432  000446                  VECT2P:   .WORD   AVECT1+6
5737  001434  000200                  PRIOR:    .WORD   APRIOR
5738
5739  001436  167774                  DR:       .WORD   167774
5740  001440  167772                  DR2:      .WORD   167772
5741  001442  170430                  TSCLC:    .WORD   170430  ;ADR. OF TESTOR CLOCK
5742  001444  170432                  TSCLD:    .WORD   170432  ;BUFFER PRESET REG.
5743  001446  000000                  $TMP0:    .WORD   0       ;TEMP STORAGE.
5744  001450  000000                  MASKNM:   0               ;CURRENT UNIT MASK BIT
5745  001452  000000                  BADUNT:   0               ;WORKING LOC. TO DETERMINE BAD UNIT #
5746  001454  000004                  VADDR:    4               ;INCREMENT TO THE NEXT MNCKW BASE ADDRESS
5747  001456  000010                  VVECTR:   10              ;INCREMENT TO THE NEXT MNCKW VECTOR ADDRESS
5748  001460  000000                  EVER:     0               ;INDICATOR IF THE UNIT COUNT HAS BEEN REPORTED
5749  001462  000000                  ROTATE:   .WORD   0               ;POINT TO DEVICE UNDER TEST.
5750  001464  000000                  UTEST:    .WORD   0               ;KEEPS TRACK OF GOOD UNITS.
```

```
 5751  001466  000000                    ERCNT:  .WORD   0                       ;COUNTS ERRORS.
 5752  001470  000000                    MDEVCT: .WORD   0                       ;COUNTS DEVICES TESTED.
 5753  001472  000000                    TSTCNT: .WORD   0                       ;MAX DEVICES TO BE TESTED.
 5754  001474  000000                    LCNT:   .WORD   0                       ;TOTAL UNITS TESTED.
 5755  001476  000000                    DWARF:  .WORD   0       ;INDICATE IF TESTOR/DWARF MODULE PRESENT
 5756                                                            ; (=1,YES DWARF,   =BIT15, YES TESTOR)
 5757  001500  000000                    ASK:    .WORD   0                       ;=1 WHEN QUESTION ASKED IN RUN.
 5758  001502  000000                    TEMP1:  0                               ;RESTART INDICATOR
 5759  001504  000000                    UNITBD: 0
 5760
 5762          001506                    TESTER=.
 5763  001506  005037  001502                    CLR     TEMP1           ;CLEAR RESTART FLAG
 5764  001512  012737  000001  001472            MOV     #1,TSTCNT       ;LOAD MAX UNIT COUNT
 5765  001520  012737  100000  001476            MOV     #BIT15,DWARF    ;INDICATE TESTER CONNECTED
 5766  001526  000412                    BR      1$
 5767          001530                    START=.
 5768  001530  012737  000010  001472            MOV     #8.,TSTCNT      ;TEST UP TO 8 UNITS.
 5769  001536  005037  001502                    CLR     TEMP1           ;CLEAR RESTART FLAG
 5770  001542  005037  001476                    CLR     DWARF           ;NO TESTER OR DWARF CONNECTED
 5771  001546  000402                    BR      1$
 5772          001550                    RESTRT=.
 5773  001550  005237  001502                    INC     TEMP1           ;SET RESTART FLAG
 5774  001554  000005                    1$:     RESET
 5775                                    .SBTTL  INITIALIZE THE COMMON TAGS
  (1)                                    ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
  (1)  001556  012706  001100                    MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
  (1)  001562  005026                    CLR     (R6)+           ;;CLEAR MEMORY LOCATION
  (1)  001564  022706  001140                    CMP     #SWR,R6 ;;DONE?
  (1)  001570  001374                    BNE     .-6             ;;LOOP BACK IF NO
  (1)  001572  012706  001100                    MOV     #STACK,SP       ;;SETUP THE STACK POINTER
  (1)                                    ;;INITIALIZE A FEW VECTORS
  (1)  001576  012737  016476  000020            MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
  (1)  001604  012737  000340  000022            MOV     #340,@#IOTVEC+2 ;;LEVEL 7
  (1)  001612  012737  016066  000030            MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
  (1)  001620  012737  000340  000032            MOV     #340,@#EMTVEC+2 ;;LEVEL 7
  (1)  001626  012737  021576  000034            MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
  (1)  001634  012737  000340  000036            MOV     #340,@#TRAPVEC+2;LEVEL 7
  (1)  001642  012737  021140  000024            MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
  (1)  001650  012737  000340  000026            MOV     #340,@#PWRVEC+2 ;;LEVEL 7
  (1)  001656  005037  001160                    CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
  (1)  001662  005037  001162                    CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
  (1)  001666  112737  000001  001115            MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
  (1)  001674  012737  001674  001106            MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
  (1)  001702  012737  001702  001110            MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
  (2)                                    ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
  (2)                                    ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
  (2)  001710  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
  (2)  001714  012737  001750  000004            MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
  (2)  001722  012737  177570  001140            MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
  (2)  001730  012737  177570  001142            MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
  (2)  001736  022777  177777  177174            CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
  (2)  001744  001012                    BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
  (2)                                                            ;;AND  THE HARDWARE SWR IS NOT = -1
  (2)  001746  000403                    BR      65$             ;;BRANCH IF NO TIMEOUT
  (2)  001750  012716  001756    64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
  (2)  001754  000002                    RTI
```

```
   (2)  001756  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
   (2)  001764  012737  000174  001142           MOV     #DISPREG,DISPLAY
   (2)  001772  012637  000004           66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
   (1)
   (2)  001776  005037  001202                   CLR     $PASS           ;;CLEAR PASS COUNT
   (2)  002002  132737  000200  001215           BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
   (2)  002010  001403                            BEQ     67$             ;;YES,USE NON-APT SWITCH
   (2)  002012  012737  001216  001140           MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
   (2)  002020                           67$:
  5776                                   ;ROUTINE TO OVERLAY 4 LOC OF THE '$TYPE'' ROUTINE
  5777  002020  012737  005046  020410           MOV     #5046,$TYPE     ;LOWER PS
  5778  002026  012737  012746  020412           MOV     #12746,$TYPE+2
  5779  002034  012737  020422  020414           MOV     #$TYPE+12,$TYPE+4
  5780  002042  012737  000002  020416           MOV     #RTI,$TYPE+6
  5781  002050  004737  017030                   JSR     PC,$TKINT       ;ENABLE TKB INTR.
  5782
   (1)  002054  012746  000000                   MOV     #0,-(SP)                ;SET CPU PRIORITY ON RETERN.
   (1)  002060  012746  002066                   MOV     #68$,-(SP)              ;SHOW RETURN ADDRESS.
   (1)  002064  000002                           RTI                             ;CAUSE A RETURN(PUTS STATUS IN STATUS REG.).
   (1)  002066                           68$:
   (1)
  5783  002066  005037  001204                   CLR     $DEVCT          ;ZERO DEVICE COUNT.
  5784  002072  012737  021316  000004           MOV     #IOTRD,@#ERRVEC ;FIX TRAP CATCHER.
  5785  002100  012737  000340  000006           MOV     #340,@#ERRVEC+2
  5786                                   .SBTTL  TYPE PROGRAM NAME
   (1)                                   ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
   (1)  002106  005227  177777                   INC     #-1             ;;FIRST TIME?
   (1)  002112  001045                           BNE     69$             ;;BRANCH IF NO
   (1)  002114  104401  002162                   TYPE    ,70$            ;;TYPE ASCIZ STRING
   (2)                                   .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
   (2)  002120  005737  000042                   TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
   (2)  002124  001012                           BNE     71$             ;;BRANCH IF YES
   (2)  002126  123727  001214  000001           CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
   (2)  002134  001406                           BEQ     71$             ;;BRANCH IF YES
   (2)  002136  023727  001140  000176           CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
   (2)  002144  001005                           BNE     72$             ;;BRANCH IF NO
   (2)  002146  104407                           GTSWR                   ;;GET SOFT-SWR SETTINGS
   (2)  002150  000403                           BR      72$
   (2)  002152  112737  000001  001134  71$:     MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
   (2)  002160                           72$:
   (1)  002160  000422                           BR      69$             ;;GET OVER THE ASCIZ
   (1)                                   ;;70$:  .ASCIZ  <CRLF>#CVMNC-B  MNCKW (CLOCK) DIAGNOSTIC#<CRLF>
   (1)  002226                           69$:
  5787  002226  105737  001134                   TSTB    $AUTOB          ;TEST IF UNDER A MONITOR
  5788  002232  001407                           BEQ     50$             ;BR IF NOT
  5789  002234  005037  001476                   CLR     DWARF           ;CLEAR DWARF FLAG
  5790  002240  012737  000010  001472           MOV     #8.,TSTCNT      ;LOAD MAX. # OF UNITS
  5791  002246  000137  002464                   JMP     LOGIC           ;RUN LOGIC TEST
  5792  002252  005737  001502           50$:    TST     TEMP1           ;TEST IF RESTARTING
  5793  002256  001004                           BNE     MTEST1          ;BR IF YES
```

```
5795                                    .SBTTL  KEYBOARD COMMAND DECODER
5796  002260  104401  023056            MTEST:  TYPE,   PRIME0          ;INFORM THE OPER. OF THE TESTS
5797  002264  104401  023231                    TYPE,   PRIME1
5798  002270  000005                    MTEST1: RESET
5799  002272  052777  000100  176644            BIS     #BIT6,@$TKS     ;ENABLE TKB INTR.
5800  002300  105037  001476                    CLRB    DWARF           ;INDICATE NO DWARF CONNECTED
5801  002304  005037  001202                    CLR     $PASS           ;INIT. PASS COUNTER
5802  002310  005037  001112                    CLR     $ERTTL          ;INIT. TOTAL ERROR COUNT
5803  002314  005037  001460                    CLR     EVER            ;INIT. # OF UNIT TYPEOUT
5804  002320  004737  003002                    JSR     PC,PRIADR       ;INIT THE ADDR. AND VECTOR
5805  002324  104401  023565                    TYPE,   DOT
5806  002330  104412                            RDLIN                   ;GET OPER. INPUT
5807  002332  013637  002462                    MOV     @(SP)+,RUNIT    ;GET 1ST CHAR
5808  002336  142737  000040  002462            BICB    #40,RUNIT       ;ENSURE UPPER CASE
5809  002344  122737  000102  002462            CMPB    #'B,RUNIT       ;TEST IF 'B'
5810  002352  001002                            BNE     1$              ;BR IF NOT
5811  002354  000137  015024                    JMP     BASEXC          ;CHANGE BASE OR VECTOR ADDRESS
5812  002360  122737  000104  002462  1$:       CMPB    #'D,RUNIT       ;TEST IF 'D'
5813  002366  001007                            BNE     3$              ;BR IF NOT
5814  002370  105237  001476                    INCB    DWARF           ;SET DWARF FLAG
5815  002374  012737  000001  001472            MOV     #1,TSTCNT       ;INDICATE ONLY 1 UNIT WITH DWARF
5816  002402  000137  002464                    JMP     LOGIC           ;RUN LOGIC TEST WITH DWARFS
5817  002406  122737  000107  002462  3$:       CMPB    #'G,RUNIT       ;TEST IF 'G'
5818  002414  001002                            BNE     5$              ;BR IF NOT
5819  002416  104407                            GTSWR                   ;GET SWITCH VALUE
5820  002420  000723                            BR      MTEST1          ;AND RETYPE DOT
5821  002422  122737  000110  002462  5$:       CMPB    #'H,RUNIT       ;TEST IF 'H'
5822  002430  001713                            BEQ     MTEST           ;BR IF YES
5823  002432  122737  000114  002462  6$:       CMPB    #'L,RUNIT       ;TEST IF 'L'
5824  002440  001005                            BNE     77$             ;BR IF NOT
5825  002442  012737  000010  001472            MOV     #8.,TSTCNT      ;LOAD # OF UNITS
5826  002450  000137  002464                    JMP     LOGIC           ;RUN LOGIC TEST WITH NO DWARFS
5827  002454  104401  001170            77$:    TYPE,   $QUES           ;TYPE '?'
5828  002460  000703                            BR      MTEST1          ;AND RETYPE DOT
5829  002462  000000            RUNIT:  0               ;CHAR. THE OPER. TYPED IN
```

```
                                                .SBTTL  DETERMINE THE NUMBER OF MNCKW'S ON THE SYSTEM
002464  013737  001250  001126  LOGIC:  MOV     $BASE,$BDDAT    ;GET BASE ADDRESS
002472  005037  001450                  CLR     MASKNM
002476  005037  001206                  CLR     $UNIT           ;CLEAR UNIT NUMBER
002502  012737  002556  000004          MOV     #2$,ERRVEC      ;LOAD RETURN ADDRESS
002510  005777  176412          1$:     TST     @$BDDAT         ;TEST IF ADDRESS EXISTS
002514  063737  001454  001126          ADD     VADDR,$BDDAT    ;UPDATE BUS ADDRESS
002522  005237  001206                  INC     $UNIT           ;UPDATE UNIT COUNT
002526  005737  001214                  TST     $ENV            ;TEST IF 'DO NOT SIZE''
002532  100423                          BMI     3$              ;BR IF NO SIZEING
002534  032777  010000  176376          BIT     #SW12,@SWR      ;TEST IF SW 12 IS SET
002542  001017                          BNE     3$              ;BR IF INHIBIT SIZING SWITCH IS SET
002544  023737  001472  001206          CMP     TSTCNT,$UNIT    ;TEST IF MAX NUMBER
002552  001356                          BNE     1$              ;BR IF NOT
002554  000412                          BR      3$              ;BR IF MAX
002556  022626                  2$:     CMP     (SP)+,(SP)+     ;RESTORE STACK
002560  005737  001206                  TST     $UNIT           ;TEST IF ANY EXIST
002564  001006                          BNE     3$              ;BR IF ANY ARE THERE
002566  005737  000042                  TST     @#42            ;TEST IF XXDP CHAIN MODE
002572  001003                          BNE     3$              ;BR IF YES
002574  104012                          ERROR   12              ;BASE ADDRESS CAUSED A BUS TRAP
002576  000137  014616                  JMP     $EOP
002602  012737  021316  000004  3$:     MOV     #IOTRD,ERRVEC   ;RESTORE ERROR VECTOR
002610  012737  000200  000006          MOV     #200,ERRVEC+2
002616  005737  001460                  TST     EVER            ;TEST IF # HAS BEEN REPORTED
002622  100426                          BMI     4$              ;BR IF IT HAS
002624  005737  001476                  TST     DWARF           ;TEST IF TESTER MODE
002630  100414                          BMI     6$              ;BR IF TESTER
002632  104401  023760                  TYPE    ,FOUND1         ;TELL OPERATOR # OF MNCKW FOUND
002636  013746  001206                  MOV     $UNIT,-(SP)
002642  104405                          TYPDS
002644  104401  024003                  TYPE    ,FOUND2
002650  005737  001206                  TST     $UNIT           ;TEST IF ANY UNITS
002654  001002                          BNE     6$              ;BR IF SOME
002656  000137  014616                  JMP     $EOP            ;REPORT EOP
002662  013737  001206  001460  6$:     MOV     $UNIT,EVER      ;SAVE THE # OF MNCKW'S FOR LATER
002670  052737  100000  001460          BIS     #BIT15,EVER     ;SET 'REPORTED NUMBER FLAG''
002676  000410                          BR      5$
002700  123737  001460  001206  4$:     CMPB    EVER,$UNIT      ;TEST IF ANY HAVE GONE AWAY
002706  001404                          BEQ     5$              ;BR IF ALL ARE STILL THERE
002710  113737  001460  001450          MOVB    EVER,MASKNM     ;SAVE FOR ERROR REPORT
002716  104014                          ERROR   14              ;EXISTING UNIT FAILED TO RESPOND
002720  005037  001206          5$:     CLR     $UNIT           ;RESET UNIT POINTER
002724  004737  003002                  JSR     PC,PRIADR       ;LOAD BUS ADDRESSES
002730  012737  000001  001450          MOV     #BIT0,MASKNM    ;LOAD DEVICE MASK
002736  005037  001452                  CLR     BADUNT          ;RESET BAD UNIT INDICATOR
002742  005037  001500                  CLR     ASK
002746  005037  001470                  CLR     MDEVCT          ;TESTING FIRST UNIT.
002752  005037  001466                  CLR     ERCNT           ;NO ERRORS.
002756  012737  000001  001462          MOV     #1,ROTATE       ;POINT TO FIRST UNIT.
002764  013737  001462  001464          MOV     ROTATE,UTEST
002772  005046                          CLR     -(SP)   ;ONE WAY TO LOWER THE CPU INTR. LEVEL
002774  012746  003106                  MOV     #TST1,-(SP)
003000  000002                          RTI
```

```
                                        .SBTTL  SUBROUTINE TO PRIME THE BASE AND VECTOR VALUES
003002  013737  001250  001420  PRIADR: MOV     $BASE,ASR       ;LOAD INITIAL BASE ADDR
003010  013737  001244  001424          MOV     $VECT1,VECT1    ;LOAD INITIAL VECTOR ADDR
003016  042737  170000  001424  FIXADR: BIC     #170000,VECT1   ;CLEAR OUT PRIORITY BITS.
003024  013737  001424  001426          MOV     VECT1,VECTP     ;NOW FIX VECTOR +2 ADDR.
003032  062737  000002  001426          ADD     #2,VECTP
003040  013737  001424  001430          MOV     VECT1,VECT2     ;LETS FIX ST2 VECTOR ADDR.
003046  062737  000004  001430          ADD     #4,VECT2        ;ITS 4 GREATER THEN THE 1ST.
003054  013737  001430  001432          MOV     VECT2,VECT2P    ;VECTOR +2 ADDR.
003062  062737  000002  001432          ADD     #2,VECT2P
003070  013737  001420  001422          MOV     ASR,ABR         ;FIX ADDR OF PRESET REG=
003076  062737  000002  001422          ADD     #2,ABR          ;CSR + 2
003104  000207                        RTS     PC              ;RETURN
                                    ;;**************************************************************
                                    ;*TEST 1         *TEST THE I.D. LINE CODE IF ON THE TESTER
                                    ;;**************************************************************
003106  000004                      TST1:   SCOPE
003110  005737  001476                      TST     DWARF           ;TEST IF ON THE TESTER
003114  100020                        BPL     TST2            ;;BR IF NOT
003116  005077  176316                      CLR     @DR2            ;ENSURE TESTER STATUS
003122  017737  176310  001126        MOV     @DR,$BDDAT      ;READ TESTER REG. <I.D. LINES>
003130  042737  177417  001126        BIC     #177417,$BDDAT  ;MASK OFF OTHER BITS
003136  012737  000120  001124        MOV     #120,$GDDAT     ;LOAD EXPECTED VALUE
003144  023737  001124  001126        CMP     $GDDAT,$BDDAT   ;COMPARE THE VALUES
003152  001401                        BEQ     TST2            ;;BR IF SAME


                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003154  104013                              ERROR   13              ;REPORT INCORRECT I.D. LINE CODE
                                                                    ;MUST BE WRONG BOARD OR I.D. FAILURE

                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
 5949
  (1)
  (1)
  (5)                                   ;;*****************************************************************
  (4)                                   ;*TEST 2          *TEST THE ADDRESSABILITY OF CLOCK CSR
  (4)                                   ;;*****************************************************************
  (3)  003156  000004                   TST2.   SCOPE
  (2)  003160  012737  000050  001160           MOV     #50,$TIMES      ;;DO 50 ITERATIONS
  (2)  003166  012737  003202  001106           MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
  (1)  003174  012737  003202  001110           MOV     #1$,$LPERR
  (1)
  (1)
  (1)
  (1)  003202  013746  000004           1$:     MOV     @#ERRVEC,-(SP)  ;SAVE CONTENTS OF ADDRS 6.
  (1)  003206  012737  003222  000004           MOV     #2$,@#ERRVEC    ;SET TIME-OUT TRAP VECTOR TO HANDLER IN CASE.
  (1)                                                                   ;WE TIME-OUT WHEN ADDRESSING THE KW11.
  (1)  003214  005777  176200                   TST     @ASR            ;ADDRESS THE CLOCK!
  (1)                                                                   ;IF CLOCK DOES NOT RETURN
  (1)                                                                   ;'BUS SSYN' THEN WE'LL TIME-OUT
  (1)
  (1)  003220  000412                           BR      3$              ;THE CLOCK WAS THERE! EXIT SUB-TEST.
  (1)  003222                           2$:
  (2)  003222  062706  000004                   ADD     #4,SP                   ;/ADD #4 TO STACK POINTER.
  (1)  003226  013737  001420  001446           MOV     ASR,$TMP0       ;FOR ERROR TYPEOUT.
  (1)
  (2)

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  003234  104012                           ERROR   12              ;REPORT ERROR=CLOCK CSR FAILED TO RETURN
  (1)                                                                   ;'BUS SSYN' WHEN ADDRESSED.
  (1)  003236  012637  000004                   MOV     (SP)+,@#ERRVEC  ;NOTE: IF PROGRAM HAS INCORRECT
  (1)  003242  000137  014616                   JMP     $EOP            ;ADDRESS THEN WE MIGHT NOT BE
  (1)                                                                   ;TALKING TO THE CLOCK. MAKE SURE
  (1)                                                                   ;OF CLOCK ADDRESS.
  (1)
  (2)

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  003246  012637  000004           3$:     MOV     (SP)+,@#ERRVEC
  (1)
```

```
 5951
  (5)                                ;;********************* ************************************************
  (4)                                ;*TEST 3          *TEST THE ADDRESSABILITY OF CLOCK BUFFER REG.
  (4)                                ;;*****************************************************************
  (3)  003252  000004                TST3:   SCOPE
  (1)
  (1)
  (1)  003254  013746  000004        1$:     MOV     @#ERRVEC,-(SP)  ;SAVE CONTENTS OF ADDRS 6.
  (1)  003260  012737  003274  000004        MOV     #2$,@#ERRVEC    ;SET TIME-OUT TRAP VECTOR  O HANDLER IN CASE.
  (1)                                                                ;WE TIME-OUT WHEN ADDRESSING THE KW11.
  (1)  003266  005777  176130                TST     @ABR            ;ADDRESS THE CLOCK!
  (1)                                                                ;IF CLOCK DOES NOT RETURN
  (1)                                                                ;'BUS SSYN'' THEN WE'LL TIME-OUT
  (1)
  (1)  003272  000412                        BR      3$              ;THE CLOCK WAS THERE! EXIT SUB-TEST.
  (1)  003274                        2$:
  (2)  003274  062706  000004                ADD     #4,SP                   ;/ADD #4 TO STACK POINTER.
  (1)  003300  013737  001422  001446        MOV     ABR,$TMPO       ;FOR ERROR TYPEOUT.
  (1)
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  003306  104012                        ERROR   12              ;REPORT ERROR=CLOCK BUFFER REG. FAILED TO RETURN
  (1)                                                                ;'BUS SSYN'' WHEN ADDRESSED.
  (1)  003310  012637  000004                MOV     (SP)+,@#ERRVEC  ;NOTE: IF PROGRAM HAS INCORRECT
  (1)  003314  000137  014616                JMP     $EOP            ;ADDRESS THEN WE MIGHT NOT BE
  (1)                                                                ;TALKING TO THE CLOCK. MAKE SURE
  (1)                                                                ;OF CLOCK ADDRESS.
  (1)
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  003320  012637  000004        3$:     MOV     (SP)+,@#ERRVEC
  (1)
 5959
 5997
 5998
```

```
(1)                                             ;/#
(5)                                         ;;********************* ************************************************
(4)                                         ;*TEST 4          *TEST THAT CLOCK A STATUS REGISTER BIT 14 CAN BE SET AND CLEARED
(5)                                         ;*
(5)                                         ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
(5)                                         ;*F/FS OR GATES
(5)                                         ;*
(5)
(4)                                         ;;*****************************************************************
(3)  003324  000004                         TST4:   SCOPE
(2)  003326  012737  000100  001160                 MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  003334  005077  176060                 CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  003340  052777  040000  176052         BIS     #BIT14,@ASR     ;/SET BIT 14.
(1)  003346  012737  040000  001124         MOV     #BIT14,$GDDAT   ;/SET FOR ERROR TYPEOUT S/B.
(1)  003354  017737  176040  001126         MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  003362  023737  001124  001126         CMP     $GDDAT,$BDDAT   ;/DID BIT 14 AND ONLY BIT 14 SET?
(1)  003370  001402                         BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)
                                            ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003372  104002                                 ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                                         ;/BIT 14 FAILED TO BIT SET.
(2)
                                            ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003374  000412                                 BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  003376  042777  040000  176014  1$:    BIC     #BIT14,@ASR     ;/TRY CLEARING BIT 14.
(1)  003404  005037  001124                 CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  003410  017737  176004  001126         MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  003416  001401                         BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)
                                            ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003420  104002                                 ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                                         ;/BIT 14 FAILED TO CLEAR.
(1)
(2)
                                            ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003422                                 2$.
(1)
5999
```

```
                                        ;/#
                                ;;********************* ***********************************************
                                ;*TEST 5         *TEST THAT CLOCK A STATUS REGISTER BIT 13 CAN BE SET AND CLEARED
                                ;*
                                ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
                                ;*F/FS OR GATES
                                ;*

                                ;;*****************************************************************
003422  000004                TST5:   SCOPE
003424  012737  000100  001160        MOV     #100,$TIMES     ;;DO 100 ITERATIONS

003432  005077  175762              CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
003436  052777  020000  175754      BIS     #BIT13,@ASR     ;/SET BIT 13.
003444  012737  020000  001124      MOV     #BIT13,$GDDAT   ;/SET FOR ERROR TYPEOUT S/B.
003452  017737  175742  001126      MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
003460  023737  001124  001126      CMP     $GDDAT,$BDDAT   ;/DID BIT 13 AND ONLY BIT 13 SET?
003466  001402                      BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003470  104002                      ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
                                                        ;/BIT 13 FAILED TO BIT SET.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003472  000412                      BR      2$              ;/BR TO END SUBTEST.

003474  042777  020000  175716  1$: BIC     #BIT13,@ASR     ;/TRY CLEARING BIT 13.
003502  005037  001124              CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
003506  017737  175706  001126      MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
003514  001401                      BEQ     2$              ;/IF ZERO - NO ERROR!

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003516  104002                      ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
                                                        ;/BIT 13 FAILED TO CLEAR.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003520                          2$:
6000
```

```
                                        ;/#
                                ;;**********  **********************************************
                                ;*TEST 5         *TEST THAT CLOCK A STATUS REGISTER BIT 11 CAN BE SET AND CLEARED
                                ;*
                                ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
                                ;*F/FS OR GATES
                                ;*
                                ;;**************************************************************
003520  000004                  TST6:   SCOPE
003522  012737  000100  001160          MOV     #100,$TIMES     ;;DO 100 ITERATIONS

003530  005077  175664                  CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
003534  052777  004000  175656          BIS     #BIT11,@ASR     ;/SET BIT 11.
003542  012737  004000  001124          MOV     #BIT11,$GDDAT   ;/SET FOR ERROR TYPEOUT S/B.
003550  017737  175644  001126          MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
003556  023737  001124  001126          CMP     $GDDAT,$BDDAT   ;/DID BIT 11 AND ONLY BIT 11 SET?
003564  001402                          BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003566  104002                          ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
                                                                ;/BIT 11 FAILED TO BIT SET.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003570  000412                          BR      2$              ;/BR TO END SUBTEST.

003572  042777  004000  175620  1$:     BIC     #BIT11,@ASR     ;/TRY CLEARING BIT 11.
003600  005037  001124                  CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
003604  017737  175610  001126          MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
003612  001401                          BEQ     2$              ;/IF ZERO - NO ERROR!

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003614  104002                          ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
                                                                ;/BIT 11 FAILED TO CLEAR.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

003616                          2$:
6001
```

```
(1)                                     ;/*
(5)                                     ;;**********************************************************
(4)                                     ;*TEST 7         *TEST THAT CLOCK A STATUS REGISTER BIT 6 CAN BE SET AND CLEARED
(5)                                     ;*
(5)                                     ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
(5)                                     ;*F/FS OR GATES
(5)                                     ;*
(5)
(4)                                     ;;**********************************************************************
(3)  003616  000004                     TST7:   SCOPE
(2)  003620  012737  000100  001160             MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  003626  005077  175566             CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  003632  052777  000100  175560     BIS     #BIT6,@ASR      ;/SET BIT 6.
(1)  003640  012737  000100  001124     MOV     #BIT6,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
(1)  003646  017737  175546  001126     MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  003654  023737  001124  001126     CMP     $GDDAT,$BDDAT   ;/DID BIT 6 AND ONLY BIT 6 SET?
(1)  003662  001402                     BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003664  104002                             ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                                     ;/BIT 6 FAILED TO BIT SET.
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003666  000412                             BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  003670  042777  000100  175522  1$:      BIC     #BIT6,@ASR      ;/TRY CLEARING BIT 6.
(1)  003676  005037  001124                     CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  003702  017737  175512  001126             MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  003710  001401                             BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003712  104002                             ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                                     ;/BIT 6 FAILED TO CLEAR.
(1)
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003714                             2$:
(1)
6002
```

```
(1)                                         ;/#
(5)                                     ;;********************* ************************************************
(4)                                     ;*TEST 10        *TEST THAT CLOCK A STATUS REGISTER BIT 5 CAN BE SET AND CLEARED
(5)                                     ;*
(5)                                     ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
(5)                                     ;*F/FS OR GATES
(5)                                     ;*
(5)
(4)                                     ;;*****************************************************************
(3)  003714  000004                     TST10:  SCOPE
(2)  003716  012737  000100  001160             MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  003724  005077  175470             CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  003730  052777  000040  175462     BIS     #BIT5,@ASR      ;/SET BIT 5.
(1)  003736  012737  000040  001124     MOV     #BIT5,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
(1)  003744  017737  175450  001126     MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  003752  023737  001124  001126     CMP     $GDDAT,$BDDAT   ;/DID BIT 5 AND ONLY BIT 5 SET?
(1)  003760  001402                     BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003762  104002                             ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                                     ;/BIT 5 FAILED TO BIT SET.
(2)

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  003764  000412                             BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  003766  042777  000040  175424  1$:     BIC     #BIT5,@ASR      ;/TRY CLEARING BIT 5.
(1)  003774  005037  001124             CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  004000  017737  175414  001126     MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  004006  001401                     BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004010  104002                             ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                                     ;/BIT 5 FAILED TO CLEAR.
(1)
(2)

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004012                             2$:
(1)
6003
```

```
(1)                                             ;/#
(5)                                     ;;********************* ******************************************
(4)                                     ;*TEST 11        *TEST THAT CLOCK A STATUS REGISTER BIT 4 CAN BE SET AND CLEARED
(5)                                     ;*
(5)                                     ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
(5)                                     ;*F/FS OR GATES
(5)                                     ;*
(5)
(4)                                     ;;**************************************************************************
(3)  004012  000004                     TST11:  SCOPE
(2)  004014  012737  000100  001160             MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  004022  005077  175372             CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  004026  052777  000020  175364     BIS     #BIT4,@ASR      ;/SET BIT 4.
(1)  004034  012737  000020  001124     MOV     #BIT4,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
(1)  004042  017737  175352  001126     MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  004050  023737  001124  001126     CMP     $GDDAT,$BDDAT   ;/DID BIT 4 AND ONLY BIT 4 SET?
(1)  004056  001402                     BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004060  104002                             ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                                     ;/BIT 4 FAILED TO BIT SET.
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004062  000412                             BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  004064  042777  000020  175326  1$:     BIC     #BIT4,@ASR      ;/TRY CLEARING BIT 4.
(1)  004072  005037  001124             CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  004076  017737  175316  001126     MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  004104  001401                     BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004106  104002                             ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                                     ;/BIT 4 FAILED TO CLEAR.
(1)
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004110                             2$:
(1)
6004
```

```
                                        ;/#
                                ;;********************* ***************************************************
                                ;*TEST 12        *TEST THAT CLOCK A STATUS REGISTER BIT 3 CAN BE SET AND CLEARED
                                ;*
                                ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
                                ;*F/FS OR GATES
                                ;*
                                ;;*****************************************************************
(3)  004110  000004             TST12:  SCOPE
(2)  004112  012737  000100  001160     MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  004120  005077  175274             CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  004124  052777  000010  175266     BIS     #BIT3,@ASR      ;/SET BIT 3.
(1)  004132  012737  000010  001124     MOV     #BIT3,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
(1)  004140  017737  175254  001126     MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  004146  023737  001124  001126     CMP     $GDDAT,$BDDAT   ;/DID BIT 3 AND ONLY BIT 3 SET?
(1)  004154  001402                     BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004156  104002                     ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                             ;/BIT 3 FAILED TO BIT SET.
(2)
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004160  000412                     BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  004162  042777  000010  175230  1$:    BIC     #BIT3,@ASR      ;/TRY CLEARING BIT 3.
(1)  004170  005037  001124             CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  004174  017737  175220  001126     MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  004202  001401                     BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004204  104002                     ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                             ;/BIT 3 FAILED TO CLEAR.
(1)
(2)
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004206                     2$:
(1)
6005
```

```
(1)                                         ;/#
(5)                                 ;;********************* ***********************************************
(4)                                 ;*TEST 13        *TEST THAT CLOCK A STATUS REGISTER BIT 2 CAN BE SET AND CLEARED
(5)                                 ;*
(5)                                 ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
(5)                                 ;*F/FS OR GATES
(5)                                 ;*
(5)
(4)                                 ;;*****************************************************************
(3)  004206  000004                 TST13:  SCOPE
(2)  004210  012737  000100  001160         MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  004216  005077  175176                 CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  004222  052777  000004  175170         BIS     #BIT2,@ASR      ;/SET BIT 2.
(1)  004230  012737  000004  001124         MOV     #BIT2,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
(1)  004236  017737  175156  001126         MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  004244  023737  001124  001126         CMP     $GDDAT,$BDDAT   ;/DID BIT 2 AND ONLY BIT 2 SET?
(1)  004252  001402                         BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004254  104002                         ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                                 ;/BIT 2 FAILED TO BIT SET.
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004256  000412                         BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  004260  042777  000004  175132  1$:    BIC     #BIT2,@ASR      ;/TRY CLEARING BIT 2.
(1)  004266  005037  001124                 CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  004272  017737  175122  001126         MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  004300  001401                         BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004302  104002                         ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                                 ;/BIT 2 FAILED TO CLEAR.
(1)
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004304                         2$:
(1)
6006
```

```
(1)                                         ;/#
(5)                                 ;;********************* ***********************************************
(4)                                 ;*TEST 14        *TEST THAT CLOCK A STATUS REGISTER BIT 1 CAN BE SET AND CLEARED
(5)                                 ;*
(5)                                 ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
(5)                                 ;*F/FS OR GATES
(5)                                 ;*
(5)
(4)                                 ;;*****************************************************************
(3)  004304  000004                 TST14:  SCOPE
(2)  004306  012737  000100  001160         MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  004314  005077  175100                 CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  004320  052777  000002  175072         BIS     #BIT1,@ASR      ;/SET BIT 1.
(1)  004326  012737  000002  001124         MOV     #BIT1,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
(1)  004334  017737  175060  001126         MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  004342  023737  001124  001126         CMP     $GDDAT,$BDDAT   ;/DID BIT 1 AND ONLY BIT 1 SET?
(1)  004350  001402                         BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004352  104002                         ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                                 ;/BIT 1 FAILED TO BIT SET.
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004354  000412                         BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  004356  042777  000002  175034  1$:    BIC     #BIT1,@ASR      ;/TRY CLEARING BIT 1.
(1)  004364  005037  001124                 CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  004370  017737  175024  001126         MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  004376  001401                         BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004400  104002                         ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                                 ;/BIT 1 FAILED TO CLEAR.
(1)
(2)
                                    ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004402                         2$:
(1)
6007
```

```
(1)                                         ;/#
(5)                                     ;;********************* ************************************************
(4)                                     ;*TEST 15        *TEST THAT CLOCK A STATUS REGISTER BIT 0 CAN BE SET AND CLEARED
(5)                                     ;*
(5)                                     ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
(5)                                     ;*F/FS OR GATES
(5)                                     ;*
(5)
(4)                                     ;;*****************************************************************
(3)  004402  000004                     TST15:  SCOPE
(2)  004404  012737  000100  001160             MOV     #100,$TIMES     ;;DO 100 ITERATIONS
(1)
(1)  004412  005077  175002             1$:     CLR     @ASR            ;/CLEAR THE STATUS REGISTER.
(1)  004416  052777  000001  174774             BIS     #BIT0,@ASR      ;/SET BIT 0.
(1)  004424  012737  000001  001124             MOV     #BIT0,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
(1)  004432  017737  174762  001126             MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
(1)  004440  023737  001124  001126             CMP     $GDDAT,$BDDAT   ;/DID BIT 0 AND ONLY BIT 0 SET?
(1)  004446  001402                             BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004450  104002                             ERROR   2               ;/ERROR CLOCK AS STATUS REGISTER
(1)                                                                     ;/BIT 0 FAILED TO BIT SET.
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004452  000412                             BR      2$              ;/BR TO END SUBTEST.
(1)
(1)  004454  042777  000001  174736  1$:     BIC     #BIT0,@ASR      ;/TRY CLEARING BIT 0.
(1)  004462  005037  001124                     CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
(1)  004466  017737  174726  001126             MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
(1)  004474  001401                             BEQ     2$              ;/IF ZERO - NO ERROR!
(1)
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004476  104002                             ERROR   2               ;/ERROR - CLOCK A STATUS REGISTER.
(1)                                                                     ;/BIT 0 FAILED TO CLEAR.
(1)
(2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

(1)  004500                             2$:
(1)
6008         000010                             .RADIX  8
```

```
 6039
 6040
  (5)                                ;;*****************************************************************
  (4)                                ;*TEST 16       *TEST THAT PATERN 125252 WILL SET AND CLEAR IN BUFFER REG.
  (4)                                ;;*****************************************************************
  (3)  004500  000004                TST16:  SCOPE
  (1)
  (1)  004502  005077  174714                CLR     @ABR              ;/CLEAR THE BUFFER REG.
  (1)  004506  012737  125252  001124        MOV     #125252,$GDDAT    ;/RECORD PATTERN: 125252 .
  (1)  004514  013777  001124  174700        MOV     $GDDAT,@ABR       ;/SET PATTERN IN BUFFER REG.
  (1)  004522  017737  174674  001126        MOV     @ABR,$BDDAT       ;/READ THE BUFFER REG.
  (1)
  (1)  004530  023737  001124  001126        CMP     $GDDAT,$BDDAT     ;/DID THE PATTERN SET OK?
  (1)  004536  001402                        BEQ     1$                ;/YES-TRY CLEARING IT.
  (1)
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004540  104003                        ERROR   3                 ;/ERROR PATTERN 125252 FAILED TO
  (1)                                                                  ;/SET PROPERLY IN BUFFER REG.
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004542  000412                        BR      2$                ;/GOTO SCOPE LOOP.
  (1)
  (1)  004544  042777  125252  174650  1$:   BIC     #125252,@ABR      ;/TRY CLEARING PATTERN.
  (1)  004552  005037  001124                CLR     $GDDAT            ;/EXPECT ZERO BACK.
  (1)  004556  017737  174640  001126        MOV     @ABR,$BDDAT       ;/READ BUFFER REG.,WAS IT ZERO?
  (1)  004564  001401                        BEQ     2$                ;/YES-NEXT TEST.
  (1)
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004566  104003                        ERROR   3                 ;/BUFFER REG. COULD NOT BE LOADED
  (1)                                                                  ;/TO A ZERO.
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004570                        2$:
  (1)
```

```
 6042
  (5)                                ;;********************* ***************************************************
  (4)                                ;*TEST 17        *TEST THAT PATERN 052525 WILL SET AND CLEAR IN BUFFER REG.
  (4)                                ;;***************************************************************************
  (3)  004570  000004                TST17:  SCOPE
  (1)
  (1)  004572  005077  174624                CLR     @ABR              ;/CLEAR THE BUFFER REG.
  (1)  004576  012737  052525  001124        MOV     #052525,$GDDAT    ;/RECORD PATTERN: 052525 .
  (1)  004604  013777  001124  174610        MOV     $GDDAT,@ABR       ;/SET PATTERN IN BUFFER REG.
  (1)  004612  017737  174604  001126        MOV     @ABR,$BDDAT       ;/READ THE BUFFER REG.
  (1)
  (1)  004620  023737  001124  001126        CMP     $GDDAT,$BDDAT     ;/DID THE PATTERN SET OK?
  (1)  004626  001402                        BEQ     1$                ;/YES-TRY CLEARING IT.
  (1)
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004630  104003                        ERROR   3                 ;/ERROR PATTERN 052525 FAILED TO
  (1)                                                                  ;/SET PROPERLY IN BUFFER REG.
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004632  000412                        BR      2$                ;/GOTO SCOPE LOOP.
  (1)
  (1)  004634  042777  052525  174560  1$:   BIC     #052525,@ABR      ;/TRY CLEARING PATTERN.
  (1)  004642  005037  001124                CLR     $GDDAT            ;/EXPECT ZERO BACK.
  (1)  004646  017737  174550  001126        MOV     @ABR,$BDDAT       ;/READ BUFFER REG.,WAS IT ZERO?
  (1)  004654  001401                        BEQ     2$                ;/YES-NEXT TEST.
  (1)
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004656  104003                        ERROR   3                 ;/BUFFER REG. COULD NOT BE LOADED
  (1)                                                                  ;/TO A ZERO.
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  004660                        2$:
  (1)
```

```
                                        .SBTTL  *
                                        .SBTTL  * PHASE 2 ADVANCED BASIC LOGIC TESTS
                                        .SBTTL  *


                                ;;**********************************************************************
                                ;*TEST 20        *TEST THE LOW BYTE OPERATION OF CLOCK'S STATUS REGISTER

                                ;*
                                ;*WE CAN SUCCESSFULLY WRITE EVERY BIT IN STATUS REG A
                                ;*NOW LETS CHECK THE BYTE OPERATION OF THIS REGISTER.
                                ;*

                                ;;**********************************************************************
004660  000004                  TST20:  SCOPE
004662  012737  000050  001160          MOV     #50,$TIMES      ;;DO 50 ITERATIONS

004670  005077  174524          CLR     @ASR            ;MAKE SURE THE STATUS REGISTER IS CLEAR.
004674  112777  127677  174516  MOVB    #127677,@ASR    ;TRY WRITING ALL BITS IN THE
                                                        ;STATUS REGISTER. LOGIC SHOULD PREVENT IT
                                                        ;FROM BEING WRITTEN INTO BECAUSE
                                                        ;WE ARE USING A DATOB INSTRUCTION.

004702  017777  174512  174216  MOV     @ASR,@$BDDAT    ;NOW EXAMINE THE
                                                        ;STATUS REGISTER.
004710  013737  001126  001124  MOV     $BDDAT,$GDDAT   ;FIX $GDDAT FOR ERROR TYPEOUT IF
004716  105037  001125          CLRB    $GDDAT+1        ;ANY RROR HAS OCCURRED, UPPER BYTE CLEARED.

004722  105737  001127          TSTB    $BDDAT+1        ;ARE ANY BITS IN THE UPPER BYTE
                                                        ;OF THE STATUS REGISTER SET?
004726  001401                  BEQ     1$              ;BRANCH NEXT TEST IF UPPER BYTE=0.


                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

004730  104001                  ERROR   1               ;ERROR - WROTE INTO UPPER BYTE OF
                                                        ;CLOCK'S STATUS WHEN
                                                        ;DOING A DATOB TO THE LOW BYTE.


                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

004732                          1$:

```

```
 6083                                   ;;*****************************************************************
  (3)                                   ;*TEST 21       *TEST THE HIGH BYTE OPERATION OF A'S STATUS REGISTER
  (4)                                   ;
  (4)                                   ;*
  (4)                                   ;*WE CAN SUCCESSFULLY WRITE EVERY BIT IN STATUS REG A
  (4)                                   ;*NOW LETS CHECK THE BYTE OPERATION OF THIS REGISTER.
  (4)                                   ;*
  (4)
  (3)                                   ;;*****************************************************************
  (2) 004732  000004                    TST21:  SCOPE
  (1) 004734  012737  000050  001160            MOV     #50,$TIMES      ;;DO 50 ITERATIONS
 6084
 6085 004742  005077  174452            CLR     @ASR            ;CLEAR THE STATUS REGISTER.
 6086
 6087 004746  005237  001420            INC     ASR             ;ADD #1 TO THE STATUS REGISTER'S ADDRESS
 6088                                                           ;SO THAT WE WILL BE WRITING INTO
 6089                                                           ;THE HIGH BYTE.
 6090
 6091 004752  112777  177213  174440    MOVB    #177213,@ASR    ;TRY WRITING ALL BITS IN THE STATUS
 6092                                                           ;REGISTER. LOGIC SHOULD PREVENT THE LOW
 6093                                                           ;BYTE OF THE STATUS REGISTER FROM
 6094                                                           ;BEING WRITTEN INTO BECAUSE WE ARE USING
 6095                                                           ;A DATOB INSTRUCTION WITH A00 SET.
 6096
 6097 004760  005337  001420            DEC     ASR             ;FIX ADDRESS OF THE STATUS REGISTER ADDR.
 6098                                                           ;SO WE CAN LOOK AT THE WHOLE WORD.
 6099 004764  017737  174430  001126    MOV     @ASR,$BDDAT     ;READ BACK WHAT THE STATUS REG. CONTAINS
 6100 004772  013737  001126  001124    MOV     $BDDAT,$GDDAT   ;FIX $GDDAT FOR ERROR TYPEOUT IF AN ERROR
 6101 005000  105037  001124            CLRB    $GDDAT          ;OCCURRED, LOWER BYTE CLEARED.
 6102 005004  105737  001126            TSTB    $BDDAT          ;IS LOWER BYTE CLEAR?
 6103 005010  001401                    BEQ     1$              ;BR IF YES TO NEXT SUBTEST.
 6104
 6105
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6106 005012  104001                            ERROR   1               ;ERROR - WROTE INTO LOWER BYTE
 6107                                                                   ;OF CLOCKS STATUS REGISTER WHEN
 6108                                                                   ;DOING A DATOB TO THE HIGH BYTE.
 6109
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6110 005014                           1$:
```

```
 6112                                   ;;*****************************************************************
  (3)                                   ;*TEST 22       *TEST CLOCK'S COUNT REGISTER WITH 125252 PATTERN
  (3)                                   ;;*****************************************************************
  (2) 005014  000004                    TST22:  SCOPE
  (1) 005016  012737  000040  001160            MOV     #40,$TIMES      ;;DO 40 ITERATIONS
 6113
 6114 005024  005077  174370                    CLR     @ASR            ;SELECT MODE 0.
 6115 005030  012777  125252  174364            MOV     #125252,@ABR    ;LOAD THE BUFFER REGISTER WITH
 6116                                                                    ;PATTERN 125252. IT WILL BE
 6117                                                                    ;TRANSFERRED TO THE COUNT REGISTER
 6118                                                                    ;SINCE THIS IS MODE 0.
 6119 005036  052777  000001  174354            BIS     #BIT0,@ASR      ;SET GO BIT(ALLOWS BUFFER-COUNT REG XFER).
 6120
 6121 005044  012737  125252  001124            MOV     #125252,$GDDAT  ;SET EXPECTED TO PATTERN IN CASE OF
 6122                                                                    ;NEED OF ERROR TYPEOUT.
 6123                                                                    ;/-RDCLK-
  (1) 005052  017746  174342                    MOV     @ASR,-(SP)      ;/SAVE CSR
  (1) 005056  052777  004007  174334            BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (1)                                                                   ;/THIS MUST BE DONE IN
  (1)                                                                   ;/ORDER TO XFERR COUNTER
  (1)                                                                   ;/TO BUFFER ON ST2.
  (1) 005064  052777  001000  174326            BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (1) 005072  012746  000010                    MOV     #8.,-(SP)       ;/NOW GENERATE
  (1) 005076  052777  000400  174314  64$:      BIS     #BIT8,@ASR
  (1) 005104  005316                            DEC     (SP)            ;/EIGHT ST1 PULSES
  (1) 005106  001373                            BNE     64$
  (1) 005110  005726                            TST     (SP)+           ;/RESET STACK
  (1) 005112  017737  174304  001126            MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (1)                                                                   ;/PREVIOUS COUNTER
  (1) 005120  012677  174274                    MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (1) 005124  005737  001126                    TST     $BDDAT          ;/RESTORE CSR
 6124
 6125 005130  023737  001124  001126            CMP     $GDDAT,$BDDAT   ;DID ALL THE BITS AND NO OTHER BITS
 6126                                                                    ;COME THROUGH?
 6127 005136  001401                            BEQ     1$              ;BR IF YES TO NEXT TEST.
 6128
 6129

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6130 005140  104005                            ERROR   5               ;DATA ERROR CLOCK - PATTERN ''125252''
 6131                                                                   ;FAILED TO TRANSFER PROPERLY BETWEEN
 6132                                                                   ;BUFFER AND COUNT REGISTERS.
 6133
 6134

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6135 005142                            1$:
```

```
 6137                                   ;;*****************************************************************
  (3)                                   ;*TEST 23       *TEST CLOCKS COUNTER REGISTER WITH 052525 PATTERN
  (3)                                   ;;*****************************************************************
  (2)  005142  000004                   TST23:  SCOPE
  (1)  005144  012737  000040  001160           MOV     #40,$TIMES      ;;DO 40 ITERATIONS
 6138
 6139  005152  005077  174242                   CLR     @ASR            ;SELECT MODE 0.
 6140  005156  012777  052525  174236           MOV     #052525,@ABR    ;LOAD THE BUFFER REGISTER WITH
 6141                                                                   ;PATTERN 052525. IT WILL BE
 6142                                                                   ;TRANSFERRED TO THE COUNT REGISTER
 6143                                                                   ;SINCE THIS IS MODE 0.
 6144  005164  052777  000001  174226           BIS     #BIT0,@ASR      ;SET BO BIT(ALLOWS BUFFER-COUNT REG XFER).
 6145
 6146  005172  012737  052525  001124           MOV     #052525,$GDDAT  ;SET EXPECTED TO PATTERN IN CASE OF
 6147                                                                   ;NEED OF ERROR TYPEOUT.
 6148                                                                   ;/-RDCLK-
  (1)  005200  017746  174214                   MOV     @ASR,-(SP)      ;/SAVE CSR
  (1)  005204  052777  004007  174206           BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (1)                                                                   ;/THIS MUST BE DONE IN
  (1)                                                                   ;/ORDER TO XFERR COUNTER
  (1)                                                                   ;/TO BUFFER ON ST2.
  (1)  005212  052777  001000  174200           BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (1)  005220  012746  000010                   MOV     #8.,-(SP)       ;/NOW GENERATE
  (1)  005224  052777  000400  174166  64$:     BIS     #BIT8,@ASR
  (1)  005232  005316                           DEC     (SP)            ;/EIGHT ST1 PULSES
  (1)  005234  001373                           BNE     64$
  (1)  005236  005726                           TST     (SP)+           ;/RESET STACK
  (1)  005240  017737  174156  001126           MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (1)                                                                   ;/PREVIOUS COUNTER
  (1)  005246  012677  174146                   MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (1)  005252  005737  001126                   TST     $BDDAT          ;/RESTORE CSR
 6149
 6150  005256  023737  001124  001126           CMP     $GDDAT,$BDDAT   ;DID ALL THE BITS AND NO OTHER BITS
 6151                                                                   ;COME THROUGH?
 6152  005264  001401                           BEQ     1$              ;BR IF YES TO NEXT TEST.
 6153
 6154

                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6155  005266  104005                           ERROR   5               ;DATA ERROR CLOCK - PATTERN '052525'
 6156                                                                   ;FAILED TO TRANSFER PROPERLY BETWEEN
 6157                                                                   ;BUFFER AND COUNT REGISTERS.
 6158
 6159

                                        .;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6160  005270                           1$:
 6161
```

```
 6169                                   ;;**********************************************************************
  (3)                                   ;*TEST 24       *TEST THAT INIT CLEARS STATUS REGISTER
  (4)                                   ;*
  (4)                                   ;*TESTING OF THE INIT LOGIC AS RECEIVED FROM THE QBUS AND BUFFERED
  (4)                                   ;*TO STATUS REGISTER F/FS.
  (4)                                   ;*
  (3)                                   ;**********************************************************************
  (2) 005270  000004                    TST24:  SCOPE
  (1) 005272  012737  000005  001160            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
 6170
 6171 005300  005037  001124            CLR     $GDDAT          ;EXPECTED DATA IS ZERO.
 6172 005304  012777  176377  174106    MOV     #176377,@ASR    ;SET ALL BITS IN THE STATUS REG.
 6173
 6174 005312  000005                    RESET                   ;SYSTEM INITIALIZE.
 6175 005314  052777  000100  173622    BIS     #BIT6,@$TKS     ;ENABLE TKB INTR.
 6176
 6177 005322  017737  174072  001126    MOV     @ASR,$BDDAT     ;READ THE STATUS REG., ALL BITS SHOULD
 6178                                                            ;HAVE BEEN CLEARED BY INIT.
 6179 005330  001402                    BEQ     1$              ;BR IF YES TO NEXT TEST.
 6180
 6181                                   ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6182
 6183 005332  104002                            ERROR   2               ;ERROR - SYSTEM INIT FAILED TO CLEAR
 6184                                                                   ;STATUS REGISTER CLOCK A.
 6185
 6186                                   ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6187 005334  000414                            BR      TST25           ;;
 6188 005336                            1$:
 6189 005336  005737  001476                    TST     DWARF           ;TEST DWARF/TESTOR INDICATOR
 6190 005342  001411                            BEQ     TST25           ;;
 6191 005344  100010                            BPL     TST25           ;;
 6192 005346  052777  016002  174064            BIS     #BIT11!BIT12!BIT10!BIT1,@DR2    ;ENABLE ST1,ST2 TO LATCH.
 6193 005354  032777  000006  174054            BIT     #6,@DR          ;ST1,ST2, OVERFLOW SET?
 6194 005362  001401                            BEQ     TST25           ;;
 6195
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6196 005364  104006                            ERROR   6               ;INIT FAILED TO CLEAR
 6197                                                                   ;ST1,ST2, AND/OR OVERFLOW
 6198
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
 6207                                   ;;*****************************************************************
  (3)                                   ;*TEST 25       *TEST THAT INIT CLEARS BUFFER REGISTER
  (4)                                   ;*
  (4)                                   ;*THIS TEST IS DESIGNED TO SEE IF ''INIT H''
  (4)                                   ;*CLEARS THE BUFFER REGISTER. WE ALREADY
  (4)                                   ;*KNOW IT CLEARS THE STATUS REG.
  (4)                                   ;*
  (3)                                   ;*****************************************************************
  (2) 005366  000004                    TST25:  SCOPE
  (1) 005370  012737  000005  001160            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
 6208
 6209 005376  005037  001124            CLR     $GDDAT          ;CLEAR EXPECTED DATA.
 6210 005402  012777  177777  174012    MOV     #177777,@ABR    ;SET ALL BITS IN THE BUFFER REGISTER.
 6211
 6212 005410  000005                    RESET                   ;ISSUE SYSTEM INITIALIZE.
 6213 005412  052777  000100  173524    BIS     #BIT6,@$TKS     ;ENABLE TKB INTR.
 6214
 6215 005420  017737  173776  001126    MOV     @ABR,$BDDAT     ;READ THE BUFFER REGISTER, ALL BITS
 6216                                                            ;SHOULD HAVE BEEN CLEARED BY INIT.
 6217 005426  001401                    BEQ     1$              ;BR IF YES TO NEXT SUBTEST.
 6218
 6219
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6220 005430  104003                    ERROR   3               ;ERROR - SYSTEM INIT FAILED
 6221                                                           ;TO CLEAR BUFFER REGISTER A.
 6222
 6223
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6224 005432                            1$:
 6225 005432  005737  001476                    TST     DWARF           ;CHECK DWARF/TESTOR INDICATOR
 6226 005436  001404                            BEQ     TST26           ;;BR IF NO DWARF/TESTER
 6227 005440  100003                            BPL     TST26           ;;BR IF ONLY DWARF
 6228 005442  052777  016000  173770            BIS     #BIT11!BIT12!BIT10,@DR2 ;ENABLE THEM
```

```
                                ;;*****************************************************************
                                ;*TEST 26       *TEST THE SETTING OF MAINTENANCE ST2 IN CLOCK BIT 15 TO SET
                                ;;*****************************************************************
005450  000004                  TST26:  SCOPE

005452  005077  173742                  CLR     @ASR            ;2 CLEAR THE CSR
005456  012777  000001  173734          MOV     #1,@ASR         ;SET THE GO BIT(ENABLES ST2 TO SET FLG).
005464  052777  001000  173726          BIS     #BIT9,@ASR      ;SET MAINTENANCE ST2.

005472  005777  173722                  TST     @ASR            ;DID BIT15 (ST2 FLAG) SET?
005476  100402                          BMI     1$              ;BR IF YES - NEXT TEST


                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

005500  104001                          ERROR   1               ;ERROR - MAINTENANCE ST2 (BIT9)
                                                                ;DID NOT SET BIT15 (ST2 FLAG).


                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

005502  000422                          BR      TST27           ;;
005504  005737  001476          1$:     TST     DWARF           ;CHECK DWRAF/TESTOR INDICATOR
005510  001407                          BEQ     2$              ;;BR IF NONE
005512  100016                          BPL     TST27           ;;BR IF DWARF
005514  032777  000004  173714          BIT     #BIT2,@DR       ;DID EXTERNAL ST2 GET SET?
005522  001002                          BNE     2$

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

005524  104006                          ERROR   6               ;ST2 OUT NOT DETECTED
                                                                ;BY TESTOR

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

005526  000410                          BR      TST27           ;;

005530  042777  100000  173662  2$:     BIC     #BIT15,@ASR     ;NOW TRY CLEARING ST2 FLAG.
005536  032777  100000  173654          BIT     #BIT15,@ASR     ;DID IT CLEAR??
005544  001401                          BEQ     TST27           ;;

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

005546  104001                          ERROR   1               ;ST2 FLAG FAILED TO CLEAR.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
                                ;;*****************************************************************
                                ;*TEST 27       *TEST THAT ST1 FLAG SETS ON MAINTENANCE ST1
                                ;;*****************************************************************
 (2) 005550 000004              TST27:  SCOPE
6264 005552 005077 173642               CLR     @ASR            ;2 CLEAR THE CSR
6265 005556 052777 000400 173634        BIS     #BIT8,@ASR      ;2 SET MAIN. ST1
6266 005564 032777 002000 173626        BIT     #BIT10,@ASR     ;2 DID ST1 FLAG SET?
6267 005572 001001                      BNE     1$              ;2 YES - CONTINUE
6268
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6269 005574 104001                      ERROR   1               ;2 ST1 FLAG FAILED TO SET
6270
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6271 005576 042777 002000 173614 1$:    BIC     #BIT10,@ASR     ;2 NOW TRY CLEARING IT.
6272 005604 032777 002000 173606        BIT     #BIT10,@ASR     ;2 DID IT CLEAR?
6273 005612 001401                      BEQ     TST30           ;;
6274
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6275 005614 104001                      ERROR   1               ;2 ST1 FLAG FAILED TO CLEAR
6276
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


6277
6278                            ;;*****************************************************************
                                ;*TEST 30       *TEST THAT BIT00 IN CLOCK STATUS REG. WILL SET WHEN BIT13 AND MAIN. ST2
                                ;;*****************************************************************
 (2) 005616 000004              TST30:  SCOPE
6279 005620 012777 020000 173572        MOV     #BIT13,@ASR     ;SET 'ST2 ENB COUNTER' IN CLK  STATUS REG.
6280 005626 052777 001000 173564        BIS     #BIT9,@ASR      ;GENERATE A MAINTENANCE ST2.
6281 005634 032777 000001 173556        BIT     #BIT00,@ASR     ;DID BIT00 (GO) SET?
6282 005642 001001                      BNE     1$              ;BR IF YES - NEXT TEST.
6283
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6284 005644 104001                      ERROR   1               ;ERROR - BIT00 OF CLOCK'S STATUS REGISTER
6285                                                            ;FAILED TO SET WHEN BIT13 WAS SET
6286                                                            ;AND A MAINTENANCE ST2 GENERATED.
6287
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6288 005646 032777 020000 173544 1$:    BIT     #BIT13,@ASR     ;DID BIT 13 CLEAR ITSELF??
6289 005654 001401                      BEQ     2$              ;YES GO AHEAD.
6290
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6291 005656 104001                      ERROR   1               ;BIT 13 FAILED TO CLEAR.
6292
                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6293 005660 005077 173534       2$:     CLR     @ASR            ;LEAVE SUBTEST WITH CLOCK CLEAR.
```

```
 6295                                           .SBTTL  *
 6296                                           .SBTTL  *PHASE 3 COUNT TESTS
 6297                                           .SBTTL  *
 6298
 6299                                   ;;*****************************************************************
  (3)                                   ;*TEST 31        *TEST TO SEE IF THE COUNTER WILL INCREMENT
  (3)                                   ;;*****************************************************************
  (2)  005664  000004                   TST31:  SCOPE
  (1)  005666  012737  000010  001160           MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 6300
 6301  005674  005077  173520                   CLR     @ASR            ;CLEAR THE CSR.
 6302  005700  005077  173516                   CLR     @ABR            ;CLEAR THE BUFFER
 6303  005704  052777  000061  173506           BIS     #BIT5!BIT4!BIT0,@ASR ;SET RATE:ST1, GO.
 6304  005712  052777  000400  173500           BIS     #BIT8,@ASR      ;GENERATE A MAINTENANCE ST1.
 6305                                                                   ;DID THE CLOCK COUNT?
 6306                                                                   ;/-RDCLK-
  (1)  005720  017746  173474                   MOV     @ASR,-(SP)      ;/SAVE CSR
  (1)  005724  052777  004007  173466           BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (1)                                                                   ;/THIS MUST BE DONE IN
  (1)                                                                   ;/ORDER TO XFERR COUNTER
  (1)                                                                   ;/TO BUFFER ON ST2.
  (1)  005732  052777  001000  173460           BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (1)  005740  012746  000010                   MOV     #8.,-(SP)       ;/NOW GENERATE
  (1)  005744  052777  000400  173446  64$:     BIS     #BIT8,@ASR
  (1)  005752  005316                           DEC     (SP)            ;/EIGHT ST1 PULSES
  (1)  005754  001373                           BNE     64$
  (1)  005756  005726                           TST     (SP)+           ;/RESET STACK
  (1)  005760  017737  173436  001126           MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (1)                                                                   ;/PREVIOUS COUNTER
  (1)  005766  012677  173426                   MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (1)  005772  005737  001126                   TST     $BDDAT          ;/RESTORE CSR
 6307  005776  001001                           BNE     1$              ;YES, NEXT TEST.
 6308
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6309  006000  104006                           ERROR   6               ;CLOCK FAILED TO INCREMENT.
 6310
 6311
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6312  006002                           1$:
 6313
```

```
 6323                                   ;;*****************************************************************
  (3)                                   ;*TEST 32       *SEE IF CLOCK WILL COUNT UP FROM A ZERO BASE, RATE:ST1
  (4)                                   ;*
  (4)                                   ;*      NOTE:   IN THIS TEST, LOOP ON ERROR WILL CAUSE A LOOP
  (4)                                   ;*              ON THE FAILING COUNT PATTERN;
  (4)                                   ;*              WHILE LOOP ON TEST WILL START THE TEST
  (4)                                   ;*              AND THE CLOCK FROM ZERO TO THE FAILING COUNT.
  (4)                                   ;*
  (3)                                   ;*****************************************************************
  (2)  006002  000004                   TST32:  SCOPE
  (1)  006004  012737  000010  001160           MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 6324
 6325  006012  005077  173402                   CLR     @ASR            ;CLEAR THE CSR.
 6326  006016  005077  173400                   CLR     @ABR            ;CLEAR THE BUFFER REG
 6327  006022  012737  000000  001124           MOV     #0,$GDDAT       ;CLEAR EXPECTED.
 6328  006030  012737  006170  001110           MOV     #2$,$LPERR
 6329
 6330  006036  012777  000061  173354  1$:      MOV     #BIT5!BIT4!BIT0,@ASR ;START CLOCK, RATE:ST1.
 6331
 6332  006044  052777  000400  173346           BIS     #BIT8,@ASR      ;GENERATE A MAINTENANCE ST1
 6333                                           ;CLOCK SHOULD COUNT ONCE.
 6334  006052  005737  001476                   TST     DWARF           ;CHECK THE DWARF/TESTER INDICATOR
 6335  006056  001407                           BEQ     10$             ;;BR IF NONE
 6336  006060  100056                           BPL     TST33           ;;BR IF DWARF
 6337  006062  032777  000002  173346           BIT     #BIT1,@DR       ;YES - DID ST1 GET SET?
 6338  006070  001002                           BNE     10$             ;YES
 6339
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6340  006072  104006                           ERROR   6               ;ST1 OUT NOT DETECTED
 6341                                           ;BY TESTOR
 6342
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6343  006074  000450                           BR      TST33           ;;
 6344  006076                        10$:
 6345                                           ;/-RDCLK1-
  (1)  006076  017746  173316                   MOV     @ASR,-(SP)      ;/SAVE CSR CONTENTS.
  (1)  006102  052777  000005  173310           BIS     #5,@ASR         ;/SET TO MODE 2,GO
  (1)  006110  042777  100000  173302           BIC     #BIT15,@ASR     ;/CLR ST FLAG.
  (1)  006116  052777  001000  173274           BIS     #BIT9,@ASR      ;/GENERATE ST2 PULSE.
  (1)  006124  017737  173272  001126           MOV     @ABR,$BDDAT     ;/READ COUNT REG.
  (1)  006132  052677  173262                   BIS     (SP)+,@ASR      ;/RESTORE CSR.
  (1)  006136  005737  001126                   TST     $BDDAT          ;/PREVIOUS CONTENTS OF COUNT REG
  (1)                                           ;/IN $BDDAT.
 6346  006142  005237  001124                   INC     $GDDAT          ;COUNT=OLD COUNT+1
 6347  006146  013737  001124  001446           MOV     $GDDAT,$TMP0    ;FOR ERROR TYPEOUT.
 6348  006154  023737  001124  001126           CMP     $GDDAT,$BDDAT   ;COUNT READ=COUNT EXP'ED?
 6349  006162  001402                           BEQ     2$              ;YES - SEE IF WE'RE THROUGH.
 6350
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6351  006164  104011                           ERROR   11              ;CLOCK FAILED TO COUNT UP PROPERLY.
 6352
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
 6353  006166  000413                          BR      3$              ;GOTO SCOPE LOOP.
 6354
 6355  006170  005077  173224          2$:     CLR     @ASR
 6356  006174  005737  001202                  TST     $PASS           ;TEST IF FIRST PASS
 6357  006200  001406                          BEQ     TST33           ;;BR IF FIRST
 6358  006202  013777  001124  173212          MOV     $GDDAT,@ABR
 6359  006210  005737  001124                  TST     $GDDAT          ;ALL DONE?
 6360  006214  001310                          BNE     1$              ;NO DO NEXT INCREMENT.
 6361  006216                          3$:
 6362                                  ;;*****************************************************************
   (3)                                 ;*TEST 33       *TEST THAT OVERFLOW (CSR BIT07) WILL SET ON OVERFLOW
   (3)                                 ;;*****************************************************************
   (2)  006216  000004                  TST33:  SCOPE
 6363
 6364  006220  005737  001476                  TST     DWARF           ;TEST DWARF/TESTER INDICATOR
 6365  006224  001406                          BEQ     2$              ;;BR IF NONE
 6366  006226  100040                          BPL     TST34           ;;BR IF DWARF
 6367
 6368  006230  052777  020002  173202          BIS     #BIT13!BIT1,@DR2
 6369  006236  012700  000010                  MOV     #8.,R0          ;SET TIME OUT NUMBER.
 6370  006242                          2$:
 6371  006242  005077  173152                  CLR     @ASR            ;CLEAR THE CSR
 6372  006246  012777  177777  173146          MOV     #-1,@ABR        ;SET PRESET BUFFER TO ALL ONES.
 6373
 6374  006254  052777  000061  173136          BIS     #BIT5!BIT4!BIT0,@ASR ;START CLOCK, RATE ST1.
 6375
 6376  006262  052777  000400  173130          BIS     #BIT8,@ASR      ;COUNT CLOCK ONCE, OVERFLOW
 6377                                                                  ;SHOULD OCCUR.
 6378  006270  105777  173124                  TSTB    @ASR            ;DID OVERFLOW SET?
 6379  006274  100402                          BMI     1$              ;YES - THEN NEXT TEST
 6380
 6381
                                  ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6382  006276  104006                          ERROR   6               ;ERROR - OVERFLOW, CSR BIT0?
 6383                                                                  ;FAILED TO SET ON OVERFLOW
 6384  006300  000413                          BR      TST34           ;;
 6385  006302  005737  001476          1$:     TST     DWARF           ;CHECK DWARF/TESTER INDICATOR
 6386  006306  001410                          BEQ     TST34           ;;BR IF NONE
 6387  006310  100007                          BPL     TST34           ;;BR IF DWARF
 6388  006312  032777  000010  173116          BIT     #BIT3,@DR
 6389  006320  001003                          BNE     TST34           ;;
 6390  006322  005300                          DEC     R0              ;DID WE ALLOW ENOUGH TIME??
 6391  006324  001366                          BNE     1$              ;NO-THEN WAIT.
 6392
                                  ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6393  006326  104006                          ERROR   6               ;OVERFLOW OUT NOT DETECTED
 6394                                                                  ;BY TESTOR
 6395
                                  ;.$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
 6397                                   ;;*****************************************************************
  (3)                                   ;*TEST 34       *TEST THAT OVERFLOW WILL CLEAR THE GO BIT
  (3)                                   ;;*****************************************************************
  (2)  006330  000004                   TST34:  SCOPE
 6398
 6399  006332  005077  173062                   CLR     @ASR            ;CLEAR THE CSR.
 6400
 6401  006336  012777  177777  173056           MOV     #-1,@ABR        ;PRESET CLOCK TO -1.
 6402
 6403  006344  052777  000061  173046           BIS     #BIT5!BIT4!BIT0,@ASR ;START CLOCK, RATE:ST1
 6404
 6405  006352  052777  000400  173040           BIS     #BIT8,@ASR      ;COUNT ONCE, OVERFLOW
 6406                                                                    ;SHOULD OCCUR CLEARING
 6407                                                                    ;ENABLE (CSR BIT00)
 6408
 6409  006360  032777  000001  173032           BIT     #BIT0,@ASR      ;DID THE ENABLE CLEAR?
 6410  006366  001401                           BEQ     1$              ;YES - NEXT TEST.
 6411
 6412                                   ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6413  006370  104006                           ERROR   6               ;ERROR - OVERFLOW FAILED
 6414                                                                    ;TO CLEAR ENABLE (CSR BIT00)
 6415
 6416  006372                           1$:
 6417
 6418                                   ;;*****************************************************************
  (3)                                   ;*TEST 35       *TEST THAT GO BIT DOES NOT CLEAR ON OVERFLOW, IF MODE 1
  (3)                                   ;;*****************************************************************
  (2)  006372  000004                   TST35:  SCOPE
 6419
 6420  006374  005077  173020                   CLR     @ASR            ;CLEAR THE CSR.
 6421  006400  012777  177777  173014           MOV     #-1,@ABR        ;PRESET BUFFER=ONE COUNT FROM OVERFLOW.
 6422  006406  052777  000063  173004           BIS     #63,@ASR        ;MODE 1, RATE:ST1, GO.
 6423
 6424  006414  052777  000400  172776           BIS     #BIT8,@ASR      ;GENERATE MAINTENANCE ST1.
 6425
 6426  006422  032777  000001  172770           BIT     #BIT0,@ASR      ;DID ENABLE (GO BIT) CLEAR?
 6427  006430  001001                           BNE     1$              ;NO (GOOD) NEXT TEST.
 6428
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6429  006432  104006                           ERROR   6               ;GO BIT CLEARED ON OVERFLOW
 6430                                                                    ;WHEN MODE 1 WAS SELECTED
 6431
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6432
 6433  006436  005077  172760           1$:     CLR     @ASR            ;CLEAR THE CLOCK.
 6434
```

```
 6472
  (5)                                   ;;**********************  ********************************************
  (4)                                   ;*TEST 36        *TEST THE ABILITY OF CLOCK TO COUNT AT 1MHZ RATE
  (5)
  (5)                                   ;*
  (5)                                   ;*THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
  (5)                                   ;*TO COUNT AT 1MHZ RATE.
  (5)                                   ;*
  (4)                                   ;;*****************************************************************************
  (3)  006440  000004                   TST36:  SCOPE
  (2)  006442  012737  000005  001160           MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)  006450  005077  172744           1$:     CLR     @ASR            ;/CLEAR CLOCK
  (1)  006454  005077  172742                   CLR     @ABR            ;/CLEAR PRESET BUFFER
  (1)  006460  012777  000011  172732           MOV     #BIT0!10,@ASR   ;/START CLOCK, MODE0, RATE:1MHZ
  (1)  006466  005000                           CLR     R0              ;/NOW WE'LL DO A LITTLE DELAY. THIS DELAY
  (1)  006470  005200                   1$:     INC     R0              ;/WILL AMOUNT TO APPROXIMATELY
  (1)  006472  001376                           BNE     1$              ;/369 MS.
  (1)
  (2)                                                                   ;/-RDCLK-
  (2)  006474  017746  172720                   MOV     @ASR,-(SP)      ;/SAVE CSR
  (2)  006500  052777  004007  172712           BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (2)                                                                   ;/THIS MUST BE DONE IN
  (2)                                                                   ;/ORDER TO XFERR COUNTER
  (2)                                                                   ;/TO BUFFER ON ST2.
  (2)  006506  052777  001000  172704           BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (2)  006514  012746  000010                   MOV     #8.,-(SP)       ;/NOW GENERATE
  (2)  006520  052777  000400  172672   64$:    BIS     #BIT8,@ASR
  (2)  006526  005316                           DEC     (SP)            ;/EIGHT ST1 PULSES
  (2)  006530  001373                           BNE     64$
  (2)  006532  005726                           TST     (SP)+           ;/RESET STACK
  (2)  006534  017737  172662  001126           MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (2)                                                                   ;/PREVIOUS COUNTER
  (2)  006542  012677  172652                   MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (2)  006546  005737  001126                   TST     $BDDAT          ;/RESTORE CSR
  (1)  006552  001004                           BNE     2$              ;/YES - NEXT TEST.
  (1)  006554  105766  177776                   TSTB    -2(SP)          ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
  (1)                                                                   ;/NOTE: CSR HAD BEEN PUT ON STACK.
  (1)  006560  100401                           BMI     2$              ;/NEXT TEST IF OVERFLOW.
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  006562  104006                           ERROR   6               ;/CLOCK FAILED TO COUNT AT
  (1)                                                                   ;/RATE:1MHZ
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  006564  005077  172630   2$:    CLR     @ASR            ;/CLEAR THE CLOCK.
  (1)
  (1)
```

```
 6474
  (5)                                ;;********************* ************************************************
  (4)                                ;*TEST 37       *TEST THE ABILITY OF CLOCK TO COUNT AT 100KHZ RATE
  (5)
  (5)                                ;*
  (5)                                ;*THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
  (5)                                ;*TO COUNT AT 100KHZ RATE.
  (5)                                ;*
  (4)                                ;;*****************************************************************
  (3)  006570  000004                TST37:  SCOPE
  (2)  006572  012737  000005  001160        MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)  006600  005077  172614                CLR     @ASR            ;/CLEAR CLOCK
  (1)  006604  005077  172612                CLR     @ABR            ;/CLEAR PRESET BUFFER
  (1)  006610  012777  000021  172602        MOV     #BIT0!20,@ASR   ;/START CLOCK, MODE0, RATE:100KHZ
  (1)  006616  005000                        CLR     R0              ;/NOW WE'LL DO A LITTLE DELAY. THIS DELAY
  (1)  006620  005200                1$:     INC     R0              ;/WILL AMOUNT TO APPROXIMATELY
  (1)  006622  001376                        BNE     1$              ;/369 MS.
  (1)
  (2)                                                                ;/-RDCLK-
  (2)  006624  017746  172570                MOV     @ASR,-(SP)      ;/SAVE CSR
  (2)  006630  052777  004007  172562        BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (2)                                                                ;/THIS MUST BE DONE IN
  (2)                                                                ;/ORDER TO XFERR COUNTER
  (2)                                                                ;/TO BUFFER ON ST2.
  (2)  006636  052777  001000  172554        BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (2)  006644  012746  000010                MOV     #8.,-(SP)       ;/NOW GENERATE
  (2)  006650  052777  000400  172542  64$:   BIS     #BIT8,@ASR
  (2)  006656  005316                        DEC     (SP)            ;/EIGHT ST1 PULSES
  (2)  006660  001373                        BNE     64$
  (2)  006662  005726                        TST     (SP)+           ;/RESET STACK
  (2)  006664  017737  172532  001126        MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (2)                                                                ;/PREVIOUS COUNTER
  (2)  006672  012677  172522                MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (2)  006676  005737  001126                TST     $BDDAT          ;/RESTORE CSR
  (1)  006702  001004                        BNE     2$              ;/YES - NEXT TEST.
  (1)  006704  105766  177776                TSTB    -2(SP)          ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
  (1)                                                                ;/NOTE: CSR HAD BEEN PUT ON STACK.
  (1)  006710  100401                        BMI     2$              ;/NEXT TEST IF OVERFLOW.
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  006712  104006                        ERROR   6               ;/CLOCK FAILED TO COUNT AT
  (1)                                                                ;/RATE:100KHZ
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  006714  005077  172500        2$:     CLR     @ASR            ;/CLEAR THE CLOCK.
  (1)
  (1)
```

```
 6476
  (5)                                ;;**********************  ***********************************************
  (4)                                ;*TEST 40        *TEST THE ABILITY OF CLOCK TO COUNT AT 10KHZ RATE
  (5)
  (5)                                ;*
  (5)                                ;*THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
  (5)                                ;*TO COUNT AT 10KHZ RATE.
  (5)                                ;*
  (4)                                ;;*****************************************************************************
  (3)  006720  000004                TST40:  SCOPE
  (2)  006722  012737  000005  001160        MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)  006730  005077  172464                 CLR     @ASR            ;/CLEAR CLOCK
  (1)  006734  005077  172462                 CLR     @ABR            ;/CLEAR PRESET BUFFER
  (1)  006740  012777  000031  172452         MOV     #BIT0!30,@ASR   ;/START CLOCK, MODE0, RATE:10KHZ
  (1)  006746  005000                         CLR     R0              ;/NOW WE'LL DO A LITTLE DELAY. THIS DELAY
  (1)  006750  005200                1$:      INC     R0              ;/WILL AMOUNT TO APPROXIMATELY
  (1)  006752  001376                         BNE     1$              ;/369 MS.
  (1)
  (2)                                                                 ;/-RDCLK-
  (2)  006754  017746  172440                 MOV     @ASR,-(SP)      ;/SAVE CSR
  (2)  006760  052777  004007  172432         BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (2)                                                                 ;/THIS MUST BE DONE IN
  (2)                                                                 ;/ORDER TO XFERR COUNTER
  (2)                                                                 ;/TO BUFFER ON ST2.
  (2)  006766  052777  001000  172424         BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (2)  006774  012746  000010                 MOV     #8.,-(SP)       ;/NOW GENERATE
  (2)  007000  052777  000400  172412  64$:    BIS     #BIT8,@ASR
  (2)  007006  005316                         DEC     (SP)            ;/EIGHT ST1 PULSES
  (2)  007010  001373                         BNE     64$
  (2)  007012  005726                         TST     (SP)+           ;/RESET STACK
  (2)  007014  017737  172402  001126         MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (2)                                                                 ;/PREVIOUS COUNTER
  (2)  007022  012677  172372                 MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (2)  007026  005737  001126                 TST     $BDDAT          ;/RESTORE CSR
  (1)  007032  001004                         BNE     2$              ;/YES - NEXT TEST.
  (1)  007034  105766  177776                 TSTB    -2(SP)          ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
  (1)                                                                 ;/NOTE: CSR HAD BEEN PUT ON STACK.
  (1)  007040  100401                         BMI     2$              ;/NEXT TEST IF OVERFLOW.
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007042  104006                         ERROR   6               ;/CLOCK FAILED TO COUNT AT
  (1)                                                                 ;/RATE:10KHZ
  (2)
                                     ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007044  005077  172350        2$:      CLR     @ASR            ;/CLEAR THE CLOCK.
  (1)
  (1)
```

```
 6478
  (5)                                   ;;**********************  ********************************************
  (4)                                   ;*TEST 41        *TEST THE ABILITY OF CLOCK TO COUNT AT 1KHZ RATE
  (5)
  (5)                                   ;*
  (5)                                   ;*THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
  (5)                                   ;*TO COUNT AT 1KHZ RATE.
  (5)                                   ;*
  (4)                                   ;;*****************************************************************
  (3)  007050  000004                   TST41:  SCOPE
  (2)  007052  012737  000005  001160            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)  007060  005077  172334                    CLR     @ASR            ;/CLEAR CLOCK
  (1)  007064  005077  172332                    CLR     @ABR            ;/CLEAR PRESET BUFFER
  (1)  007070  012777  000041  172322            MOV     #BIT0!40,@ASR   ;/START CLOCK, MODE0, RATE:1KHZ
  (1)  007076  005000                            CLR     R0              ;/NOW WE'LL DO A LITTLE DELAY. THIS DELAY
  (1)  007100  005200                   1$:      INC     R0              ;/WILL AMOUNT TO APPROXIMATELY
  (1)  007102  001376                            BNE     1$              ;/369 MS.
  (1)
  (2)                                                                    ;/-RDCLK-
  (2)  007104  017746  172310                    MOV     @ASR,-(SP)      ;/SAVE CSR
  (2)  007110  052777  004007  172302            BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (2)                                                                    ;/THIS MUST BE DONE IN
  (2)                                                                    ;/ORDER TO XFERR COUNTER
  (2)                                                                    ;/TO BUFFER ON ST2.
  (2)  007116  052777  001000  172274            BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (2)  007124  012746  000010                    MOV     #8.,-(SP)       ;/NOW GENERATE
  (2)  007130  052777  000400  172262  64$:      BIS     #BIT8,@ASR
  (2)  007136  005316                            DEC     (SP)            ;/EIGHT ST1 PULSES
  (2)  007140  001373                            BNE     64$
  (2)  007142  005726                            TST     (SP)+           ;/RESET STACK
  (2)  007144  017737  172252  001126            MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (2)                                                                    ;/PREVIOUS COUNTER
  (2)  007152  012677  172242                    MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (2)  007156  005737  001126                    TST     $BDDAT          ;/RESTORE CSR
  (1)  007162  001004                            BNE     2$              ;/YES - NEXT TEST.
  (1)  007164  105766  177776                    TSTB    -2(SP)          ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
  (1)                                                                    ;/NOTE: CSR HAD BEEN PUT ON STACK.
  (1)  007170  100401                            BMI     2$              ;/NEXT TEST IF OVERFLOW.
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007172  104006                            ERROR   6               ;/CLOCK FAILED TO COUNT AT
  (1)                                                                    ;/RATE:1KHZ
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007174  005077  172220   2$:      CLR     @ASR            ;/CLEAR THE CLOCK.
  (1)
  (1)
```

```
 6480
  (5)                                   ;;**********************  ********************************************
  (4)                                   ;*TEST 42        *TEST THE ABILITY OF CLOCK TO COUNT AT 100HZ RATE
  (5)
  (5)                                   ;*
  (5)                                   ;*THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
  (5)                                   ;*TO COUNT AT 100HZ RATE.
  (5)                                   ;*
  (4)                                   ;;*************************************************************************
  (3)  007200  000004                   TST42:  SCOPE
  (2)  007202  012737  000005  001160           MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)  007210  005077  172204                   CLR     @ASR            ;/CLEAR CLOCK
  (1)  007214  005077  172202                   CLR     @ABR            ;/CLEAR PRESET BUFFER
  (1)  007220  012777  000051  172172           MOV     #BIT0!50,@ASR   ;/START CLOCK, MODE0, RATE:100HZ
  (1)  007226  005000                           CLR     R0              ;/NOW WE'LL DO A LITTLE DELAY. THIS DELAY
  (1)  007230  005200                   1$:     INC     R0              ;/WILL AMOUNT TO APPROXIMATELY
  (1)  007232  001376                           BNE     1$              ;/369 MS.
  (1)
  (2)                                                                   ;/-RDCLK-
  (2)  007234  017746  172160                   MOV     @ASR,-(SP)      ;/SAVE CSR
  (2)  007240  052777  004007  172152           BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (2)                                                                   ;/THIS MUST BE DONE IN
  (2)                                                                   ;/ORDER TO XFERR COUNTER
  (2)                                                                   ;/TO BUFFER ON ST2.
  (2)  007246  052777  001000  172144           BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (2)  007254  012746  000010                   MOV     #8.,-(SP)       ;/NOW GENERATE
  (2)  007260  052777  000400  172132   64$:    BIS     #BIT8,@ASR
  (2)  007266  005316                           DEC     (SP)            ;/EIGHT ST1 PULSES
  (2)  007270  001373                           BNE     64$
  (2)  007272  005726                           TST     (SP)+           ;/RESET STACK
  (2)  007274  017737  172122  001126           MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (2)                                                                   ;/PREVIOUS COUNTER
  (2)  007302  012677  172112                   MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (2)  007306  005737  001126                   TST     $BDDAT          ;/RESTORE CSR
  (1)  007312  001004                           BNE     2$              ;/YES - NEXT TEST.
  (1)  007314  105766  177776                   TSTB    -2(SP)          ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
  (1)                                                                   ;/NOTE: CSR HAD BEEN PUT ON STACK.
  (1)  007320  100401                           BMI     2$              ;/NEXT TEST IF OVERFLOW.
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007322  104006                           ERROR   6               ;/CLOCK FAILED TO COUNT AT
  (1)                                                                   ;/RATE:100HZ
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007324  005077  172070           2$:     CLR     @ASR            ;/CLEAR THE CLOCK.
  (1)
  (1)
```

```
 6482
  (5)                                   ;;********************* ********************************************
  (4)                                   ;*TEST 43        *TEST THE ABILITY OF CLOCK TO COUNT AT LINEFREQ RATE
  (5)
  (5)                                   ;*
  (5)                                   ;*THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
  (5)                                   ;*TO COUNT AT LINEFREQ RATE.
  (5)                                   ;*
  (4)                                   ;;*****************************************************************************
  (3)  007330  000004                   TST43:  SCOPE
  (2)  007332  012737  000005  001160           MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)  007340  005077  172054           10$:    CLR     @ASR            ;/CLEAR CLOCK
  (1)  007344  005077  172052                   CLR     @ABR            ;/CLEAR PRESET BUFFER
  (1)  007350  032777  002000  171562           BIT     #SW10,@SWR      ;TEST IT LINE FREQ IS ENABLED
  (3)  007356  001452                           BEQ     TST44           ;;BR IF SW10=0
  (1)  007360  052777  000100  172030           BIS     #BIT6,@KWL      ;NO- ENABLE LINE CLOCK
  (1)  007366                           10$:
  (1)  007366  012777  000071  172024           MOV     #BIT0!70,@ASR   ;/START CLOCK, MODE0, RATE:LINEFREQ
  (1)  007374  005000                           CLR     R0              ;/NOW WE'LL DO A LITTLE DELAY. THIS DELAY
  (1)  007376  005200                   1$:     INC     R0              ;/WILL AMOUNT TO APPROXIMATELY
  (1)  007400  001376                           BNE     1$              ;/369 MS.
  (1)
  (2)                                           ;/-RDCLK-
  (2)  007402  017746  172012                   MOV     @ASR,-(SP)      ;/SAVE CSR
  (2)  007406  052777  004007  172004           BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (2)                                           ;/THIS MUST BE DONE IN
  (2)                                           ;/ORDER TO XFERR COUNTER
  (2)                                           ;/TO BUFFER ON ST2.
  (2)  007414  052777  001000  171776           BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (2)  007422  012746  000010                   MOV     #8.,-(SP)       ;/NOW GENERATE
  (2)  007426  052777  000400  171764   64$:    BIS     #BIT8,@ASR
  (2)  007434  005316                           DEC     (SP)            ;/EIGHT ST1 PULSES
  (2)  007436  001373                           BNE     64$
  (2)  007440  005726                           TST     (SP)+           ;/RESET STACK
  (2)  007442  017737  171754  001126           MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (2)                                           ;/PREVIOUS COUNTER
  (2)  007450  012677  171744                   MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (2)  007454  005737  001126                   TST     $BDDAT          ;/RESTORE CSR
  (1)  007460  001004                           BNE     2$              ;/YES - NEXT TEST.
  (1)  007462  105766  177776                   TSTB    -2(SP)          ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
  (1)                                           ;/NOTE: CSR HAD BEEN PUT ON STACK.
  (1)  007466  100401                           BMI     2$              ;/NEXT TEST IF OVERFLOW.
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007470  104006                           ERROR   6               ;/CLOCK FAILED TO COUNT AT
  (1)                                           ;/RATE:LINEFREQ
  (2)
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

  (1)  007472  005077  171722   2$:    CLR     @ASR            ;/CLEAR THE CLOCK.
  (1)
  (1)
 6483  007476  042777  000100  171712           BIC     #BIT6,@KWL      ;DISABLE LINE CLOCK
 6484
 6485                                   ;;*****************************************************************************
```

```
  (3)                                   ;*TEST 44       *TEST THAT COUNTER DOESN'T COUNT WHEN ''SLAVE IN'' RATE IS SELECTED
  (3)                                   ;;********************* *********************************************
  (2)  007504  000004                   TST44:  SCOPE
  (1)  007506  012737  000020  001160            MOV     #20,$TIMES      ;;DO 20 ITERATIONS
 6486
 6487  007514  005077  171700                    CLR     @ASR            ;CLEAR CSR.
 6488  007520  005077  171676                    CLR     @ABR            ;CLEAR PRESET BUFFER.
 6489
 6490  007524  052777  000001  171666            BIS     #BIT0,@ASR      ;SET GO BIT, RATE:''SLAVE IN''
 6491
 6492  007532  052777  000400  171660            BIS     #BIT8,@ASR      ;GENERATE A MAINTENANCE ST1.
 6493  007540  005000                            CLR     R0
 6494
 6495  007542  005200                   1$:     INC     R0              ;NOW DO A SHORT DELAY.
 6496  007544  001376                            BNE     1$
 6497
 6498                                                                    ;/-RDCLK-
  (1)  007546  017746  171646                    MOV     @ASR,-(SP)      ;/SAVE CSR
  (1)  007552  052777  004007  171640            BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (1)                                                                   ;/THIS MUST BE DONE IN
  (1)                                                                   ;/ORDER TO XFERR COUNTER
  (1)                                                                   ;/TO BUFFER ON ST2.
  (1)  007560  052777  001000  171632            BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (1)  007566  012746  000010                    MOV     #8.,-(SP)       ;/NOW GENERATE
  (1)  007572  052777  000400  171620  64$:     BIS     #BIT8,@ASR
  (1)  007600  005316                            DEC     (SP)            ;/EIGHT ST1 PULSES
  (1)  007602  001373                            BNE     64$
  (1)  007604  005726                            TST     (SP)+           ;/RESET STACK
  (1)  007606  017737  171610  001126            MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (1)                                                                   ;/PREVIOUS COUNTER
  (1)  007614  012677  171600                    MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (1)  007620  005737  001126                    TST     $BDDAT          ;/RESTORE CSR
 6499  007624  001003                            BNE     2$              ;IF ANY COUNT, ERROR.
 6500  007626  105777  171566                    TSTB    @ASR            ;CHECK FOR OVERFLOW.
 6501  007632  100003                            BPL     3$              ;IF NO OVERFLOW, NEXT TEST
 6502
 6503  007634  005037  001124           2$:     CLR     $GDDAT          ;CLEAR FOR TYPEOUT, EXP'D ZERO.
 6504
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6505  007640  104011                            ERROR   11              ;CLOCK COUNTED WHEN RATE:''SLAVE IN''
 6506                                                                   ;WAS SELECTED.
 6507
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6508  007642  005077  171552           3$:     CLR     @ASR            ;CLEAR CLOCK'S CSR.
```

```
6510                                    ;;*****************************************************************
 (3)                                    ;*TEST 45       *TEST THAT THE CLOCK WILL COUNT IN MODE 1
 (3)                                    ;;*****************************************************************
 (2) 007646 000004                      TST45:  SCOPE
 (1) 007650 012737 000020 001160                MOV     #20,$TIMES      ;;DO 20 ITERATIONS
6511
6512 007656 005077 171536                       CLR     @ASR            ;CLEAR THE CSR.
6513 007662 012777 177777 171532                MOV     #-1,@ABR        ;PRESET BUFFER REG.
6514 007670 052777 000013 171522                BIS     #13,@ASR        ;START CLOCK,RATE:1 MHZ,MODE 1.
6515 007676 005000                              CLR     R0
6516 007700 105200                      1$:     INCB    R0              ;NOW DO A SHORT DELAY SO THAT THE
6517 007702 001376                              BNE     1$              ;CLOCK CAN COUNT TO OVERFLOW.
6518
6519 007704 105777 171510                       TSTB    @ASR            ;OVERFLOW SHOULD HAVE SET.
6520 007710 100401                              BMI     2$              ;GOTO NEXT TEST IF SO.
6521
6522
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6523 007712 104006                              ERROR   6               ;CLOCK FAILED TO COUNT UP AND SET
6524                                                                    ;OVERFLOW IN MODE 1.
6525
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

6526 007714                             2$:
6527
6528
6529                                            .SBTTL  *
6530                                            .SBTTL  *PHASE 4 CLOCK INTERRUPT TEST.
6531                                            .SBTTL  *
6532
6540
6541                                    ;;*****************************************************************
 (3)                                    ;*TEST 46       *TEST THAT THE CLOCK WILL INTERRUPT ON OVERFLOW
 (3)                                    ;;*****************************************************************
 (2) 007714 000004                      TST46:  SCOPE
 (1) 007716 012737 000020 001160                MOV     #20,$TIMES      ;;DO 20 ITERATIONS
6542
6543
 (1) 007724 012746 000340                       MOV     #340,-(SP)      ;PUT PRIORITY ON STACK.
 (1) 007730 012746 007736                       MOV     #64$,-(SP)      ;PUT RETURN ADDRESS ON STACK
 (1) 007734 000002                              RTI                     ;DO AN RTI, PUTS PRIORITY IN CPU.
 (1) 007736                             64$:
6544
6545 007736 005077 171456                       CLR     @ASR            ;CLEAR CLOCK'S CSR.
6546 007742 012777 177777 171452                MOV     #-1,@ABR        ;SET PRESET BUFFER TO ALL ONES.
6547
6548 007750 012777 000161 171442                MOV     #161,@ASR       ;START CLOCK, RATE:ST1.
6549 007756 052777 000400 171434                BIS     #BIT8,@ASR      ;GENERATE A MAINTENANCE ST1.
6550 007764 012777 010022 171432                MOV     #1$,@VECT1      ;SET INTERRUPT ADDR.
6551
 (1) 007772 012746 000000                       MOV     #0,-(SP)        ;PUT PRIORITY ON STACK.
 (1) 007776 012746 010004                       MOV     #65$,-(SP)      ;PUT RETURN ADDRESS ON STACK
 (1) 010002 000002                              RTI                     ;DO AN RTI, PUTS PRIORITY IN CPU.
 (1) 010004                             65$:
6552
```

```
 6553  010004  000240                          NOP                     ;STALL TIME
 6554  010006  000240                          NOP
 6555  010010  000240                          NOP
 6556  010012  005077  171402                  CLR     @ASR            ;REMOVE IRQ. ENABLE
 6557
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6558  010016  104007                          ERROR   7               ;CLOCK FAILED TO INTERRUPT.
 6559
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6560  010020  000402                          BR      2$
 6561  010022                        1$:
  (1)  010022  062706  000004                  ADD     #4,SP                   ;/ADD #4 TO STACK POINTER.
 6562  010026  005077  171366        2$:       CLR     @ASR            ;CLEAR THE CLOCK.
 6563  010032  013777  001426  171364          MOV     VECTP,@VECT1    ;2 RESTORE VECTOR.
 6564
 6565                                   ;;*****************************************************************
  (3)                                   ;*TEST 47       *TEST THAT ST2 WILL CAUSE AN INTERRUPT
  (3)                                   ;;*****************************************************************
  (2)  010040  000004                   TST47:  SCOPE
  (1)  010042  012737  000020  001160           MOV     #20,$TIMES      ;;DO 20 ITERATIONS
 6566
 6567
  (1)  010050  012746  000340                   MOV     #340,-(SP)      ;PUT PRIORITY ON STACK.
  (1)  010054  012746  010062                   MOV     #64$,-(SP)      ;PUT RETURN ADDRESS ON STACK
  (1)  010060  000002                           RTI                     ;DO AN RTI, PUTS PRIORITY IN CPU.
  (1)  010062                           64$:
 6568
 6569  010062  005077  171332                  CLR     @ASR            ;CLEAR CLOCKS CSR.
 6570  010066  012777  010140  171334          MOV     #1$,@VECT2      ;SET UP INTERRUPT VECTOR.
 6571  010074  012777  040001  171316          MOV     #BIT14!BIT0,@ASR;SET ''INT2'',AND GO BIT.
 6572  010102  052777  001000  171310          BIS     #BIT9,@ASR      ;GENERATE A MAINTENANCE ST2.
 6573
  (1)  010110  012746  000000                   MOV     #0,-(SP)        ;PUT PRIORITY ON STACK.
  (1)  010114  012746  010122                   MOV     #65$,-(SP)      ;PUT RETURN ADDRESS ON STACK
  (1)  010120  000002                           RTI                     ;DO AN RTI, PUTS PRIORITY IN CPU.
  (1)  010122                           65$:
 6574
 6575  010122  000240                          NOP                     ;STALL TIME
 6576  010124  000240                          NOP
 6577  010126  000240                          NOP
 6578  010130  005077  171264                  CLR     @ASR            ;REMOVE IRQ. ENABLE
 6579
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6580  010134  104007                          ERROR   7               ;CLOCK FAILED TO INTERRUPT ON ST2.
 6581
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6582  010136  000402                          BR      2$
 6583  010140                        1$:
  (1)  010140  062706  000004                  ADD     #4,SP                   ;/ADD #4 TO STACK POINTER.
 6584  010144  005077  171250        2$:       CLR     @ASR            ;CLEAR CLOCK'S CSR.
 6585  010150  013777  001432  171252          MOV     VECT2P,@VECT2   ;2 RESTORE VECTOR.
 6586
```

```
 6587                                   ;;*********************************************************************
  (3)                                   ;*TEST 50       *TEST THAT ST1 WILL CAUSE AN INTERRUPT
  (3)                                   ;;*********************************************************************
  (2)  010156  000004                   TST50:  SCOPE
  (1)  010160  012737  000020  001160            MOV     #20,$TIMES      ;;DO 20 ITERATIONS
 6588
 6589
  (1)  010166  012746  000340                    MOV     #340,-(SP)      ;PUT PRIORITY ON STACK.
  (1)  010172  012746  010200                    MOV     #64$,-(SP)      ;PUT RETURN ADDRESS ON STACK
  (1)  010176  000002                            RTI                     ;DO AN RTI, PUTS PRIORITY IN CPU.
  (1)  010200                           64$:
 6590  010200  005077  171214                    CLR     @ASR            ;2 CLEAR CSR
 6591  010204  012777  010250  171216            MOV     #1$,@VECT2      ;2 SET UP INTR. VECTOR.
 6592  010212  052777  040400  171200            BIS     #BIT14!BIT8,@ASR ;2 INTR ENABLE AND ST1.
 6593
  (1)  010220  012746  000000                    MOV     #0,-(SP)        ;PUT PRIORITY ON STACK.
  (1)  010224  012746  010232                    MOV     #65$,-(SP)      ;PUT RETURN ADDRESS ON STACK
  (1)  010230  000002                            RTI                     ;DO AN RTI, PUTS PRIORITY IN CPU.
  (1)  010232                           65$:
 6594  010232  000240                            NOP                     ;2 INTR. FROM HERE.
 6595  010234  000240                            NOP
 6596  010236  000240                            NOP
 6597  010240  005077  171154                    CLR     @ASR            ;REMOVE IRQ. ENABLE
 6598
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6599  010244  104007                            ERROR   7               ;2 CLOCK FAILED TO INTR. ON ST1.
 6600  010246  000402                            BR      2$
 6601  010250                           1$:
  (1)  010250  062706  000004                    ADD     #4,SP                   ;/ADD #4 TO STACK POINTER.
 6602  010254  005077  171140           2$:     CLR     @ASR            ;2 CLEAR CSR.
 6603  010260  013777  001432  171142            MOV     VECT2P,@VECT2   ;2 RESTORE INTR. VECTOR.
 6604
  (1)  010266  012746  000000                    MOV     #0,-(SP)        ;PUT PRIORITY ON STACK.
  (1)  010272  012746  010300                    MOV     #66$,-(SP)      ;PUT RETURN ADDRESS ON STACK
  (1)  010276  000002                            RTI                     ;DO AN RTI, PUTS PRIORITY IN CPU.
  (1)  010300                           66$:
 6605
 6606                                           .SBTTL  *
 6607                                           .SBTTL  *PHASE 5 ADVANCED TESTING
 6608                                           .SBTTL  *
 6609
```

```
 6697                                   ;;*****************************************************************
  (3)                                   ;*TEST 51       *TEST THAT THE 'FOR' BIT WILL SET ON 2 ST2'S
  (3)                                   ;;*****************************************************************
  (2)  010300  000004                   TST51:  SCOPE
 6698  010302  005077  171112                   CLR     @ASR            ;START WITH CSR CLEAR.
 6699  010306  005277  171106                   INC     @ASR            ;SET GO BIT.
 6700  010312  052777  001000  171100           BIS     #BIT9,@ASR      ;GENERATE THE 1ST ST2 PULSE.
 6701  010320  052777  001000  171072           BIS     #BIT9,@ASR      ;GENERATE 2ND ST2 PULSE.
 6702                                                                   ;THIS SHOULD CAUSE FOR BIT TO SET.
 6703  010326  032777  010000  171064           BIT     #BIT12,@ASR     ;DID FOR BIT SET?
 6704  010334  001007                           BNE     1$              ;YES-THEN NEXT TEST.
 6705
 6706  010336  017737  171056  001126           MOV     @ASR,$BDDAT     ;RECORD CSR.
 6707  010344  012737  110001  001124           MOV     #BIT15!BIT12!BIT0,$GDDAT        ;RECORD S/B.
 6708
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6709  010352  104001                           ERROR   1               ;ERROR-'FOR' BIT FAILED TO SET ON
 6710                                                                   ;ON TWO SUCCESIVE ST2 PULSES.
 6711
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6712  010354                           1$:
 6713                                   ;;*****************************************************************
  (3)                                   ;*TEST 52       *TEST THAT THE 'FOR' BIT WILL SET ON 2 ST1'S
  (3)                                   ;;*****************************************************************
  (2)  010354  000004                   TST52:  SCOPE
 6714  010356  005077  171036                   CLR     @ASR            ;START WITH THE CSR CLEAR.
 6715  010362  005277  171032                   INC     @ASR            ;SET GO BIT.
 6716  010366  052777  000400  171024           BIS     #BIT08,@ASR     ;GENERATE AN ST1.
 6717  010374  052777  000400  171016           BIS     #BIT08,@ASR     ;GENERATE ANOTHER ST1.
 6718                                                                   ;AT THIS POINT THE 'FOR' BIT SHOULD HAVE SET
 6719  010402  032777  010000  171010           BIT     #BIT12,@ASR     ;DID THE FOR BIT SET?
 6720  010410  001007                           BNE     TST53           ;;
 6721  010412  017737  171002  001126           MOV     @ASR,$BDDAT     ;RECORD CSR.
 6722  010420  012737  012001  001124           MOV     #BIT10!BIT12!BIT0,$GDDAT        ;RECORD S/B.
 6723
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6724  010426  104001                           ERROR   1               ;ERROR- 'FOR' BIT FAILED TO SET ON
 6725                                                                   ;TWO SUCCESIVE ST1 PULSES.
 6726
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
 6728                                   ;;*****************************************************************
  (3)                                   ;*TEST 53       *TEST THAT FOR BIT WILL SET ON TWO OVERFLOWS
  (3)                                   ;;*****************************************************************
  (2)  010430  000004                   TST53:  SCOPE
  (1)  010432  012737  000002  001160            MOV     #2,$TIMES       ;;DO 2 ITERATIONS
 6729
 6730  010440  005077  170754                    CLR     @ASR            ;START WITH CSR CLEAR.
 6731  010444  012777  177777  170750            MOV     #-1,@ABR        ;PRELOAD BUFFER REG.
 6732  010452  052777  000013  170740            BIS     #13,@ASR        ;START CLOCK,MODE 1,1MHZ.
 6733  010460  005000                            CLR     R0              ;NOW DO A SHORT DELAY.
 6734  010462  105200                   1$:      INCB    R0              ;DURING THIS DELAY, THE CLOCK WILL
 6735  010464  001376                            BNE     1$              ;HAVE OVERFLOWED TWICE-
 6736                                                                    ;THIS SHOULD CAUSE THE FOR BIT TO SET.
 6737  010466  032777  010000  170724            BIT     #BIT12,@ASR     ;DID FOR SET?
 6738  010474  001007                            BNE     2$              ;YES-NEXT TEST.
 6739  010476  017737  170716  001126            MOV     @ASR,$BDDAT     ;NO-RECORD THE CSR.
 6740  010504  012737  010213  001124            MOV     #010213,$GDDAT  ;RECORD S/B.
 6741
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6742  010512  104001                            ERROR   1               ;ERROR FOR BIT FAILED TO SET ON
 6743                                                                    ; TWO SUCCESSIVE OVERFLOWS.
 6744
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6745  010514                           2$:
 6746                                   ;;*****************************************************************
  (3)                                   ;*TEST 54       *TEST THAT FOR BIT WILL CLEAR IF GO BIT IS SET
  (3)                                   ;;*****************************************************************
  (2)  010514  000004                   TST54:  SCOPE
 6747  010516  012777  000001  170674            MOV     #BIT0,@ASR      ;CLEAR CSR,SET GO BIT.
 6748  010524  052777  001000  170666            BIS     #BIT9,@ASR      ;SET 1ST ST2 PULSE.
 6749  010532  052777  001000  170660            BIS     #BIT9,@ASR      ;GENERATE 2ND ST2 PULSE.
 6750                                                                    ;FOR BIT SETS HERE.
 6751  010540  042777  100001  170652            BIC     #BIT0!BIT15,@ASR        ;CLEAR GO BITAND ST2 FLAG.
 6752  010546  052777  000001  170644            BIS     #BIT0,@ASR      ;SET THE ''GO'' BIT AGAIN -
 6753                                                                    ;SHOULD CLEAR FOR BIT.
 6754  010554  017737  170640  001126            MOV     @ASR,$BDDAT     ;READ THE CSR.
 6755  010562  012737  100001  001124            MOV     #100001,$GDDAT  ;RECORD WHAT CSR S/B.
 6756  010570  032737  010000  001126            BIT     #BIT12,$BDDAT   ;DID FOR BIT CLEAR?
 6757  010576  001401                            BEQ     1$              ;YES NEXT TEST.
 6758
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6759  010600  104001                            ERROR   1               ;FOR BIT FALED TO CLEAR
 6760                                                                    ;WHEN ''GO'' BIT WAS SET.
 6761
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6762  010602  005077  170612           1$:      CLR     @ASR            ;CLEAR THE CSR.
```

```
 6764                                   ;;*****************************************************************
  (3)                                   ;*TEST 55       *TEST THAT WE CAN DISABLE THE INTERNAL OSC
  (3)                                   ;;*****************************************************************
  (2)  010606  000004                   TST55:  SCOPE
  (1)  010610  012737  000005  001160            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
 6765
 6766  01061c  005077  170576                    CLR     @ASR            ;CLEAR THE CSR
 6767  010622  005077  170574                    CLR     @ABR            ;CLEAR THE PRESET BUFFER
 6768  010626  005037  001124                    CLR     $GDDAT          ;CLEAR EXPED.
 6769
 6770  010632  012777  004000  170560            MOV     #BIT11,@ASR     ;DISABLE THE INTERNAL OSC.
 6771  010640  052777  000011  170552            BIS     #BIT3.BIT0,@ASR ;START CLOCK:RATE 1MHZ.
 6772  010646  005000                    CLR     R0
 6773  010650  105200                    1$:     INCB    R0              ;DELAY A SHORT TIME.
 6774  010652  001376                    BNE     1$
 6775                                                                   ;/-RDCLK-
  (1)  010654  017746  170540                    MOV     @ASR,-(SP)      ;/SAVE CSR
  (1)  010660  052777  004007  170532            BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (1)                                                                    ;/THIS MUST BE DONE IN
  (1)                                                                    ;/ORDER TO XFERR COUNTER
  (1)                                                                    ;/TO BUFFER ON ST2.
  (1)  010666  052777  001000  170524            BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (1)  010674  012746  000010                    MOV     #8.,-(SP)       ;/NOW GENERATE
  (1)  010700  052777  000400  170512  64$:     BIS     #BIT8,@ASR
  (1)  010706  005316                    DEC     (SP)            ;/EIGHT ST1 PULSES
  (1)  010710  001373                    BNE     64$
  (1)  010712  005726                    TST     (SP)+           ;/RESET STACK
  (1)  010714  017737  170502  001126            MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (1)                                                                    ;/PREVIOUS COUNTER
  (1)  010722  012677  170472                    MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (1)  010726  005737  001126                    TST     $BDDAT          ;/RESTORE CSR
 6776  010732  001401                    BEQ     2$              ;NO - GOOD - NEXT TEST.
 6777
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6778  010734  104011                    ERROR   11              ;CLOCK DISABLE INTERNAL
 6779                                                           ;OSC. DID NOT WORK.
 6780
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6781  010736  005077  170456            2$:     CLR     @ASR            ;CLEAR THE CSR.
```

```
 6783                                   ;;*****************************************************************
  (3)                                   ;*TEST 56       *TEST THAT CLOCK CAN BE COUNTED USING MAINTENANCE OSC
  (3)                                   ;;*****************************************************************
  (2)  010742  000004                   TST56:  SCOPE
 6784
 6785  010744  005077  170450                   CLR     @ASR            ;CLEAR THE CSR.
 6786  010750  005077  170446                   CLR     @ABR            ;CLEAR THE PRESET BUFFER.
 6787  010754  052777  004000  170436           BIS     #BIT11,@ASR     ;DISABLE THE INTERNAL OSC.
 6788  010762  052777  000011  170430           BIS     #BIT3!BIT0,@ASR ;START CLOCK, 1MHZ, GO.
 6789  010770  012700  177754                   MOV     #-20.,R0        ;SET TO COUNT 20 TIMES
 6790
 6791  010774  052777  000400  170416  1$:      BIS     #BIT8,@ASR      ;2 GENERATE 1 MAINTENANCE OSC.
 6792                                                                   ;NOTE: AT 1MHZ, IT TAKES 10
 6793                                                                   ;MAINT. OSC TO EQUAL 1 COUNT
 6794  011002  005200                           INC     R0              ;DO 20 MAINTENANCE OSC.
 6795  011004  001373                           BNE     1$
 6796
 6797                                                                   ;/-RDCLK-
  (1)  011006  017746  170406                   MOV     @ASR,-(SP)      ;/SAVE CSR
  (1)  011012  052777  004007  170400           BIS     #4007,@ASR      ;/SET MODE 3,DIS INTR OSC NO RATE
  (1)                                                                   ;/THIS MUST BE DONE IN
  (1)                                                                   ;/ORDER TO XFERR COUNTER
  (1)                                                                   ;/TO BUFFER ON ST2.
  (1)  011020  052777  001000  170372           BIS     #BIT9,@ASR      ;/GENERATE ON ST2 PULSE
  (1)  011026  012746  000010                   MOV     #8.,-(SP)       ;/NOW GENERATE
  (1)  011032  052777  000400  170360  64$:     BIS     #BIT8,@ASR
  (1)  011040  005316                           DEC     (SP)            ;/EIGHT ST1 PULSES
  (1)  011042  001373                           BNE     64$
  (1)  011044  005726                           TST     (SP)+           ;/RESET STACK
  (1)  011046  017737  170350  001126           MOV     @ABR,$BDDAT     ;/READ THE PRESET BUFFER,
  (1)                                                                   ;/PREVIOUS COUNTER
  (1)  011054  012677  170340                   MOV     (SP)+,@ASR      ;/CONTENTS ARE IN $BDDAT.
  (1)  011060  005737  001126                   TST     $BDDAT          ;/RESTORE CSR
 6798  011064  001001                           BNE     2$              ;YES - NEXT TEST.
 6799
 6800
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6801  011066  104011                           ERROR   11              ;ERROR COULD NOT COUNT USING
 6802                                                                   ;MAINTENANCE OSC.
 6803
                                       ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6804  011070  005077  170324           2$:     CLR     @ASR            ;CLEAR THE CSR.
 6805
 6935
```

```
 6937
  (1)
  (5)                                   ;;*********************************************************************
  (4)                                   ;*TEST 57        *TEST THE CLOCK'S 1MHZ DIVIDER
  (5)                                   ;*
  (5)                                   ;*IN THIS TEST WE WILL CHECH OUT PART OF THE DIVIDER CHAIN LOGIC.
  (5)                                   ;*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS,THIS TEST CHECKS
  (5)                                   ;*THAT 100,000 MAIN OSC PULSES GIVES US 10000. COUNTS AT 1MHZ RATE.
  (4)                                   ;;*********************************************************************
  (3)  011074  000004                   TST57:  SCOPE
  (2)  011076  012737  000005  001160           MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)                                                                   ;/-DIVCH-
  (1)  011104  012700  000012           1$:     MOV     #10.,R0
  (1)  011110  005077  170304           2$:     CLR     @ASR
  (1)  011114  005077  170302                   CLR     @ABR
  (1)  011120  052777  004000  170272           BIS     #BIT11,@ASR             ;/DISABLE INTERNAL OSC.
  (1)  011126  052777  000017  170264           BIS     #7!10,@ASR              ;/SET GO,RATE: 1MHZ.,MODE 3.
  (1)
  (1)  011134  012701  023420                   MOV     #10000.,R1              ;/DO THAT MANY TIMES.
  (1)  011140  052777  000400  170252           BIS     #BIT8,@ASR              ;/GENERATE AN OSC PULSE.
  (1)  011146  005301                           DEC     R1
  (1)  011150  001373                           BNE     2$
  (1)  011152  005300                           DEC     R0
  (1)  011154  001367                           BNE     1$
  (1)  011156  052777  001000  170234           BIS     #BIT9,@ASR      ;/ST2
  (1)  011164  012700  000010                   MOV     #8.,R0
  (1)  011170  052777  000400  170222  3$:      BIS     #BIT8,@ASR
  (1)  011176  005300                           DEC     R0
  (1)  011200  001373                           BNE     3$
  (1)
  (1)  011202  017737  170214  001126           MOV     @ABR,$BDDAT             ;/READ COUNT.
  (1)  011210  012737  023420  001446           MOV     #10000.,$TMP0           ;/EXPECT THESE MANY COUNTS.
  (1)  011216  023737  001446  001126           CMP     $TMP0,$BDDAT            ;/DID WE GET THEM??
  (3)  011224  001401                           BEQ     TST60           ;;
  (1)  011226  104011                           ERROR   11                      ;/ERROR  100,000. OSC PULSES
  (1)                                                                   ;/DID NOT GENERATE 10000.
  (1)                                                                   ;/COUNTS AT RATE 1MHZ
  (1)
```

```
6939
 (1)
 (5)                                    ;;*****************************************************************
 (4)                                    ;*TEST 60       *TEST THE CLOCK'S 100KHZ DIVIDER
 (5)                                    ;*
 (5)                                    ;*IN THIS TEST WE WILL CHECH OUT PART OF THE DIVIDER CHAIN LOGIC.
 (5)                                    ;*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS,THIS TEST CHECKS
 (5)                                    ;*THAT 100,000 MAIN OSC PULSES GIVES US 1000. COUNTS AT 100KHZ RATE.
 (4)                                    ;;*****************************************************************
 (3)  011230  000004                    TST60:  SCOPE
 (2)  011232  012737  000005  001160            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
 (1)                                                                    ;/-DIVCH-
 (1)  011240  012700  000012            1$:     MOV     #10.,R0
 (1)  011244  005077  170150                    CLR     @ASR
 (1)  011250  005077  170146                    CLR     @ABR
 (1)  011254  052777  004000  170136            BIS     #BIT11,@ASR             ;/DISABLE INTERNAL OSC.
 (1)  011262  052777  000027  170130            BIS     #7!20,@ASR              ;/SET GO,RATE: 100KHZ.,MODE 3.
 (1)
 (1)  011270  012701  023420            1$:     MOV     #10000.,R1              ;/DO THAT MANY TIMES.
 (1)  011274  052777  000400  170116    2$:     BIS     #BIT8,@ASR              ;/GENERATE AN OSC PULSE.
 (1)  011302  005301                            DEC     R1
 (1)  011304  001373                            BNE     2$
 (1)  011306  005300                            DEC     R0
 (1)  011310  001367                            BNE     1$
 (1)  011312  052777  001000  170100            BIS     #BIT9,@ASR      ;/ST2
 (1)  011320  012700  000010                    MOV     #8.,R0
 (1)  011324  052777  000400  170066    3$:     BIS     #BIT8,@ASR
 (1)  011332  005300                            DEC     R0
 (1)  011334  001373                            BNE     3$
 (1)
 (1)  011336  017737  170060  001126            MOV     @ABR,$BDDAT             ;/READ COUNT.
 (1)  011344  012737  001750  001446            MOV     #1000.,$TMP0            ;/EXPECT THESE MANY COUNTS.
 (1)  011352  023737  001446  001126            CMP     $TMP0,$BDDAT            ;/DID WE GET THEM??
 (3)  011360  001401                            BEQ     TST61           ;;
 (1)  011362  104011                            ERROR   11              ;/ERROR  100,000. OSC PULSES
 (1)                                                                    ;/DID NOT GENERATE 1000.
 (1)                                                                    ;/COUNTS AT RATE 100KHZ
 (1)
```

```
 6941
  (1)
  (5)                                   ;;*********************************************************************
  (4)                                   ;*TEST 61        *TEST THE CLOCK'S 10KHZ DIVIDER
  (5)                                   ;*
  (5)                                   ;*IN THIS TEST WE WILL CHECH OUT PART OF THE DIVIDER CHAIN LOGIC.
  (5)                                   ;*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS,THIS TEST CHECKS
  (5)                                   ;*THAT 100,000 MAIN OSC PULSES GIVES US 100. COUNTS AT 10KHZ RATE.
  (4)                                   ;;*********************************************************************
  (3)  011364  000004                   TST61:  SCOPE
  (2)  011366  012737  000005  001160            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)                                                                     ;/-DIVCH-
  (1)  011374  012700  000012                    MOV     #10.,R0
  (1)  011400  005077  170014                    CLR     @ASR
  (1)  011404  005077  170012                    CLR     @ABR
  (1)  011410  052777  004000  170002            BIS     #BIT11,@ASR             ;/DISABLE INTERNAL OSC.
  (1)  011416  052777  000037  167774            BIS     #7!30,@ASR              ;/SET GO,RATE: 10KHZ.,MODE 3.
  (1)
  (1)  011424  012701  023420          1$:       MOV     #10000.,R1              ;/DO THAT MANY TIMES.
  (1)  011430  052777  000400  167762  2$:       BIS     #BIT8,@ASR              ;/GENERATE AN OSC PULSE.
  (1)  011436  005301                            DEC     R1
  (1)  011440  001373                            BNE     2$
  (1)  011442  005300                            DEC     R0
  (1)  011444  001367                            BNE     1$
  (1)  011446  052777  001000  167744            BIS     #BIT9,@ASR      ;/ST2
  (1)  011454  012700  000010                    MOV     #8.,R0
  (1)  011460  052777  000400  167732  3$:       BIS     #BIT8,@ASR
  (1)  011466  005300                            DEC     R0
  (1)  011470  001373                            BNE     3$
  (1)
  (1)  011472  017737  167724  001126            MOV     @ABR,$BDDAT             ;/READ COUNT.
  (1)  011500  012737  000144  001446            MOV     #100.,$TMP0             ;/EXPECT THESE MANY COUNTS.
  (1)  011506  023737  001446  001126            CMP     $TMP0,$BDDAT            ;/DID WE GET THEM??
  (3)  011514  001401                            BEQ     TST62           ;;
  (1)  011516  104011                            ERROR   11              ;/ERROR  100,000. OSC PULSES
  (1)                                                                     ;/DID NOT GENERATE 100.
  (1)                                                                     ;/COUNTS AT RATE 10KHZ
  (1)
```

```
 6943
  (1)
  (5)                                   ;;*****************************************************************
  (4)                                   ;*TEST 62       *TEST THE CLOCK'S 1KHZ DIVIDER
  (5)                                   ;*
  (5)                                   ;*IN THIS TEST WE WILL CHECH OUT PART OF THE DIVIDER CHAIN LOGIC.
  (5)                                   ;*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS,THIS TEST CHECKS
  (5)                                   ;*THAT 100,000 MAIN OSC PULSES GIVES US 10. COUNTS AT 1KHZ RATE.
  (4)                                   ;;*****************************************************************
  (3)  011520  000004                   TST62:  SCOPE
  (2)  011522  012737  000005  001160           MOV     #5,$TIMES       ;;DO 5 ITERATIONS
  (1)                                                                   ;/-DIVCH-
  (1)  011530  012700  000012                   MOV     #10.,R0
  (1)  011534  005077  167660                   CLR     @ASR
  (1)  011540  005077  167656                   CLR     @ABR
  (1)  011544  052777  004000  167646           BIS     #BIT11,@ASR             ;/DISABLE INTERNAL OSC.
  (1)  011552  052777  000047  167640           BIS     #7!40,@ASR              ;/SET GO,RATE: 1KHZ.,MODE 3.
  (1)
  (1)  011560  012701  023420           1$:     MOV     #10000.,R1              ;/DO THAT MANY TIMES.
  (1)  011564  052777  000400  167626   2$:     BIS     #BIT8,@ASR              ;/GENERATE AN OSC PULSE.
  (1)  011572  005301                           DEC     R1
  (1)  011574  001373                           BNE     2$
  (1)  011576  005300                           DEC     R0
  (1)  011600  001367                           BNE     1$
  (1)  011602  052777  001000  167610           BIS     #BIT9,@ASR      ;/ST2
  (1)  011610  012700  000010                   MOV     #8.,R0
  (1)  011614  052777  000400  167576   3$:     BIS     #BIT8,@ASR
  (1)  011622  005300                           DEC     R0
  (1)  011624  001373                           BNE     3$
  (1)
  (1)  011626  017737  167570  001126           MOV     @ABR,$BDDAT             ;/READ COUNT.
  (1)  011634  012737  000012  001446           MOV     #10.,$TMP0              ;/EXPECT THESE MANY COUNTS.
  (1)  011642  023737  001446  001126           CMP     $TMP0,$BDDAT            ;/DID WE GET THEM??
  (3)  011650  001401                           BEQ     TST63           ;;
  (1)  011652  104011                           ERROR   11                      ;/ERROR  100,000. OSC PULSES
  (1)                                                                   ;/DID NOT GENERATE 10.
  (1)                                                                   ;/COUNTS AT RATE 1KHZ
  (1)
```

```
6945
 (1)
 (5)                                    ;;*****************************************************************
 (4)                                    ;*TEST 63       *TEST THE CLOCK'S 100HZ DIVIDER
 (5)                                    ;*
 (5)                                    ;*IN THIS TEST WE WILL CHECH OUT PART OF THE DIVIDER CHAIN LOGIC.
 (5)                                    ;*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS,THIS TEST CHECKS
 (5)                                    ;*THAT 100,000 MAIN OSC PULSES GIVES US 1 COUNTS AT 100HZ RATE.
 (4)                                    ;;*****************************************************************
 (3)  011654  000004                    TST63:  SCOPE
 (2)  011656  012737  000005  001160            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
 (1)                                                                    ;/-DIVCH-
 (1)  011664  012700  000012            1$:     MOV     #10.,R0
 (1)  011670  005077  167524                    CLR     @ASR
 (1)  011674  005077  167522                    CLR     @ABR
 (1)  011700  052777  004000  167512            BIS     #BIT11,@ASR             ;/DISABLE INTERNAL OSC.
 (1)  011706  052777  000057  167504            BIS     #7!50,@ASR              ;/SET GO,RATE: 100HZ.,MODE 3.
 (1)
 (1)  011714  012701  023420            1$:     MOV     #10000.,R1              ;/DO THAT MANY TIMES.
 (1)  011720  052777  000400  167472    2$:     BIS     #BIT8,@ASR              ;/GENERATE AN OSC PULSE.
 (1)  011726  005301                            DEC     R1
 (1)  011730  001373                            BNE     2$
 (1)  011732  005300                            DEC     R0
 (1)  011734  001367                            BNE     1$
 (1)  011736  052777  001000  167454            BIS     #BIT9,@ASR      ;/ST2
 (1)  011744  012700  000010                    MOV     #8.,R0
 (1)  011750  052777  000400  167442    3$:     BIS     #BIT8,@ASR
 (1)  011756  005300                            DEC     R0
 (1)  011760  001373                            BNE     3$
 (1)
 (1)  011762  017737  167434  001126            MOV     @ABR,$BDDAT             ;/READ COUNT.
 (1)  011770  012737  000001  001446            MOV     #1,$TMP0                ;/EXPECT THESE MANY COUNTS.
 (1)  011776  023737  001446  001126            CMP     $TMP0,$BDDAT            ;/DID WE GET THEM??
 (3)  012004  001401                            BEQ     TST64           ;;
 (1)  012006  104011                            ERROR   11              ;/ERROR  100,000. OSC PULSES
 (1)                                                                    ;/DID NOT GENERATE 1
 (1)                                                                    ;/COUNTS AT RATE 100HZ
 (1)
6946
```

```
                                ;;********************* ******************* *********************
                                ;*TEST 64        *TEST THE CLOCK'S MODE 2 OPERATION
                                ;*
                                ;*IN THIS TEST WE'LL CHECK MODE 2 OPERATION
                                ;*MODE 2: EXTERNAL EVENTS TIMING MODE
                                ;*SETTING THE GO BIT CAUSES THE COUNTER TO BEGIN COUNTING FROM
                                ;*ZERO AND TO FREE-RUN UNTIL THE GO BIT IS WRITTEN
                                ;*TO A ZERO THE COUNTER WILL CONTINUE COUNTING AFTER
                                ;*OVERFLOW. AN EXTERNAL PULSE FROM SCHMITZ TRIGGER 2
                                ;*(WHEN ST2 GO ENABLE IS A '0') CAUSES DATA TO
                                ;*TRANSFER FROM THE COUNTER TO THE BUFFER/PRESET REG.
                                ;*WHILE THE COUNTER CONTINUES TO RUN.
                                ;*
                                ;*TO TEST THIS MODE, WE'LL DISABLE THE INTERNAL OSC AND USE
                                ;*MAINTENANCE OSC PULSES AS WELL AS A MAINTENANCE
                                ;*ST2.
                                ;*
                                ;;*****************************************************************
012010  000004                  TST64:  SCOPE
012012  012737  000020  001160          MOV     #20,$TIMES      ;;DO 20 ITERATIONS

012020  005077  167374                  CLR     @ASR            ;CLEAR THE CSR.
012024  005077  167372                  CLR     @ABR            ;CLEAR THE PRESET REG.
012030  012777  000060  167362          MOV     #60,@ASR        ;SET RATE ST1
012036  052777  000005  167354          BIS     #5,@ASR         ;START CLOCK MODE 2.

012044  012700  177776          1$:     MOV     #-2.,R0         ;SET TO GIVE 2 ST1 PULSES.
012050  052777  000400  167342  2$:     BIS     #BIT8,@ASR      ; GENERATE AN ST1 PULSE
012056  005200                          INC     R0
012060  001373                          BNE     2$              ;IF NOT DONE 2 TIMES, LOOP.

012062                          3$:
012062  052777  001000  167330          BIS     #BIT9,@ASR      ;HERE'S THE BIGGIE! AN ST2 HAS BEEN GENERATED
012070  012737  000002  001124          MOV     #2,$GDDAT       ;THE PRESET BUFFER SHOULD BE 2.
012076  017737  167320  001126          MOV     @ABR,$BDDAT     ;READ THE PRESET BUFFER.
012104  023737  001126  001124          CMP     $BDDAT,$GDDAT   ;DID A COUNTER TO PRESET BUFFER OCCUR?
012112  001402                          BEQ     4$              ;YES - NEXT SUBTEST.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

012114  104005                          ERROR   5               ;A COUNTER TO PRESET BUFFER DID NOT
                                                                ;HAPPEN PROPERLY.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

012116  000427                          BR      5$

012120                          4$:
                                                                ;/-RDCLK1-
012120  017746  167274                  MOV     @ASR,-(SP)      ;/SAVE CSR CONTENTS.
012124  052777  000005  167266          BIS     #5,@ASR         ;/SET TO MODE 2,GO
012132  042777  100000  167260          BIC     #BIT15,@ASR     ;/CLR ST FLAG.
012140  052777  001000  167252          BIS     #BIT9,@ASR      ;/GENERATE ST2 PULSE.
012146  017737  167250  001126          MOV     @ABR,$BDDAT     ;/READ COUNT REG.
012154  052677  167240                  BIS     (SP)+,@ASR      ;/RESTORE CSR.
```

```
  (1)  012160  005737  001126          TST     $BDDAT          ;/PREVIOUS CONTENTS OF COUNT REG
  (1)                                                          ;/IN $BDDAT.
 6991  012164  023737  001124  001126  CMP     $GDDAT,$BDDAT   ;WAS THE COUNTER ACCIDENTLY ZEROED?
 6992  012172  001401                  BEQ     5$              ;NO - NEXT TEST.
 6993
                                      ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6994  012174  104005                  ERROR   5               ;THE COUNT REGISTER SHOULD NOT
 6995                                                         ;HAVE BEEN EFFECTED BY THE ST2
 6996                                                         ;IN MODE 2.
 6997
                                      ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 6998  012176                          5$:
 6999  012176  042777  000070  167214          BIC     #70,@ASR        ;CLEAR THE RATE BITS.
 7000  012204  005037  001124                  CLR     $GDDAT
 7001  012210  052777  000400  167202          BIS     #BIT8,@ASR      ;SET ST1 SIM.
 7002                                                         ;/-RDCLK1-
  (1)  012216  017746  167176                  MOV     @ASR,-(SP)      ;/SAVE CSR CONTENTS.
  (1)  012222  052777  000005  167170          BIS     #5,@ASR         ;/SET TO MODE 2,GO
  (1)  012230  042777  100000  167162          BIC     #BIT15,@ASR     ;/CLR ST FLAG.
  (1)  012236  052777  001000  167154          BIS     #BIT9,@ASR      ;/GENERATE ST2 PULSE.
  (1)  012244  017737  167152  001126          MOV     @ABR,$BDDAT     ;/READ COUNT REG.
  (1)  012252  052677  167142                  BIS     (SP)+,@ASR      ;/RESTORE CSR.
  (1)  012256  005737  001126                  TST     $BDDAT          ;/PREVIOUS CONTENTS OF COUNT REG
  (1)                                                          ;/IN $BDDAT.
 7003  012262  001401                          BEQ     TST65           ;;
 7004
                                      ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 7005  012264  104001                  ERROR   1               ;CLOCK COUNT REG SHOULD BE ZERO
 7006
                                      ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 7007
 7015
 7016                                 ;;*****************************************************************
  (3)                                 ;*TEST 65       *TEST THE CLOCK'S MODE 3 OPERATION
  (4)                                 ;*
  (4)                                 ;*IN THIS TEST WE'LL CHECK MODE 3 OPERATION.
  (4)                                 ;*MODE 3 IS JUST LIKE MODE 2 EXCEPT THAT THE COUNT
  (4)                                 ;*REG IS ZEROED AFTER AN ST2.
  (4)                                 ;*
  (3)                                 ;*****************************************************************
  (2)  012266  000004                  TST65:  SCOPE
  (1)  012270  012737  000020  001160          MOV     #20,$TIMES      ;;DO 20 ITERATIONS
 7017
 7018  012276  005077  167116                  CLR     @ASR            ;CLEAR THE CSR.
 7019  012302  005077  167114                  CLR     @ABR            ;CLEAR THE BUFFER REG.
 7020  012306  012777  000060  167104          MOV     #60,@ASR        ;SET RATE: ST1
 7021  012314  052777  000007  167076          BIS     #7,@ASR         ;START CLOCK: MODE 3
 7022
 7023  012322  012700  177776          1$:     MOV     #-2.,R0         ;SET TO GIVE 2 ST1 PULSES
 7024  012326  052777  000400  167064  2$:     BIS     #BIT8,@ASR      ;GENERATE AN ST1 PULSE
 7025  012334  005200                          INC     R0
 7026  012336  001373                          BNE     2$              ;IF NOT DONE 2 TIMES, LOOP.
```

```
7027
7028  012340                                  3$:
7029  012340  052777  001000  167052            BIS     #BIT9,@ASR      ;HERE'S THE BIGGIE! AN ST2 HAS BEEN GENERATED
7030  012346  012737  000002  001124            MOV     #2,$GDDAT       ;THE PRESET BUFFER SHOULD BE 2.
7031  012354  017737  167042  001126            MOV     @ABR,$BDDAT     ;READ THE PRESET BUFFER.
7032  012362  023737  001126  001124            CMP     $BDDAT,$GDDAT   ;DID A COUNTER TO PRESET BUFFER OCCUR?
7033  012370  001402                            BEQ     4$              ;YES - NEXT SUBTEST.
7034
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

7035  012372  104005                            ERROR   5               ;A COUNTER TO PRESET BUFFER DID NOT
7036                                                                    ;HAPPEN PROPERLY.
7037
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

7038  012374  000440                            BR      TST66           ;;
7039  012376  005037  001124            4$:     CLR     $GDDAT          ;EXPECT ZERO BACK FROM COUNT REG.
7040                                                                    ;/-RDCLK1-
 (1)  012402  017746  167012                    MOV     @ASR,-(SP)      ;/SAVE CSR CONTENTS.
 (1)  012406  052777  000005  167004            BIS     #5,@ASR         ;/SET TO MODE 2,GO
 (1)  012414  042777  100000  166776            BIC     #BIT15,@ASR     ;/CLR ST FLAG.
 (1)  012422  052777  001000  166770            BIS     #BIT9,@ASR      ;/GENERATE ST2 PULSE.
 (1)  012430  017737  166766  001126            MOV     @ABR,$BDDAT     ;/READ COUNT  'G.
 (1)  012436  052677  166756                    BIS     (SP)+,@ASR      ;/RESTORE CSR.
 (1)  012442  005737  001126                    TST     $BDDAT          ;/PREVIOUS CONTENTS OF COUNT REG
 (1)                                                                    ;/IN $BDDAT.
7041  012446  001402                            BEQ     5$              ;IF SO - NEXT TEST.
7042
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

7043  012450  104005                            ERROR   5               ;THE CLOCK FORGOT TO ZERO THE COUNT
7044                                                                    ;REG. AFTER AN ST2 OCCURRED ON
7045                                                                    ;A MODE 3 COUNT.
7046
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

7047  012452  000411                            BR      TST66           ;;
7048  012454                            5$:
7049  012454  005077  166740                    CLR     @ASR            ;NOW TRY CLEARING THE CSR.
7050  012460  017737  166734  001126            MOV     @ASR,$BDDAT     ;READ THE CSR - DID IT CLEAR?
7051  012466  001403                            BEQ     TST66           ;;
7052  012470  005037  001124                    CLR     $GDDAT          ;NO - RECORD S/B.
7053
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

7054  012474  104002                            ERROR   2               ;CSR FAILED TO CLEAR
7055
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

7056
```

```
 7065                                   ;;*********************************************************************
  (3)                                   ;*TEST 66       *TEST MODULE TEST OF OVERFLOW OUT ST2 IN AND OUT,AND ST1 IN
  (4)                                   ;*
  (4)                                   ;* IN THIS TEST WE WILL TEST OVERFLOW OUT,ST2 IN ST2 OUT, AND ST1
  (4)                                   ;* IN.   TO DO THIS WE CONNECTED THEM TOGETHER WITH THE TEST MODULE.
  (4)                                   ;*  NOTE: THIS TEST ONLY RAN IF TEST MODULE MODE SELECTED.
  (4)                                   ;*
  (3)                                   ;*********************************************************************
  (2) 012476 000004                     TST66:  SCOPE
  (1) 012500 012737 000010 001160               MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 7066 012506 105737 001476                      TSTB    DWARF           ;DWARF MODE
 7067 012512 001541                             BEQ     TST67           ;;BR IF NOT
 7068 012514 012737 012740 001110               MOV     #1$,$LPERR
 7069 012522 012737 012740 001106               MOV     #1$,$LPADR
 7070 012530 104401 012536                      TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 012534 000424                             BR      64$             ;;GET OVER THE ASCIZ
  (1)                                   ;;65$:  .ASCIZ  <200><7>'DWARF: ALL SWITCHES OFF, S2-1,S2-4 ON''
  (1) 012606                            64$:
 7071 012606 104401 012614                      TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1) 012612 000422                             BR      66$             ;;GET OVER THE ASCIZ
  (1)                                   ;;67$:  .ASCIZ  <200>'PANEL: ST1 IN (TTL), SLOPE IN (+).''
  (1) 012660                            66$:
 7072 012660 104401 012666                      TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1) 012664 000423                             BR      68$             ;;GET OVER THE ASCIZ
  (1)                                   ;;69$:  .ASCIZ  <200>''       ST2 IN (TTL), SLOPE IN (+).''<7>
  (1) 012734                            68$:
 7073 012734 004737 015166                      JSR PC,ANY2
 7074 012740 005077 166454              1$:     CLR     @ASR            ;CLEAR CSR
 7075 012744 012777 177777 166450               MOV     #-1,@ABR        ;PRESET FOR OVERFLOW.
 7076 012752 012777 000011 166440               MOV     #11,@ASR        ;SET 1 MHZ,GO.
 7077 012760 000240                             NOP
 7078 012762 000240                             NOP
 7079 012764 000240                             NOP
 7080 012766 000240                             NOP                     ;ALLOW TIME FOR OVERFLOW.
 7081 012770 012737 102210 001124               MOV     #102210,$GDDAT  ;EXPECT ST1 AND ST2 FLAGS TO BE SET.
 7082 012776 017737 166416 001126               MOV     @ASR,$BDDAT     ;READ CSR
 7083 013004 023737 001124 001126               CMP     $GDDAT,$BDDAT   ;OK?
 7084 013012 001401                             BEQ     TST67           ;;
 7085
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 7086 013014 104001                             ERROR   1               ;ST1 AND/OR ST2 FLAG FAILED TO SET
 7087
                                        ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 7088
```

```
 7096                                          ;;*****************************************************************
  (3)                                          ;*TEST 67       *DWARF TEST OF OVERFLOW OUT,ST1 IN AND OUT,AND ST2 IN.
  (4)                                          ;*
  (4)                                          ;*       IN THIS TEST,WE'LL TEST OVERFLOW OUT,ST1 IN,ST1 OUT, AND
  (4)                                          ;*       ST2 IN.
  (4)                                          ;*
  (3)                                          ;*****************************************************************
  (2)  013016  000004                          TST67:  SCOPE
  (1)  013020  012737  000010  001160                  MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 7097  013026  105737  001476                          TSTB    DWARF           ;DWARF TEST
 7098  013032  001510                                  BEQ     TST70           ;;
 7099  013034  012737  013200  001106                  MOV     #1$,$LPADR
 7100  013042  012737  013200  001110                  MOV     #1$,$LPERR
 7101  013050  104401  013056                          TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  013054  000426                                  BR      64$             ;;GET OVER THE ASCIZ
  (1)                                          ;;65$:  .ASCIZ  <200><7>'DWARF: ALL SWITCHES OFF, S2-2 AND S2-3 ON''
  (1)  013132                                  64$:
 7102  013132  104401  013140                          TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  013136  000416                                  BR      66$             ;;GET OVER THE ASCIZ
  (1)                                          ;;67$:  .ASCIZ  <200>'PANEL: SAME AS LAST TEST'<7>
  (1)  013174                                  66$:
 7103  013174  004737  015166                          JSR PC,ANY2
 7104  013200  005077  166214                  1$:     CLR     @ASR
 7105  013204  012777  177777  166210                  MOV     #-1,@ABR                ;PRESET FOR OVERFLOW.
 7106  013212  012777  000011  166200                  MOV     #11,@ASR                ;SET 1MHZ,GO.
 7107  013220  000240                                  NOP
 7108  013222  000240                                  NOP
 7109  013224  000240                                  NOP
 7110  013226  012737  102210  001124                  MOV     #102210,$GDDAT          ;EXPECT ST1,ST2 FLAGS SET.
 7111  013234  017737  166160  001126                  MOV     @ASR,$BDDAT             ;READ CSR.
 7112  013242  023737  001124  001126                  CMP     $GDDAT,$BDDAT
 7113  013250  001401                                  BEQ     TST70           ;;
 7114
                                              ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 7115  013252  104001                                  ERROR 1                         ;ST1 AND/OR ST2 FLAG(S) FAIL TO SET.
 7116
                                              ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
 7118                                   ;;*****************************************************************
  (3)                                   ;*TEST 70       *IF ENABLED,CHECK THRESHOLD ST1 FROM TESTOR
  (3)                                   ;;*****************************************************************
  (2) 013254 000004                     TST70:  SCOPE
 7119
 7120 013256 005737 001202              TST     $PASS           ;CHECK IF FIRST PASS
 7121 013262 001003                     BNE     1$              ;BR IF NOT
 7122 013264 005737 001476              TST     DWARF           ;OPERATING IN TESTOR/DWARF MODE?
 7123 013270 001002                     BNE     2$              ;YES DO THIS TEST.
 7124 013272 000137 014512      1$:     JMP     ENDP            ;NO-END PASS
 7125 013276 100450             2$:     BMI     4$              ;BR IF TESTER
 7126 013300 005737 001500              TST     ASK             ;QUESTION       ALLREADY BEEN ASKED?
 7127 013304 001032                     BNE     3$
 7128 013306 104401 013314              TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 013312 000421                     BR      64$             ;;GET OVER THE ASCIZ
  (1)                           ;;65$:  .ASCIZ  <200>#15 VOLT SUPPLY TO DWARF?(Y OR N)#
  (1) 013356                    64$:
 7129 013356 104411                     RDCHR
 7130 013360 012637 001500              MOV     (SP)+,ASK
 7131 013364 042737 000240 001500       BIC     #240,ASK        ;STRIP PARITY AND LOWER CASE.
 7132 013372 123727 001500 000131 3$:   CMPB    ASK,#'Y         ;DID HE ANSWER YES?
 7133 013400 001407                     BEQ     4$
 7134 013402 123727 001500 000116       CMPB    ASK,#'N
 7135 013410 001730                     BEQ     1$
 7136 013412 005037 001500              CLR     ASK
 7137 013416 000727                     BR      2$
 7138 013420                    4$:
  (1) 013420 104401 013426              TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1) 013424 000425                     BR      66$             ;;GET OVER THE ASCIZ
  (1)                           ;;67$:  .ASCIZ  <200>#PANEL: ST1 AND ST2 POTS OUT AND TURN CCW#
  (1) 013500                    66$:
 7139 013500 104401 013506              TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1) 013504 000425                     BR      68$             ;;GET OVER THE ASCIZ
  (1)                           ;;69$:  .ASCIZ  <200>#DWARF: S2-7 AND S2-8 ON, ALL OTHERS OFF#
  (1) 013560                    68$:
 7140 013560 005077 165634              CLR     @ASR            ;CLEAR CSR
 7141 013564 004737 015166              JSR PC,ANY2
 7142 013570 005077 165624              CLR     @ASR
 7143 013574 004737 015120              JSR PC,ANYKEY
 7144 013600 017737 165614 001126       MOV     @ASR,$BDDAT     ;READ CSR
 7145 013606 012737 000000 001124       MOV     #0,$GDDAT
 7146 013614 032737 102000 001126       BIT     #BIT15!BIT10,$BDDAT     ;DID ANY FLAG SET?
 7147 013622 001401                     BEQ     TST71           ;;
 7148 013624 104002                     ERROR 2                 ;ST1 OR ST2 THRESHOLD LEVEL ERROR
 7149                                                           ;FLAGS SHOULD NOT HAVE SET!
```

```
 7151                                   ;;*****************************************************************
  (3)                                   ;*TEST 71        *ST1,ST2 THRESHOLD TEST #2,POTS CW
  (3)                                   ;;*****************************************************************
  (2)  013626  000004                   TST71:  SCOPE
 7152  013630  012737  013704  001110           MOV     #1$,$LPERR
 7153  013636  012737  013704  001106           MOV     #1$,$LPADR
 7154
 7155  013644  104401  013652                   TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  013650  000415                           BR      64$             ;;GET OVER THE ASCIZ
  (1)                                   ;;65$:  .ASCIZ  <200>#PANEL: TURN BOTH POTS CW#
  (1)  013704                           64$:
 7156  013704  005077  165510           1$:     CLR     @ASR
 7157  013710  004737  015166                   JSR PC,ANY2
 7158  013714  005077  165500                   CLR     @ASR
 7159  013720  004737  015120                   JSR PC,ANYKEY
 7160  013724  017737  165470  001126           MOV     @ASR,$BDDAT
 7161  013732  032737  102000  001126           BIT     #BIT15!BIT10,$BDDAT     ;DID ANY FLAG SET?
 7162  013740  001401                           BEQ     TST72           ;;
 7163  013742  104002                           ERROR   2                       ;ST1 OR ST2 THRESHOLD ERROR.
 7164
 7165                                   ;;*****************************************************************
  (3)                                   ;*TEST 72        *ST1,ST2 THRESHOLD TEST #3 MID RANGE
  (3)                                   ;;*****************************************************************
  (2)  013744  000004                   TST72:  SCOPE
 7166  013746  012737  014044  001110           MOV     #1$,$LPERR
 7167  013754  012737  014044  001106           MOV     #1$,$LPADR
 7168  013762  104401  013770                   TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  013766  000426                           BR      64$             ;;GET OVER THE ASCIZ
  (1)                                   ;;65$:  .ASCIZ  <200>#PANEL: SET ST1 AND ST2 POTS TO MID-RANGE.#
  (1)  014044                           64$:
 7169  014044  005077  165350           1$:     CLR     @ASR
 7170  014050  004737  015166                   JSR PC,ANY2
 7171  014054  005077  165340                   CLR     @ASR
 7172  014060  004737  015120                   JSR PC,ANYKEY
 7173  014064  017737  165330  001126           MOV     @ASR,$BDDAT
 7174  014072  012737  102000  001124           MOV     #BIT15!BIT10,$GDDAT
 7175  014100  042737  075777  001126           BIC     #075777,$BDDAT
 7176  014106  023737  001124  001126           CMP     $GDDAT,$BDDAT           ;AT MID RANGE THEY BOTH SHOLD SET.
 7177  014114  001401                           BEQ     TST73           ;;
 7178  014116  104002                           ERROR   2                       ;ST1 OR ST2 FAILED TO SET.
```

```
 7180                                          ;;*****************************************************************
  (3)                                          ;*TEST 73       *TEST CLOCK REPEATABILITY IF ON TESTOR
  (3)                                          ;;*****************************************************************
  (2) 014120 000004                            TST73:  SCOPE
  (1) 014122 012737 000010 001160                      MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 7181 014130 005737 001476                             TST     DWARF           ;TESTOR MODE ENABLED??
 7182 014134 100402                                    BMI     10$             ;BR IF YES
 7183 014136 000137 014512                             JMP     ENDP            ;NO REPORT END PASS.
 7184 014142 012737 014354 001110 10$:       MOV     #1$,$LPERR
 7185 014150 012737 014354 001106            MOV     #1$,$LPADR
 7186 014156 104401 014164                   TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1) 014162 000416                          BR      64$             ;;GET OVER THE ASCIZ
  (1)                                 ;;65$:  .ASCIZ  <200>#PANEL: ST1 POT OUT AND C#
  (1) 014220                          64$:
 7187 014220 104401 014226                   TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1) 014224 000423                          BR      66$             ;;GET OVER THE ASCIZ
  (1)                                 ;;67$:  .ASCIZ  <200>#       ST2 POT IN AND SLOPE OUT (-)#
  (1) 014274                          66$:
 7188 014274 104401 014302                   TYPE    ,69$            ;;TYPE ASCIZ STRING
  (1) 014300 000423                          BR      68$             ;;GET OVER THE ASCIZ
  (1)                                 ;;69$:  .ASCIZ  <200>#DWARF: S2 ALL SWITCHES OFF, S2-6 C#
  (1) 014350                          68$:
 7189 014350 004737 015166                   JSR PC,ANY2
 7190 014354 012777 020016 165036 1$:        MOV     #BIT13!16,@ASR  ;SET 1MHZ,MODE 3,ST2 GO ENABLE. TEST CLOCK
 7191 014362 005077 165054                   CLR     @TSCLC          ;CLEAR STATUS REG.
 7192 014366 012777 100000 165050            MOV     #100000,@TSCLD  ;PRESET COUNT REG.
 7193 014374 012777 000013 165040            MOV     #13,@TSCLC      ;SET 1MHZ,MODE 1,GO
 7194 014402 105777 165034           2$:     TSTB    @TSCLC          ;WAIT FOR CLOCK OVERFLOW.
 7195 014406 100375                          BPL     2$
 7196 014410 042777 100000 165002            BIC     #BIT15,@ASR
 7197 014416 042777 000200 165016            BIC     #200,@TSCLC     ;CLEAR OVERFLOW FLAG.
 7198 014424 105777 165012           3$:     TSTB    @TSCLC          ;WAIT FOR NEXT OVERFLOW.
 7199 014430 100375                          BPL     3$
 7200 014432 005077 164762                   CLR     @ASR            ;STOP CLOCK.
 7201 014436 005077 165000                   CLR     @TSCLC
 7202 014442 017737 164754 001126            MOV     @ABR,$BDDAT     ;READ RESULTS
 7203 014450 012737 100000 001124            MOV     #100000,$GDDAT  ;S/B COUNT
 7204 014456 013700 001124                   MOV     $GDDAT,R0
 7205 014462 163700 001126                   SUB     $BDDAT,R0
 7206 014466 100001                          BPL     4$              ;+DIF.
 7207 014470 005100                          COM     R0              ;OTHERWIZE MAKE IT
 7208 014472 020027 000007           4$:     CMP     R0,#7           ;SHOULD NOT VARY MORE THAN 7 COUNTS.
 7209 014476 003401                          BLE     TST74           ;;
 7210
 7211 014500 104010                          ERROR   10              ;CLOCK REPEATABLITY ERROR.
```

```
 7213                                   ;;*********************************************************************
  (3)                                   ;*TEST 74              END OF TESTS
  (3)                                   ;*********************************************************************
  (2)  014502  000004                   TST74:  SCOPE
  (1)  014504  012737  000001  001160            MOV     #1,$TIMES       ;;DO 1 ITERATION
 7214                                           ; WE'LL FIND OUT IF THERE ARE OTHER CLOCKS OUT THERE TO TEST.
 7215                                           ;
 7216  014512                           ENDP:
 7217  014512  000005                           RESET
 7218  014514  052777  000100  164422          BIS     #BIT6,@$TKS     ;ENABLE TKB INTR.
 7219
 7220  014522  005737  001476                  TST     DWARF           ;CHECK TESTER/DWARF INDICATOR
 7221  014526  001407                          BEQ     5$              ;BR IF NOT
 7222  014530  005737  001202                  TST     $PASS           ;CHECK IF FIRST PASS
 7223  014534  001004                          BNE     5$              ;BR IF NOT
 7224  014536  104401  023056                  TYPE,   PRIME0          ;TELL OPER. TO RESET POTS/SWITCHES
 7225  014542  004737  015166                  JSR     PC,ANY2         ;WAIT FOR ANY CHAR
 7226  014546  005237  001206          5$:     INC     $UNIT           ;UPDATE # OF UNITS COUNTER
 7227  014552  123737  001206  001460          CMPB    $UNIT,EVER      ;FIND IF DONE
 7228  014560  001416                          BEQ     $EOP            ;;BR IF FINISHED
 7229  014562  063737  001454  001420          ADD     VADDR,ASR       ;UPDATE THE ADDRESSES
 7230  014570  063737  001456  001424          ADD     VVECTR,VECT1    ;UPDATE THE VECTOR
 7231  014576  004737  003016                  JSR     PC,FIXADR       ;FIX THE OTHER VECTORS AND ADDR.
 7232  014602  005037  001102                  CLR     $TSTNM          ;INIT THE TEST #
 7233  014606  006337  001450                  ASL     MASKNM          ;ROTATE THE UNIT INDICATOR
 7234  014612  000137  003106                  JMP     TST1            ;AND RUN THAT UNIT
 7235
 7236                                   .SBTTL  END OF PASS ROUTINE
  (1)
  (2)                                   ;;*********************************************************************
  (1)                                   ;*INCREMENT THE PASS NUMBER ($PASS)
  (1)                                   ;*TYPE 'END PASS #XXXXX'' (WHERE XXXXX IS A DECIMAL NUMBER)
  (1)                                   ;*IF THERES A MONITOR GO TO IT
  (1)                                   ;*IF THERE ISN'T JUMP TO EXTMSG
  (1)
  (1)  014616                           $EOP:
  (2)  014616  000240                           NOP
  (1)  014620  005037  001102                   CLR     $TSTNM          ;;ZERO THE TEST NUMBER
  (1)  014624  005037  001160                   CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
  (1)  014630  005237  001202                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
  (1)  014634  042737  100000  001202          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
  (1)  014642  005327                           DEC     (PC)+           ;;LOOP?
  (1)  014644  000001                   $EOPCT: .WORD   1
  (1)  014646  003022                           BGT     $DOAGN          ;;YES
  (1)  014650  012737                           MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
  (1)  014652  000001                   $ENDCT: .WORD   1
  (1)  014654  014644                           $EOPCT
  (1)  014656  104401  014723                   TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
  (2)  014662  013746  001202                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
  (2)  014666  104405                           TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
  (1)  014670  104401  014720                   TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
  (1)  014674  013700  000042           $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
  (1)  014700  001405                           BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
  (1)  014702  000005                           RESET                   ;;CLEAR THE WORLD
  (1)  014704  004710                   $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
  (1)  014706  000240                           NOP                     ;;SAVE ROOM
```

```
  (1)  014710  000240                          NOP                         ;;FOR
  (1)  014712  000240                          NOP                         ;;ACT11
  (1)  014714                          $DOAGN:
  (1)  014714  000137                          JMP     @(PC)+              ;;RETURN
  (1)  014716  014740                  $RTNAD: .WORD   EXTMSG
  (1)  014720     377      377     000 $ENULL: .BYTE   -1,-1,0             ;;NULL CHARACTER STRING
  (1)  014723     015   042412  042116 $ENDMG: .ASCIZ  <15><12>/END PASS #/
  (1)  014730  050040  051501  020123
  (1)  014736  000043
 7237
 7238  014740  052777  000100  164176 EXTMSG: BIS     #BIT6,@$TKS         ;ENABLE TKB INTR.
 7239  014746  005737  001112                  TST     $ERTTL              ;TEST IF ANY ERRORS
 7240  014752  001416                          BEQ     1$                  ;BR IF NONE
 7241  014754  104401  024027                  TYPE    ,ERRTOT
 7242  014760  013746  001112                  MOV     $ERTTL,-(SP)
 7243  014764  104405                          TYPDS
 7244  014766  022737  000001  001450          CMP     #1,MASKNM           ;TEST IF ADDITIONAL UNITS
 7245  014774  001405                          BEQ     1$                  ;BR IF NOT
 7246  014776  104401  024056                  TYPE,   MESGD               ;INFORM OPER. OF BAD UNITS
 7247  015002  013746  001452                  MOV     BADUNT,-(SP)
 7248  015006  104406                          TYPBN                       ;TYPE BIN MAP
 7249  015010  104401  014720          1$:     TYPE,   $ENULL              ;ENSURE ALL TEXT GOT TYPED
 7250  015014  004737  015260                  JSR     PC,CTRLCG           ;TEST FOR CTRL C/G
 7251  015020  000137  002464                  JMP     LOGIC
 7252
 7253                                ;SUBROUTINE TO CHANGE BASE OR VECTOR ADDRESS
 7254
 7255  015024  104401  023655          BASEXC: TYPE,   ADROUT              ;TYPE OUT STARTING
 7256  015030  013746  001250                  MOV     $BASE,-(SP)         ;GET CURRENT DEFAULT
 7257  015034  104402                          TYPOC                       ;TELL OPER.
 7258  015036  104401  023753                  TYPE,   ENDOUT              ;ADD END TEXT
 7259  015042  104413                          RDOCT                       ;GET INPUT
 7260  015044  005726                          TST     (SP)+               ;TEST INPUT
 7261  015046  001403                          BEQ     1$                  ;BR IF NONE
 7262  015050  016637  177776  001250          MOV     -2(SP),$BASE        ;GET OPER. INPUT
 7263  015056  104401  023713          1$      TYPE,   VECOUT              ;TYPE OUT STARTING VECTOR
 7264  015062  013746  001244                  MOV     $VECT1,-(SP)        ;GET CURRENT DEFAULT
 7265  015066  104402                          TYPOC                       ;TELL OPER.
 7266  015070  104401  023753                  TYPE,   ENDOUT              ;ADD END TEXT
 7267  015074  104413                          RDOCT                       ;GET INPUT
 7268  015076  005726                          TST     (SP)+               ;TEST INPUT
 7269  015100  001403                          BEQ     2$                  ;BR IF NONE
 7270  015102  016637  177776  001244          MOV     -2(SP),$VECT1       ;GET OPER. INPUT
 7271  015110  004737  003002          2$:     JSR     PC,PRIADR           ;PRIME ADD AND VECTOR
 7272  015114  000137  002270                  JMP     MTEST1              ;RETYPE DOT
```

```
 7274
 7275                                        ;
 7276                                        ;THIS ROUTINE TYPES LAST MESSAGE AND WAITS FOR AN  OPERATOR
 7277                                        ;RESPONCE.
 7278                                        ;
 7279
 7280  015120  105777  164022       ANYKEY: TSTB    @$TKB           ;CLEAR TTY READY FLAG.
 7281  015124  104401  015132               TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  015130  000416                       BR      64$             ;;GET OVER THE ASCIZ
  (1)                              ;;65$:   .ASCIZ  <200><7>#DWARF: SWITCH S1 3 TIMES#
  (1)  015166                      64$:
 7282  015166                      ANY2:
  (1)  015166  104401  015174               TYPE    ,65$            ;;TYPE ASCIZ STRING
  (1)  015172  000423                       BR      64$             ;;GET OVER THE ASCIZ
  (1)                              ;;65$:   .ASCIZ  <200><7>#DEPRESS 'RETURN'' KEY WHEN DONE...#<7>
  (1)  015242                      64$:
 7283  015242  104411                       RDCHR                   ;GET 'RETURN''
 7284  015244  005726                       TST     (SP)+
 7285  015246  104401  015254               TYPE    ,67$            ;;TYPE ASCIZ STRING
  (1)  015252  000401                       BR      66$             ;;GET OVER THE ASCIZ
  (1)                              ;;67$:   .ASCIZ  <200>##
  (1)  015256                      66$:
 7286  015256  000207                       RTS PC
 7287                             ;SUBROUTINE TO TRAP CTRL C/G
 7288  015260  000207             CTRLCG: RTS     PC
 7289  015262  105777  163656               TSTB    @$TKS           ;TEST IF INPUT
 7290  015266  100022                       BPL     2$              ;BR IF NONE
 7291  015270  017737  163652  015336       MOV     @$TKB,CTRCHA    ;READ CHAR.
 7292  015276  042737  177640  015336       BIC     #177640,CTRCHA  ;MASK OFF BITS
 7293  015304  022737  000003  015336       CMP     #3,CTRCHA       ;TEST FOR CTRL C
 7294  015312  001003                       BNE     1$              ;BR IF NOT
 7295  015314  005726                       TST     (SP)+           ;CLEAN STACK
 7296  015316  000137  002270               JMP     MTEST1          ;RETYPE DOT
 7297  015322  022737  000007  015336 1$:   CMP     #7,CTRCHA       ;TEST FOR CTRL G
 7298  015330  001001                       BNE     2$              ;BR IF NOT
 7299  015332  104407                       GTSWR                   ;GET NEW SWITCHS
 7300  015334  000207                 2$:   RTS     PC              ;EXIT
 7301  015336  000000             CTRCHA: 0               ;CHAR. THE OPER TYPED DURING RUNNING
 7302
```

```
 7304
 7305                                           .SBTTL
 7306                                           .SBTTL  *SYSMAC ROUTINES
 7307                                           .SBTTL
 7308
 7309                                   .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
  (1)
  (2)                                  ;;*********************************************************************
  (1)                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
  (1)                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
  (1)                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
  (1)                                  ;*CALL:
  (1)                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
  (1)                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
  (1)                                  ;*      .BYTE   M               ;;M=1 OR 0
  (1)                                  ;*                                      ;;1=TYPE LEADING ZEROS
  (1)                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
  (1)                                  ;*
  (1)                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
  (1)                                  ;*$TYPOS OR $TYPOC
  (1)                                  ;*CALL:
  (1)                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
  (1)                                  ;*
  (1)                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
  (1)                                  ;*CALL:
  (1)                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
  (1)                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
  (1)
  (1)  015340  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
  (1)  015344  116637  000001  015563          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
  (1)  015352  112637  015565                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
  (1)  015356  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
  (1)  015362  000406                          BR      $TYPON
  (1)  015364  112737  000001  015563  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
  (1)  015372  112737  000006  015565          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
  (1)  015400  112737  000005  015562  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
  (1)  015406  010346                          MOV     R3,-(SP)        ;;SAVE R3
  (1)  015410  010446                          MOV     R4,-(SP)        ;;SAVE R4
  (1)  015412  010546                          MOV     R5,-(SP)        ;;SAVE R5
  (1)  015414  113704  015565                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
  (1)  015420  005404                          NEG     R4
  (1)  015422  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
  (1)  015426  110437  015564                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
  (1)  015432  113704  015563                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
  (1)  015436  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
  (1)  015442  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
  (1)  015444  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
  (1)  015446  000404                          BR      3$              ;;GO DO MSB
  (1)  015450  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
  (1)  015452  006105                          ROL     R5
  (1)  015454  006105                          ROL     R5
  (1)  015456  010503                          MOV     R5,R3
  (1)  015460  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
  (1)  015462  105337  015564                  DECB    $OMODE          ;;TYPE THIS DIGIT?
```

```
(1)  015466  100016                        BPL     7$              ;;BR IF NO
(1)  015470  042703  177770                BIC     #177770,R3      ;;GET RID OF JUNK
(1)  015474  001002                        BNE     4$              ;;TEST FOR 0
(1)  015476  005704                        TST     R4              ;;SUPPRESS THIS 0?
(1)  015500  001403                        BEQ     5$              ;;BR IF YES
(1)  015502  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  015504  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  015510  052703  000040        5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  015514  110337  015560                MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  015520  104401  015560                TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  015524  105337  015562        7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  015530  003347                        BGT     2$              ;;BR IF MORE TO DO
(1)  015532  002402                        BLT     6$              ;;BR IF DONE
(1)  015534  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  015536  000744                        BR      2$              ;;GO DO THE LAST DIGIT
(1)  015540  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  015542  012604                        MOV     (SP)+,R4        ;;RESTORE R4
(1)  015544  012603                        MOV     (SP)+,R3        ;;RESTORE R3
(1)  015546  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  015554  012616                        MOV     (SP)+,(SP)
(1)  015556  000002                        RTI                     ;;RETURN
(1)  015560     000                8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  015561     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  015562     000                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  015563     000                $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  015564  000000                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
7310                                .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
(1)
(2)                                 ;;**********************************************************************
(1)                                 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
(1)                                 ;*BINARY-ASCII NUMBER AND TYPE IT.
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
(1)                                 ;*      TYPBN                   ;;TYPE IT
(1)
(1)  015566  010146                $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
(1)  015570  016601  000006                MOV     6(SP),R1        ;;GET THE INPUT NUMBER
(1)  015574  000261                        SEC                     ;;SET ''C'' SO CAN KEEP TRACK OF THE NUMBER OF BITS
(1)  015576  112737  000060  015640  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII ''0''.
(1)  015604  006101                        ROL     R1              ;;GET THIS BIT
(1)  015606  001406                        BEQ     2$              ;;DONE?
(1)  015610  105537  015640                ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
(1)  015614  104401  015640                TYPE    ,$BIN           ;;GO TYPE THIS BIT
(1)  015620  000241                        CLC                     ;;CLEAR ''C'' SO CAN KEEP TRACK OF BITS
(1)  015622  000765                        BR      1$              ;;GO DO THE NEXT BIT
(1)  015624  012601                2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
(1)  015626  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  015634  012616                        MOV     (SP)+,(SP)
(1)  015636  000002                        RTI                     ;;RETURN TO USER
(1)  015640     000     000        $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
```

```
 7312                                   .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
  (1)                                   ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
  (1)                                   ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
  (1)                                   ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
  (1)                                   ;*REPLACED WITH SPACES.
  (1)                                   ;*CALL:
  (1)                                   ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
  (1)                                   ;*      TYPDS                   ;;GO TO THE ROUTINE
  (1)
  (1)  015642                           $TYPDS:
  (3)  015642  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (3)  015644  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (3)  015646  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (3)  015650  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (3)  015652  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
  (1)  015654  012746  020200                   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
  (1)  015660  016605  000020                   MOV     20(SP),R5       ;;GET THE INPUT NUMBER
  (1)  015664  100004                           BPL     1$              ;;BR IF INPUT IS POS.
  (1)  015666  005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
  (1)  015670  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
  (1)  015676  005000                   1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
  (1)  015700  012703  016056                   MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
  (1)  015704  112723  000040                   MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
  (1)  015710  005002                   2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
  (1)  015712  016001  016046                   MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
  (1)  015716  160105                   3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
  (1)  015720  002402                           BLT     4$              ;;BR IF DONE
  (1)  015722  005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
  (1)  015724  000774                           BR      3$
  (1)  015726  060105                   4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
  (1)  015730  005702                           TST     R2              ;;CHECK IF BCD DIGIT=0
  (1)  015732  001002                           BNE     5$              ;;FALL THROUGH IF 0
  (1)  015734  105716                           TSTB    (SP)            ;;STILL DOING LEADING 0'S?
  (1)  015736  100407                           BMI     7$              ;;BR IF YES
  (1)  015740  106316                   5$:     ASLB    (SP)            ;;MSD?
  (1)  015742  103003                           BCC     6$              ;;BR IF NO
  (1)  015744  116663  000001  177777           MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
  (1)  015752  052702  000060           6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
  (1)  015756  052702  000040           7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
  (1)  015762  110223                           MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
  (1)  015764  005720                           TST     (R0)+           ;;JUST INCREMENTING
  (1)  015766  020027  000010                   CMP     R0,#10          ;;CHECK THE TABLE INDEX
  (1)  015772  002746                           BLT     2$              ;;GO DO THE NEXT DIGIT
  (1)  015774  003002                           BGT     8$              ;;GO TO EXIT
  (1)  015776  010502                           MOV     R5,R2           ;;GET THE LSD
  (1)  016000  000764                           BR      6$              ;;GO CHANGE TO ASCII
  (1)  016002  105726                   8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
  (1)  016004  100003                           BPL     9$              ;;BR IF NO
  (1)  016006  116663  177777  177776           MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
  (1)  016014  105013                   9$:     CLRB    (R3)            ;;SET THE TERMINATOR
  (3)  016016  012605                           MOV     (SP)+,R5        ;;POP STACK INTO R5
  (3)  016020  012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
  (3)  016022  012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
```

```
   (3)  016024  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
   (3)  016026  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
   (1)  016030  104401  016056                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
   (1)  016034  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
   (1)  016042  012616                          MOV     (SP)+,(SP)
   (1)  016044  000002                          RTI                     ;;RETURN TO USER
   (1)  016046  023420                $DTBL:    10000.
   (1)  016050  001750                          1000.
   (1)  016052  000144                          100.
   (1)  016054  000012                          10.
   (1)  016056  000004                $DBLK:    .BLKW   4
  7313
```

```
 7327                                   .SBTTL  ERROR HANDLER ROUTINE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
  (1)                                   ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
  (1)                                   ;*AND GO TO $ERRTYP ON ERROR
  (1)                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
  (1)                                   ;*SW15=1        HALT ON ERROR
  (1)                                   ;*SW13=1        INHIBIT ERROR TYPEOUTS
  (1)                                   ;*SW10=1        BELL ON ERROR
  (1)                                   ;*SW09=1        LOOP ON ERROR
  (1)                                   ;*CALL
  (1)                                   ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
  (1)
  (1)  016066                           $ERROR:
  (1)  016066  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
  (3)  016070  004737  015260                   JSR     PC,CTRLCG       ;TEST FOR CTRL C/G
  (3)  016074  053737  001450  001452           BIS     MASKNM,BADUNT           ;SET THE FAILING UNIT
  (3)  016102  013737  001450  016132           MOV     MASKNM,11$              ;SAVE IT
  (3)  016110  012737  000000  001504           MOV     #0,UNITBD               ;PRIME THE UNIT #
  (3)  016116  006237  016132           10$:    ASR     11$                     ;SHIFT LEFT
  (3)  016122  001404                           BEQ     12$                     ;BR IF FINISHED
  (3)  016124  005237  001504                   INC     UNITBD                  ;UPDATE BAD UNIT VALUE
  (3)  016130  000772                           BR      10$                     ;BR AGAIN
  (3)  016132  000000                   11$:    0
  (3)  016134  000240                   12$:    NOP
  (1)  016136  105237  001103           7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
  (1)  016142  001775                           BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
  (1)  016144  013777  001102  162770           MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
  (1)  016152  032777  002000  162760           BIT     #BIT10,@SWR     ;;BELL ON ERROR?
  (1)  016160  001402                           BEQ     1$              ;;NO - SKIP
  (1)  016162  104401  001164                   TYPE    ,$BELL          ;;RING BELL
  (1)  016166  005237  001112           1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
  (1)  016172  011637  001116                   MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
  (1)  016176  162737  000002  001116           SUB     #2,$ERRPC
  (1)  016204  117737  162706  001114           MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
  (1)  016212  032777  020000  162720           BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
  (1)  016220  001004                           BNE     20$             ;;SKIP TYPEOUTS
  (1)  016222  004737  016322                   JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
  (1)  016226  104401  001171                   TYPE    ,$CRLF
  (1)  016232                           20$:
  (1)  016232  122737  000001  001214           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)  016240  001007                           BNE     2$              ;;NO,SKIP APT ERROR REPORT
  (1)  016242  113737  001114  016254           MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
  (1)  016250  004737  020710                   JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
  (1)  016254     000                   21$:    .BYTE   0
  (1)  016255     000                           .BYTE   0
  (1)  016256  000777                   22$:    BR      22$             ;;APT ERROR LOOP
  (1)  016260  005777  162654           2$:     TST     @SWR            ;;HALT ON ERROR
  (1)  016264  100002                           BPL     3$              ;;SKIP IF CONTINUE
  (1)  016266  000000                           HALT                    ;;HALT ON ERROR!
  (1)  016270  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
  (1)  016272  032777  001000  162640   3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
  (1)  016300  001402                           BEQ     4$              ;;BR IF NO
  (1)  016302  013716  001110                   MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
  (1)  016306  005737  001162           4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
```

```
(1)  016312  001402                        BEQ     5$              ;;BR IF NONE
(1)  016314  013716  001162                MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  016320                          5$:
(1)  016320  000002                        RTI                     ;;RETURN
7328                                 .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1)                                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1)                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  016322                          $ERRTYP:
(1)  016322  104401  001171                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016326  010046                        MOV     R0,-(SP)        ;;SAVE R0
(1)  016330  005000                        CLR     R0              ;;PICKUP THE ITEM INDEX
(1)  016332  153700  001114                BISB    @#$ITEMB,R0
(1)  016336  001004                        BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)                                                                ;;TYPE THE PC OF THE ERROR
(2)  016340  013746  001116                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                                ;;ERROR ADDRESS
(2)  016344  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016346  000445                        BR      10$             ;;GET OUT
(1)  016350  005300                  1$:   DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
(1)  016352  006300                        ASL     R0              ;;      WORK FOR THE ERROR TABLE
(1)  016354  006300                        ASL     R0
(1)  016356  006300                        ASL     R0
(1)  016360  062700  001256                ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
(1)  016364  012037  016374                MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
(1)  016370  001404                        BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
(1)  016372  104401                        TYPE                    ;;TYPE THE "ERROR MESSAGE"
(1)  016374  000000                  2$:   .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
(1)  016376  104401  001171                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016402  012037  016412          3$:   MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
(1)  016406  001404                        BEQ     5$              ;;SKIP TYPEOUT IF 0
(1)  016410  104401                        TYPE                    ;;TYPE THE "DATA HEADER"
(1)  016412  000000                  4$:   .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
(1)  016414  104401  001171                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1)  016420  010146                  5$:   MOV     R1,-(SP)        ;;SAVE R1
(1)  016422  012001                        MOV     (R0)+,R1        ;;PICKUP "DATA TABLE" POINTER
(1)  016424  001415                        BEQ     9$              ;;BR IF NO DATA TO BE TYPED
(1)  016426  012000                        MOV     (R0)+,R0        ;;PICKUP "DATA FORMAT" POINTER
(1)  016430  105720                  6$:   TSTB    (R0)+           ;;"OCTAL" OR "DECIMAL"
(1)  016432  001003                        BNE     7$              ;;BR IF DECIMAL
(2)  016434  013146                        MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(2)  016436  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  016440  000402                        BR      8$
(1)  016442                          7$:
(2)  016442  013146                        MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(2)  016444  104405                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1)  016446  005711                  8$:   TST     (R1)            ;;IS THERE ANOTHER NUMBER?
(1)  016450  001403                        BEQ     9$              ;;BR IF NO
(1)  016452  104401  016472                TYPE    ,11$            ;;TYPE TWO(2) SPACES
(1)  016456  000764                        BR      6$              ;;LOOP
(1)
(1)  016460  012601                  9$:   MOV     (SP)+,R1        ;;RESTORE R1
(1)  016462  012600                  10$:  MOV     (SP)+,R0        ;;RESTORE R0
```

```
(1)  016464  104401  001171            TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
(1)  016470  000207                    RTS     PC              ;;RETURN
(1)  016472  020040     000    11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
(1)          016476                    .EVEN
7329                           .SBTTL  SCOPE HANDLER ROUTINE
(1)
(2)                            ;;*******************************************************************
(1)                            ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1)                            ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1)                            ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1)                            ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                            ;*SW14=1        LOOP ON TEST
(1)                            ;*SW11=1        INHIBIT ITERATIONS
(1)                            ;*SW09=1        LOOP ON ERROR
(1)                            ;*SW08=1        LOOP ON TEST IN SWR<7:0>
(1)                            ;*CALL
(1)                            ;*      SCOPE           ;;SCOPE=IOT
(1)
(1)  016476                    $SCOPE:
(1)  016476  104410                    CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
(2)  016500  004737  015260            JSR     PC,CTRLCG       ;TEST FOR CTRL C/G
(1)  016504  032777  040000  162426 1$:  BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
(1)  016512  001114                    BNE     $OVER           ;;YES IF SW14=1
(1)                            ;#####START OF CODE FOR THE XOR TESTER#####
(1)  016514  000416            $XTSTR: BR      6$              ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
(1)                                                            ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
(1)  016516  013746  000004            MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1)  016522  012737  016542  000004    MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
(1)  016530  005737  177060            TST     @#177060        ;;TIME OUT ON XOR?
(1)  016534  012637  000004            MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  016540  000463                    BR      $SVLAD          ;;GO TO THE NEXT TEST
(1)  016542  022626            5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
(1)  016544  012637  000004            MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
(1)  016550  000423                    BR      7$              ;;LOOP ON THE PRESENT TEST
(1)  016552                    6$:;#####END OF CODE FOR THE XOR TESTER#####
(1)  016552  032777  000400  162360    BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
(1)  016560  001404                    BEQ     2$              ;;BR IF NO
(1)  016562  127737  162352  001102    CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
(1)  016570  001465                    BEQ     $OVER           ;;BR IF YES
(1)  016572  105737  001103    2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
(1)  016576  001421                    BEQ     3$              ;;BR IF NO
(1)  016600  123737  001115  001103    CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  016606  101015                    BHI     3$              ;;BR IF NO
(1)  016610  032777  001000  162322    BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
(1)  016616  001404                    BEQ     4$              ;;BR IF NO
(1)  016620  013737  001110  001106 7$:  MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  016626  000446                    BR      $OVER
(1)  016630  105037  001103    4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
(1)  016634  005037  001160            CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  016640  000415                    BR      1$              ;;ESCAPE TO THE NEXT TEST
(1)  016642  032777  004000  162270 3$:  BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
(1)  016650  001011                    BNE     1$              ;;BR IF YES
(1)  016652  005737  001202            TST     $PASS           ;;IF FIRST PASS OF PROGRAM
(1)  016656  001406                    BEQ     1$              ;;      INHIBIT ITERATIONS
(1)  016660  005237  001104            INC     $ICNT           ;;INCREMENT ITERATION COUNT
(1)  016664  023737  001160  001104    CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
```

```
(1)  016672  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
(1)  016674  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
(1)  016702  013737  016760  001160          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
(1)  016710  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
(1)  016714  113737  001102  001200          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1)  016722  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1)  016726  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  016732  005037  001162                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  016736  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  016744  013777  001102  162170  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  016752  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(1)  016756  000002                          RTI                     ;;FIXES PS
(1)  016760  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
```

```
  7331                                  .SBTTL  TTY INPUT ROUTINE
   (1)
   (2)                                  ;;*****************************************************************
   (1)                                  .ENABL  LSB
   (1)  016762  000000                  $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
   (1)  016764  000000                  $TKQIN: .WORD   0               ;;INPUT POINTER
   (1)  016766  000000                  $TKQOUT: .WORD  0               ;;OUTPUT POINTER
   (1)  016770  000040                  $TKQSRT: .BLKB  32.             ;;TTY KEYBOARD QUEUE
   (1)          017030                  $TKQEND=.
   (1)
   (1)                                  ;*TK INITIALIZE ROUTINE
   (1)                                  ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
   (1)                                  ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
   (1)                                  ;
   (1)                                  ;*CALL:
   (1)                                  ;*      JSR     PC,$TKINT
   (1)                                  ;*      RETURN
   (1)                                  ;
   (1)  017030  005037  016762          $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
   (1)  017034  012737  016770  016764          MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
   (1)  017042  013737  016764  016766          MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
   (1)  017050  012737  017100  000060          MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
   (1)  017056  012737  000200  000062          MOV     #200,@#TKVEC+2  ;;'BR'' LEVEL 4
   (1)  017064  005777  162056                  TST     @$TKB           ;;CLEAR DONE FLAG
   (1)  017070  012777  000100  162046          MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
   (1)  017076  000207                          RTS     PC              ;;RETURN TO CALLER
   (1)
   (1)                                  ;*TK SERVICE ROUTINE
   (1)                                  ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
   (1)                                  ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
   (1)                                  ;*IT IN THE QUEUE.
   (1)                                  ;*IF THE CHARACTER IS A ''CONTROL-C'' (^C) $TKINT IS CALLED AND
   (1)                                  ;*UPON RETURN EXIT IS MADE TO THE ''CONTROL-C'' RESTART ADDRESS (MTEST1)
   (1)                                  ;
   (1)  017100  117746  162042          $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
   (1)  017104  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
   (1)  017110  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL C?
   (1)  017114  001007                          BNE     1$              ;;BRANCH IF NO
   (1)  017116  104401  020244                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
   (1)  017122  004737  017030                  JSR     PC,$TKINT       ;;INIT THE KEYBOARD
   (1)  017126  005726                          TST     (SP)+           ;;CLEAN UP STACK
   (1)  017130  000137  002270                  JMP     MTEST1          ;;CONTROL C RESTART
   (1)  017134  021627  000007          1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
   (1)  017140  001004                          BNE     2$              ;;BRANCH IF NO
   (1)  017142  022737  000176  001140          CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
   (1)  017150  001500                          BEQ     6$              ;;GO TO SWR CHANGE
   (1)
   (1)  017152                          2$:
   (1)  017152  022737  000040  016762          CMP     #32.,$TKCNT     ;;IS THE QUEUE FULL?
   (1)  017160  001004                          BNE     3$              ;;BRANCH IF NO
   (1)  017162  104401  001164                  TYPE    ,$BELL          ;;RING THE TTY BELL
   (1)  017166  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
   (1)  017170  000451                          BR      5$              ;;EXIT
   (1)  017172  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
   (1)  017176  001021                          BNE     32$             ;;BRANCH IF NO
   (1)  017200  005077  161740                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
```

```
(1)  017204  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
(1)  017206  105777  161732          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
(1)  017212  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
(1)  017214  117746  161726                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
(1)  017220  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  017224  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  017230  001366                          BNE     31$             ;;BRANCH IF NO
(1)  017232  012777  000100  161704          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
(1)  017240  000002                          RTI                     ;;RETURN
(1)  017242  005237  016762          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
(1)  017246  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
(1)* 017252  002405                          BLT     4$              ;;BRANCH IF YES
(1)  017254  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
(1)  017260  003002                          BGT     4$              ;;BRANCH IF YES
(1)  017262  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
(1)  017266  112677  177472          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
(1)  017272  005237  016764                  INC     $TKQIN          ;;UPDATE THE POINTER
(1)  017276  023727  016764  017030          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
(1)  017304  001003                          BNE     5$              ;;BRANCH IF NO
(1)  017306  012737  016770  016764          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
(1)  017314  000002                  5$:     RTI                     ;;RETURN
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)                                  ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
(1)                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
(1)                                  ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
(1)  017316  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
(1)  017324  001124                          BNE     15$             ;;EXIT IF NOT
(1)  017326  105777  161612                  TSTB    @$TKS           ;;IS A CHAR WAITING?
(1)  017332  100121                          BPL     15$             ;;IF NOT, EXIT
(1)  017334  117746  161606                  MOVB    @$TKB,-(SP)     ;;YES
(1)  017340  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
(1)  017344  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
(1)  017350  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
(1)                                                                  ;;AND EXIT
(1)
(2)                                  ;;**********************************************************************
(1)                                  ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
(1)                                  ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
(1)                                  ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
(1)  017352  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
(1)  017360  001674                          BEQ     2$              ;;BRANCH IF YES
(1)  017362  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
(1)  017364  004737  017030                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
(1)  017370  005077  161550                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
(1)  017374  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
(1)
(1)  017402  104401  020256                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
(1)  017406  104401  020263          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
(2)  017412  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
(2)  017416  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1)  017420  104401  020274                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
(1)  017424  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
(1)  017426  005046                          CLR     -(SP)           ;;THE NEW SWR
(1)  017430  105777  161510          7$:     TSTB    @$TKS           ;;CHAR THERE?
```

```
(1)  017434  100375                        BPL     7$              ;;IF NOT TRY AGAIN
(1)
(1)  017436  117746  161504                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
(1)  017442  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)
(1)  017446  021627  000003                CMP     (SP),#3         ;;IS IT A CONTROL-C?
(1)  017452  001015                        BNE     9$              ;;BRANCH IF NOT
(1)  017454  104401  020244                TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
(1)  017460  062706  000006                ADD     #6,SP           ;;CLEAN UP STACK
(1)  017464  123727  001135  000001        CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
(1)  017472  001003                        BNE     8$              ;;BRANCH IF NO
(1)  017474  012777  000100  161442        MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
(1)  017502  000137  002270        8$:     JMP     MTEST1          ;;CONTROL-C RESTART
(1)
(1)
(1)  017506  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
(1)  017512  001005                        BNE     10$             ;;BRANCH IF NOT
(1)  017514  104401  020251                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
(1)  017520  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
(1)  017524  000737                        BR      19$             ;;LET'S TRY IT AGAIN
(1)
(1)
(1)  017526  021627  000015        10$:    CMP     (SP),#15        ;;IS IT A <CR>?
(1)  017532  001022                        BNE     16$             ;;BRANCH IF NO
(1)  017534  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
(1)  017540  001403                        BEQ     11$             ;;BRANCH IF YES
(1)  017542  016677  000002  161370        MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  017550  062706  000006        11$:    ADD     #6,SP           ;;CLEAR UP STACK
(1)  017554  104401  001171        14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
(1)  017560  123727  001135  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  017566  001003                        BNE     15$             ;;BRANCH IF NOT
(1)  017570  012777  000100  161346        MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  017576  000002                15$:    RTI                     ;;RETURN
(1)  017600  004737  020622        16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  017604  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
(1)  017610  002420                        BLT     18$             ;;BRANCH IF YES
(1)  017612  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
(1)  017616  003015                        BGT     18$             ;;BRANCH IF YES
(1)  017620  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  017624  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  017630  001403                        BEQ     17$             ;;BRANCH IF YES
(1)  017632  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  017634  006316                        ASL     (SP)            ;;    CHAR OVER TO MAKE
(1)  017636  006316                        ASL     (SP)            ;;    ROOM FOR NEW ONE.
(1)  017640  005266  000002        17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  017644  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  017650  000667                        BR      7$              ;;GET THE NEXT ONE
(1)  017652  104401  001170        18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
(1)  017656  000720                        BR      20$             ;;SIMULATE CONTROL-U
(1)                                .DSABL  LSB
(1)
(1)
(2)                                ;;******************************************************************
(1)                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                ;*CALL:
(1)                                ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
```

```
(1)                                     ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                     ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                     ;
(1)
(1)  017660  011646                     $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
(1)  017662  016666  000004  000002             MOV     4(SP),2(SP)     ;;THE PS
(1)  017670  005066  000004                     CLR     4(SP)           ;;GET READY FOR A CHARACTER
(2)  017674  005046                             CLR     -(SP)           ;;PUT NEW PS ON STACK
(2)  017676  012746  017704                     MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
(2)  017702  000002                             RTI                     ;;POP NEW PC AND PS
(2)  017704                             64$:
(1)  017704  005737  016762             1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
(1)  017710  001775                             BEQ     1$
(1)  017712  005337  016762                     DEC     $TKCNT          ;;DECREMENT THE COUNTER
(1)  017716  117766  177044  000004             MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
(1)  017724  005237  016766                     INC     $TKQOUT         ;;UPDATE THE POINTER
(1)  017730  023727  016766  017030             CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
(1)  017736  001003                             BNE     2$              ;;BRANCH IF NO
(1)  017740  012737  016770  016766             MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
(1)  017746  000002                     2$:     RTI                     ;;RETURN
(2)                                     ;;**********************************************************************
(1)                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                     ;*CALL:
(1)                                     ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                     ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                     ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  017750  010346                     $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  017752  005046                             CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  017754  012703  020204             1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  017760  022703  020244             2$:     CMP     #$TTYIN+32.,R3  ;;BUFFER FULL?
(1)  017764  101456                             BLOS    4$              ;;BR IF YES
(1)  017766  104411                             RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  017770  112613                             MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  017772  122713  000177             10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  017776  001022                             BNE     5$              ;;BR IF NO
(1)  020000  005716                             TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  020002  001007                             BNE     6$              ;;BR IF NO
(1)  020004  112737  000134  020202             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  020012  104401  020202                     TYPE    ,9$
(1)  020016  012716  177777                     MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  020022  005303                     6$:     DEC     R3              ;;BACKUP BY ONE
(1)  020024  020327  020204                     CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  020030  103434                             BLO     4$              ;;BR IF YES
(1)  020032  111337  020202                     MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  020036  104401  020202                     TYPE    ,9$             ;;GO TYPE
(1)  020042  000746                             BR      2$              ;;GO READ ANOTHER CHAR.
(1)  020044  005716                     5$:     TST     (SP)            ;;RUBOUT KEY SET?
(1)  020046  001406                             BEQ     7$              ;;BR IF NO
(1)  020050  112737  000134  020202             MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  020056  104401  020202                     TYPE    ,9$
(1)  020062  005016                             CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  020064  122713  000025             7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  020070  001003                             BNE     8$              ;;BR IF NO
(1)  020072  104401  020251                     TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U''
(1)  020076  000726                             BR      1$              ;;GO START OVER
```

```
(1)  020100  122713  000022          8$:    CMPB   #22,(R3)          ;;IS CHARACTER A ''^R''?
(1)  020104  001011                        BNE    3$                 ;;BRANCH IF NO
(1)  020106  105013                        CLRB   (R3)               ;;CLEAR THE CHARACTER
(1)  020110  104401  001171                TYPE   ,$CRLF             ;;TYPE A ''CR'' & ''LF''
(1)  020114  104401  020204                TYPE   ,$TTYIN            ;;TYPE THE INPUT STRING
(1)  020120  000717                        BR     2$                 ;;GO PICKUP ANOTHER CHACTER
(1)  020122  104401  001170          4$:   TYPE   ,$QUES             ;;TYPE A '?'
(1)  020126  000712                        BR     1$                 ;;CLEAR THE BUFFER AND LOOP
(1)  020130  111337  020202          3$:   MOVB   (R3),9$            ;;ECHO THE CHARACTER
(1)  020134  104401  020202                TYPE   ,9$
(1)  020140  122723  000015                CMPB   #15,(R3)+          ;;CHECK FOR RETURN
(1)  020144  001305                        BNE    2$                 ;;LOOP IF NOT RETURN
(1)  020146  105063  177777                CLRB   -1(R3)             ;;CLEAR RETURN (THE 15)
(1)  020152  104401  001172                TYPE   ,$LF               ;;TYPE A LINE FEED
(1)  020156  005726                        TST    (SP)+              ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  020160  012603                        MOV    (SP)+,R3           ;;RESTORE R3
(1)  020162  011646                        MOV    (SP),-(SP)         ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  020164  016666  000004  000002        MOV    4(SP),2(SP)        ;;      FIRST ASCII CHARACTER ON IT
(1)  020172  012766  020204  000004        MOV    #$TTYIN,4(SP)
(1)  020200  000002                        RTI                       ;;RETURN
(1)  020202     000          9$:   .BYTE   0                         ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  020203     000                .BYTE   0                         ;;TERMINATOR
(1)  020204  000040          $TTYIN: .BLKB  32.                      ;;RESERVE 32. BYTES FOR TTY INPUT
(1)  020244  041536  005015     000  $CNTLC: .ASCIZ  /^C/<15><12>     ;;CONTROL ''C''
(1)  020251     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>     ;;CONTROL ''U''
(1)  020256  043536  005015     000  $CNTLG: .ASCIZ  /^G/<15><12>     ;;CONTROL ''G''
(1)  020263     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  020270  036440  000040
(1)  020274  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
(1)  020302  020075     000
(1)          020306                 .EVEN
7332                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)                                 ;;*****************************************************************
(1)                                 ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)                                 ;*CHANGE IT TO BINARY.
(1)                                 ;*CALL:
(1)                                 ;*      RDOCT                      ;;READ AN OCTAL NUMBER
(1)                                 ;*      RETURN HERE                ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1)                                 ;*                                 ;;HIGH ORDER BITS ARE IN $HIOCT
(1)
(1)  020306  011646          $RDOCT: MOV    (SP),-(SP)          ;;PROVIDE SPACE FOR THE
(1)  020310  016666  000004  000002  MOV    4(SP),2(SP)         ;;INPUT NUMBER
(3)  020316  010046                  MOV    R0,-(SP)            ;;PUSH R0 ON STACK
(3)  020320  010146                  MOV    R1,-(SP)            ;;PUSH R1 ON STACK
(3)  020322  010246                  MOV    R2,-(SP)            ;;PUSH R2 ON STACK
(1)  020324  104412          1$:     RDLIN                      ;;READ AN ASCIZ LINE
(1)  020326  012600                  MOV    (SP)+,R0            ;;GET ADDRESS OF 1ST CHARACTER
(1)  020330  005001                  CLR    R1                  ;;CLEAR DATA WORD
(1)  020332  005002                  CLR    R2
(1)  020334  112046          2$:     MOVB   (R0)+,-(SP)         ;;PICKUP THIS CHARACTER
(1)  020336  001412                  BEQ    3$                  ;;IF ZERO GET OUT
(1)  020340  006301                  ASL    R1                  ;;*2
(1)  020342  006102                  ROL    R2
(1)  020344  006301                  ASL    R1                  ;;*4
(1)  020346  006102                  ROL    R2
```

```
(1)  020350  006301                    ASL     R1              ;;*8
(1)  020352  006102                    ROL     R2
(1)  020354  042716  177770            BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
(1)  020360  062601                    ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
(1)  020362  000764                    BR      2$              ;;LOOP
(1)  020364  005726                3$: TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
(1)  020366  010166  000012            MOV     R1,12(SP)       ;;SAVE THE RESULT
(1)  020372  010237  020406            MOV     R2,$HIOCT
(3)  020376  012602                    MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  020400  012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  020402  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  020404  000002                    RTI                     ;;RETURN
(1)  020406  000000            $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
 7334                                   ;CAUTION        THE FIRST 4 LOCATIONS ARE OVERLAYED TO LOWER CPU LEVEL
 7335                                   ;       THIS OVERLAY OCCURS AFTER ''SETUP''
 7336                                   .SBTTL  TYPE ROUTINE
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
  (1)                                   ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
  (1)                                   ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
  (1)                                   ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
  (1)                                   ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
  (1)                                   ;*
  (1)                                   ;*CALL:
  (1)                                   ;*1) USING A TRAP INSTRUCTION
  (1)                                   ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
  (1)                                   ;*OR
  (1)                                   ;*      TYPE
  (1)                                   ;*      MESADR
  (1)                                   ;*
  (1)
  (1)  020410  105737  001157           $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
  (1)  020414  100002                           BPL     1$              ;;BR IF YES
  (1)  020416  000000                           HALT                    ;;HALT HERE IF NO TERMINAL
  (1)  020420  000430                           BR      3$              ;;LEAVE
  (1)  020422  010046                   1$:     MOV     R0,-(SP)        ;;SAVE R0
  (1)  020424  017600  000002                   MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
  (1)  020430  122737  000001  001214           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
  (1)  020436  001011                           BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
  (1)  020440  132737  000100  001215           BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
  (1)  020446  001405                           BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
  (1)  020450  010037  020460                   MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
  (1)  020454  004737  020700                   JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
  (1)  020460  000000                   61$.    .WORD   0               ;;MESSAGE ADDRESS
  (1)  020462  132737  000040  001215   62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
  (1)  020470  001003                           BNE     60$             ;;YES,SKIP TYPE OUT
  (1)  020472  112046                   2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
  (1)  020474  001005                           BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
  (1)  020476  005726                           TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
  (1)  020500  012600                   60$:    MOV     (SP)+,R0        ;;RESTORE R0
  (1)  020502  062716  000002           3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
  (1)  020506  000002                           RTI                     ;;RETURN
  (1)  020510  122716  000011           4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
  (1)  020514  001430                           BEQ     8$
  (1)  020516  122716  000200                   CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
  (1)  020522  001006                           BNE     5$
  (1)  020524  005726                           TST     (SP)+           ;;POP  <CR><LF> EQUIV
  (1)  020526  104401                           TYPE                    ;;TYPE A CR AND LF
  (1)  020530  001171                           $CRLF
  (1)  020532  105037  020666                   CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
  (1)  020536  000755                           BR      2$              ;;GET NEXT CHARACTER
  (1)  020540  004737  020622           5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
  (1)  020544  123726  001156           6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
  (1)  020550  001350                           BNE     2$              ;;IF NO GO GET NEXT CHAR.
  (1)  020552  013746  001154                   MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
  (1)                                           ;;AND THE NULL CHAR.
  (1)  020556  105366  000001           7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
  (1)  020562  002770                           BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
```

```
(1)  020564  004737  020622                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  020570  105337  020666                    DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  020574  000770                            BR      7$              ;;LOOP
(1)
(1)                                     ;HORIZONTAL TAB PROCESSOR
(1)
(1)  020576  112716  000040            8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  020602  004737  020622            9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  020606  132737  000007  020666            BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  020614  001372                            BNE     9$              ;;TAB STOP
(1)  020616  005726                            TST     (SP)+           ;;POP SPACE OFF STACK
(1)  020620  000724                            BR      2$              ;;GET NEXT CHARACTER
(1)  020622  105777  160322            $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  020626  100375                            BPL     $TYPEC
(1)  020630  116677  000002  160314            MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  020636  122766  000015  000002            CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  020644  001003                            BNE     1$              ;;BRANCH IF NO
(1)  020646  105037  020666                    CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  020652  000406                            BR      $TYPEX          ;;EXIT
(1)  020654  122766  000012  000002  1$:       CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  020662  001402                            BEQ     $TYPEX          ;;BRANCH IF YES
(1)  020664  105227                            INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  020666  000000            $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  020670  000207            $TYPEX: RTS     PC
(1)
7337                                    .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                     ;;**************************************************************************
(1)  020672  112737  000001  021136  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  020700  112737  000001  021134  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  020706  000403                            BR      $ATYC
(1)  020710  112737  000001  021136  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  020716                          $ATYC:
(3)  020716  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  020720  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  020722  105737  021134                    TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  020726  001450                            BEQ     5$              ;;IF NOT:  BR
(1)  020730  122737  000001  001214            CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  020736  001031                            BNE     3$              ;;IF NOT:  BR
(1)  020740  132737  000100  001215            BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  020746  001425                            BEQ     3$              ;;IF NOT:  BR
(1)  020750  017600  000004                    MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  020754  062766  000002  000004            ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  020762  005737  001174            1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  020766  001375                            BNE     1$              ;;IF NOT:  WAIT
(1)  020770  010037  001210                    MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  020774  105720                    2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  020776  001376                            BNE     2$
(1)  021000  163700  0012*0                    SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  021004  006200                            ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  021006  010037  001212                    MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  021012  012737  000004  001174            MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  021020  000413                            BR      5$
(1)  021022  017637  000004  021046  3$:       MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  021030  062766  000002  000004            ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  021036  013746  177776                    MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
```

```
(1)  021042  004737  020410              JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  021046  000000              4$:     .WORD   0
(1)  021050                      5$:
(1)  021050  105737  021136     10$:     TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  021054  001416                       BEQ     12$             ;;IF NOT:  BR
(1)  021056  005737  001214               TST     $ENV            ;;RUNNING UNDER APT?
(1)  021062  001413                       BEQ     12$             ;;IF NOT:  BR
(1)  021064  005737  001174     11$:     TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  021070  001375                       BNE     11$             ;;IF NOT:  WAIT
(1)  021072  017637  000004  001176       MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  021100  062766  000002  000004       ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  021106  005237  001174               INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  021112  105037  021136     12$:     CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  021116  105037  021135               CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  021122  105037  021134               CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  021126  012601                       MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  021130  012600                       MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  021132  000207                       RTS     PC              ;;RETURN
(1)  021134     000              $MFLG:   .BYTE   0               ;;MESSG. FLAG
(1)  021135     000              $LFLG:   .BYTE   0               ;;LOG FLAG
(1)  021136     000              $FFLG:   .BYTE   0               ;;FATAL FLAG
(1)          021140                       .EVEN
(1)          000200              APTSIZE=200
(1)          000001              APTENV=001
(1)          000100              APTSPOOL=100
(1)          000040              APTCSUP=040
```

```
 7339                                   .SBTTL  POWER DOWN AND UP ROUTINES
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   ;POWER DOWN ROUTINE
  (1)  021140  012737  021300  000024   $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
  (1)  021146  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (3)  021154  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (3)  021156  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (3)  021160  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (3)  021162  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (3)  021164  010446                           MOV     R4,-(SP)        ;;PUSH R4 ON STACK
  (3)  021166  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
  (3)  021170  017746  157744                   MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
  (1)  021174  010637  021304                   MOV     SP,$SAVR6       ;;SAVE SP
  (1)  021200  012737  021212  000024           MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
  (1)  021206  000000                           HALT
  (1)  021210  000776                           BR      .-2             ;;HANG UP
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   ;POWER UP ROUTINE
  (1)  021212  012737  021300  000024   $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
  (1)  021220  013706  021304                   MOV     $SAVR6,SP       ;;GET SP
  (1)  021224  005037  021304                   CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
  (1)  021230  005237  021304           1$:     INC     $SAVR6          ;;WAIT FOR THE INC
  (1)  021234  001375                           BNE     1$              ;;OF  WORD
  (3)  021236  012677  157676                   MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
  (3)  021242  012605                           MOV     (SP)+,R5        ;;POP STACK INTO R5
  (3)  021244  012604                           MOV     (SP)+,R4        ;;POP STACK INTO R4
  (3)  021246  012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
  (3)  021250  012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
  (3)  021252  012601                           MOV     (SP)+,R1        ;;POP STACK INTO R1
  (3)  021254  012600                           MOV     (SP)+,R0        ;;POP STACK INTO R0
  (1)  021256  012737  021140  000024           MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
  (1)  021264  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;PRIO:7
  (1)  021272  104401                           TYPE                    ;REPORT THE POWER FAILURE
  (1)  021274  021306                   $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
  (1)  021276  000002                           RTI
  (1)  021300  000000                   $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
  (1)  021302  000776                           BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
  (1)  021304  000000                   $SAVR6: 0                       ;;PUT THE SP HERE
  (1)  021306  005015  047520  042527   $POWER: .ASCIZ  <15><12>'POWER'
  (1)  021314  000122
  (1)                                           .EVEN
```

```
                                        ;*
                                        ;*THIS ROUTINE WILL PROTECT THE PROGRAM
                                        ;*FROM INTERUPTS (BAD ONES).
                                        ;*
                                        ;*THE TRAP CATCHER IS SET UP FOR
                                        ;*      .WORD   .+2
                                        ;*      JSR PC,R0
                                        ;*
                                        ;*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 4 WE HAVE A
                                        ;*POINTER HERE. IF THIS CONDITION CAUSES A TRAP TO LOC. 4.
                                        ;*WE WILL REPORT IT IN THE SAME MANNER THAT WER WOULD
                                        ;*REPORT ANY OTHER ERROR.
                                        ;*IF A BUSS ERROR TRAP DID OCCUT AND CAUSE A TRAP TO 4.
                                        ;*WE WILL HALT.

021316  011637  021572          IOTRD:  MOV     (SP),TRTO       ;GET WHERE WE CAME TO.
021322  162737  000004  021572          SUB     #4,TRTO         ;FORM READ ADDR.

021330  023727  021572  001000          CMP     TRTO,#1000      ;DID TRAP FROM LESS THAN ADDR. 1000?
021336  003402                          BLE     2$              ;NO-CONTINUE.

021340  000000                  1$:     HALT                    ;A BUSS ERROR TIME OUT TRAP BROUGHT US HERE.
                                                                ;ADDRESS CONTAINED IN TRTO.

021342  000776                          BR      1$              ;DON'T ALLOW CONTINUE.

021344  016637  000004  021574  2$:     MOV     4(SP),TRFRO     ;GET TRAPPED FROM ADDR.
021352  062706  000004                  ADD     #4,SP                   ;/ADD #4 TO STACK POINTER.

021356  122737  000044  001102          CMPB    #44,$TSTNM      ;LESS THAN INTERRUPT TESTS?
021364  003402                          BLE     3$              ;NO MUST BE WRONG VECTOR.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

021366  104004                          ERROR   4               ;ERROR! ILLEGAL INTERRUPT OR
                                                                ;INTERRUPT TO WRONG VECTOR.
                                                                ;IF TEST NO. IS LESS THAN 46,ITS
                                                                ;LIKELY(BUT NO EXCLUSIVELY)TO BE A
                                                                ;DEVICE OTHER THAN THE DEVICE UNDER TEST.
                                                                ;IF THE INTERRUPT OCCURED
                                                                ;DURING AN INTERRUPT TEST, I'D
                                                                ;SUSPECT A PROBLEM WITH THE DEVICE UNDER TEST.
                                                                ;IF THE ADDRESS THE INTERRUPT
                                                                ;VECTORED TO IS WITHIN THE RANGE OF
                                                                ;VECTORS ASSIGNED TO THE DEVICE,
                                                                ;THEN I'D SUSPECT THE DEVICE
                                                                ;INTERRUPTD 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 INTERRUPT
                                                        ;VICTOR ON THE BUS DURING THE INTERRUPT
                                                        ;PROCESS.
                                                        ;       NOTE:
                                                        ;FOR THIS ERROR - DON'T USE
                                                        ; 'LOOP ON ERROR'' OPTION.
                                                        ;ALSO EXPECT THAT THE INTERRUPT TEST TO
                                                        ;WILL REPOT THAT THE DEVICE DIDN'T
                                                        ;INTERRUPT.
                                                        ;FOLLOW THE RECOMMENDED PROCEEDURE
                                                        ; IN THE DOCUMENT (ON THIS DIAGNOSTIC)
                                                        ;FOR LOOPING ON TEST.

                                ;;$$$$$$$$$$$$$$$$$$$$$$$$>>> ERROR <<<$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

021370  000002                        RTI
021372  022626                3$:     CMP     (SP)+,(SP)+     ;CLEAN THE STACK
021374  005737  001202                TST     $PASS           ;IS THIS THE FIRST PASS
021400  001026                        BNE     4$              ;NO DONT REPORT
021402  104401  024073                TYPE,   VTMSG           ;TELL THE OPER.
021406  013746  001206                MOV     $UNIT,-(SP)
021412  005216                        INC     (SP)
021414  104405                        TYPDS
021416  000240                        NOP
021420  000240                        NOP
021422  104401  024121                TYPE,   VTMSG3          ;TYPE EXPECT. INTR.
021426  013746  001424                MOV     VECT1,-(SP)     ;GET DEFAULT
021432  104403                        TYPOS
021434     003     001                .BYTE   3,1
021436  104401  024151                TYPE,   VTMSG1          ;TYPE RCVD TEXT
021442  013746  021572                MOV     TRTO,-(SP)      ;GET ACTUAL
021446  104403                        TYPOS
021450     003     001                .BYTE   3,1
021452  104401  024201                TYPE,   VTMSG2          ;ADD REMAINDER TEXT
021456  013777  001426  157740  4$:   MOV     VECTP,@VECT1
021464  013777  001432  157736        MOV     VECT2P,@VECT2
021472  012777  004700  157726        MOV     #4700,@VECTP
021500  012777  004700  157724        MOV     #4700,@VECT2P
021506  013737  021572  001424        MOV     TRTO,VECT1
021514  042737  000007  001424        BIC     #7,VECT1
021522  013737  001424  001426        MOV     VECT1,VECTP
021530  062737  000002  001426        ADD     #2,VECTP
021536  013737  001424  001430        MOV     VECT1,VECT2
021544  062737  000004  001430        ADD     #4,VECT2
021552  013737  001430  001432        MOV     VECT2,VECT2P
021560  062737  000002  001432        ADD     #2,VECT2P
021566  000177  157314                JMP     @$LPADR         ;START TEST OVER AGAIN.
021572  000000              TRTO:     .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTERRUPTED TO.
021574  000000              TRFRO:    .WORD   0               ;CONTAINS ADDR. WE TRAPPED OR INTR. FROM.
```

```
 7444                                   .SBTTL  TRAP DECODER
  (1)
  (2)                                   ;;**********************************************************************
  (1)                                   ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
  (1)                                   ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
  (1)                                   ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
  (1)                                   ;*GO TO THAT ROUTINE.
  (1)
  (1)  021576  010046                   $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
  (1)  021600  016600  000002                   MOV     2(SP),R0        ;;GET TRAP ADDRESS
  (1)  021604  005740                           TST     -(R0)           ;;BACKUP BY 2
  (1)  021606  111000                           MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
  (1)  021610  006300                           ASL     R0              ;;POSITION FOR INDEXING
  (1)  021612  016000  021632                   MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
  (1)  021616  000200                           RTS     R0              ;;GO TO ROUTINE
  (1)
  (1)
  (1)                                   ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
  (1)
  (1)  021620  011646                   $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
  (1)  021622  016666  000004  000002           MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
  (1)  021630  000002                           RTI                     ;;RESTORE THE PSW
  (1)
  (3)                                   .SBTTL  TRAP TABLE
  (3)
  (3)                                   ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
  (3)                                   ;*BY THE ''TRAP'' INSTRUCTION.
  (3)
  (3)                                   ;       ROUTINE
  (3)                                   ;       -------
  (3)  021632  021620                   $TRPAD: .WORD   $TRAP2
  (3)  021634  020410                           $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
  (3)  021636  015364                           $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
  (3)  021640  015340                           $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
  (3)  021642  015400                           $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
  (3)  021644  015642                           $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
  (3)  021646  015566                           $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
  (1)
  (3)  021650  017406                           $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
  (1)
  (3)  021652  017316                           $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
  (3)  021654  017660                           $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
  (3)  021656  017750                           $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
  (3)  021660  020306                           $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
```

```
 7446                                                   .SBTTL  ASCII MESSAGES
 7447
 7448  021662  005015  047115  045503  EM1:    .ASCIZ  <15><12>/MNCKW (CLOCK)  STATUS FUNCTION ERROR/
       021670  020127  041450  047514
       021676  045503  004451  052123
       021704  052101  051525  043040
       021712  047125  052103  047511
       021720  020116  051105  047522
       021726  000122
 7449  021730  005015  047115  045503  EM2:    .ASCIZ  <15><12>/MNCKW (CLOCK)  STATUS DATA ERROR/
       021736  020127  041450  047514
       021744  045503  004451  052123
       021752  052101  051525  042040
       021760  052101  020101  051105
       021766  047522  000122
 7450  021772  005015  047115  045503  EM3:    .ASCIZ  <15><12>/MNCKW (CLOCK)  BUFFER DATA ERROR/
       022000  020127  041450  047514
       022006  045503  004451  052502
       022014  043106  051105  042040
       022022  052101  020101  051105
       022030  047522  000122
 7451  022034  046600  041516  053513  EM4:    .ASCIZ  <200>/MNCKW (CLOCK)     INTERRUPT ERROR/
       022042  024040  046103  041517
       022050  024513  044411  052116
       022056  051105  052522  052120
       022064  042440  051122  051117
       022072     000
 7452  022073     015  046412  041516  EM5:    .ASCIZ  <15><12>/MNCKW (CLOCK)  COUNT REG. ERROR/
       022100  053513  024040  046103
       022106  041517  024513  041411
       022114  052517  052116  051040
       022122  043505  020056  051105
       022130  047522  000122
 7453  022134  046600  041516  053513  EM6:    .ASCIZ  <200>/MNCKW (CLOCK)     EXISTING UNIT FAILED TO RESPOND/
       022142  024040  046103  041517
       022150  024513  042411  044530
       022156  052123  047111  020107
       022164  047125  052111  043040
       022172  044501  042514  020104
       022200  047524  051040  051505
       022206  047520  042116     000
 7454  022213     015  046412  041516  EM11:   .ASCIZ  <15><12>#MNCKW (CLOCK)  COUNT ERROR #
       022220  053513  024040  046103
       022226  041517  024513  041411
       022234  052517  052116  042440
       022242  051122  051117  000040
 7455  022250  005015  047115  045503  EM12:   .ASCIZ  <15><12>#MNCKW (CLOCK)  COUNT FUNCTION ERROR#
       022256  020127  041450  047514
       022264  045503  004451  047503
       022272  047125  020124  052506
       022300  041516  044524  047117
       022306  042440  051122  051117
       022314     000
 7456  022315     200  047115  045503  EM13:   .ASCIZ  <200>#MNCKW (CLOCK)     INCORRECT I.D. VALUE#
       022322  020127  041450  047514
       022330  045503  004451  047111
```

```
        022336  047503  051122  041505
        022344  020124  027111  027104
        022352  053040  046101  042525
        022360     000
 7457   022361     015  046412  041516  EM16:   .ASCIZ  <15><12>#MNCKW (CLOCK)  CLOCK INTERRUPT ERROR #
        022366  053513  024040  046103
        022374  041517  024513  041411
        022402  047514  045503  044440
        022410  052116  051105  052522
        022416  052120  042440  051122
        022424  051117  000040
 7458   022430  005015  047115  045503  EM20:   .ASCIZ  <15><12>#MNCKW (CLOCK)  REPEATABILITY ERROR #
        022436  020127  041450  047514
        022444  045503  004451  042522
        022452  042520  052101  041101
        022460  046111  052111  020131
        022466  051105  047522  020122
        022474     000
 7459   022475     015  046412  041516  EM26:   .ASCIZ  <15><12>#MNCKW (CLOCK)  DOES NOT EXIST <BUS ERROR> CHECK BASE ADDRESS SW
        022502  053513  024040  046103
        022510  041517  024513  042011
        022516  042517  020123  047516
        022524  020124  054105  051511
        022532  020124  041074  051525
        022540  042440  051122  051117
        022546  020076  044103  041505
        022554  020113  040502  042523
        022562  040440  042104  042522
        022570  051523  051440  044527
        022576  041524  042510  000123
 7460
 7461   022604  005015  047125  052111  DH1:    .ASCIZ  <15><12>#UNIT    ERRPC   ASR     WAS     S/B#
        022612  042411  051122  041520
        022620  040411  051123  053411
        022626  051501  051411  041057
        022634     000
 7462   022635     200  047125  052111  DH2:    .ASCIZ  <200>#UNIT       ERRPC#
        022642  042411  051122  041520
        022650     000
 7463   022651     015  052412  044516  DH3:    .ASCIZ  <15><12>#UNIT    ERRPC   ABR     WAS     S/B#
        022656  004524  051105  050122
        022664  004503  041101  004522
        022672  040527  004523  027523
        022700  000102
 7464   022702  052600  044516  004524  DH4A:   .ASCIZ  <200>#UNIT       ERRPC   TO      FROM ADDR.#
        022710  051105  050122  020103
        022716  020040  047524  020040
        022724  020040  020040  051106
        022732  046517  040440  042104
        022740  027122     000
 7465   022743     015  052412  044516  DH12:   .ASCIZ  <15><12>#UNIT    ERRPC   ASR     #
        022750  004524  051105  050122
        022756  004503  051501  004522
        022764     000
 7466   022765     015  052412  044516  DH20:   .ASCIZ  <15><12>#UNIT    ERRPC   ASR     2NDCNT  1STNCT  #
        022772  004524  051105  050122
```

```
        023000  004503  051501  004522
        023006  047062  041504  052116
        023014  030411  052123  041516
        023022  004524     000
 7467   023025     015  052412  044516  DH26:    .ASCIZ  <15><12>#UNIT   ERRPC   CLOCK ADDR.#
        023032  004524  051105  050122
        023040  004503  046103  041517
        023046  020113  042101  051104
        023054  000056
 7468   023056  050200  047101  046105  PRIME0: .ASCII  <200>'PANEL: PULL OUT BOTH ST POTS AND THEN TURN'
        023064  020072  052520  046114
        023072  047440  052125  041040
        023100  052117  020110  052123
        023106  050040  052117  020123
        023114  047101  020104  044124
        023122  047105  052040  051125
        023130     116
 7469   023131     200  020040  020040          .ASCII  <200>''       THEM COMPLETELY CW OR CCW''<200>
        023136  020040  052040  042510
        023144  020115  047503  050115
        023152  042514  042524  054514
        023160  041440  020127  051117
        023166  041440  053503     200
 7470   023173     104  040527  043122          .ASCIZ  'DWARF: S2  ALL SWITCHES OFF''<200><200>
        023200  020072  031123  020040
        023206  046101  020114  053523
        023214  052111  044103  051505
        023222  047440  043106  100200
        023230     000
 7471   023231     114  036440  046040  PRIME1: .ASCII  /L = LOGIC TEST WITH NO TEST MODULE CONNECTED/
        023236  043517  041511  052040
        023244  051505  020124  044527
        023252  044124  047040  020117
        023260  042524  052123  046440
        023266  042117  046125  020105
        023274  047503  047116  041505
        023302  042524     104
 7472   023305     015     012                  .BYTE   15,12
 7473   023307     104  036440  046040          .ASCII  /D = LOGIC TEST WITH A TEST MODULE CONNECTED TO ONE UNIT/
        023314  043517  041511  052040
        023322  051505  020124  044527
        023330  044124  040440  052040
        023336  051505  020124  047515
        023344  052504  042514  041440
        023352  047117  042516  052103
        023360  042105  052040  020117
        023366  047117  020105  047125
        023374  052111
 7474   023376     015     012                  .BYTE   15,12
 7475   023400  020107  020075  042507          .ASCII  /G = GET NEW SWITCH REGISTER VALUE/
        023406  020124  042516  020127
        023414  053523  052111  044103
        023422  051040  043505  051511
        023430  042524  020122  040526
        023436  052514     105
 7476   023441     015     012                  .BYTE   15,12
```

```
 7477  023443   102  036440  041040          .ASCII  /B = BASE OR VECTOR ADDRESS CHANGE/
       023450  051501  020105  051117
       023456  053040  041505  047524
       023464  020122  042101  051104
       023472  051505  020123  044103
       023500  047101  042507
 7478  023504   015   012                    .BYTE   15,12
 7479  023506  020110  020075  042510        .ASCIZ  /H = HELP THE OPERATOR AND RETYPE THIS LIST    /
       023514  050114  052040  042510
       023522  047440  042520  040522
       023530  047524  020122  047101
       023536  020104  042522  054524
       023544  042520  052040  044510
       023552  020123  044514  052123
       023560  020040  020040   000
 7480  023565   015   012              DOT:  .BYTE   15,12
 7481  023567   124  050131  020105          .ASCIZ  /TYPE THE ''TEST CHARACTER'' THEN DEPRESS 'RETURN KEY''  /
       023574  044124  020105  052042
       023602  051505  020124  044103
       023610  051101  041501  042524
       023616  021122  052040  042510
       023624  020116  042504  051120
       023632  051505  020123  051042
       023640  052105  051125  020116
       023646  042513  021131  020040
       023654   000
 7482  023655   200  047115  045503  ADROUT: .ASCIZ  <200>/MNCKW (CLOCK) BASE ADDRESS </
       023662  020127  041450  047514
       023670  045503  020051  040502
       023676  042523  040440  042104
       023704  042522  051523  036040
       023712   000
 7483  023713   200  047115  045503  VECOUT: .ASCIZ  <200>/MNCKW (CLOCK) VECTOR ADDRESS </
       023720  020127  041450  047514
       023726  045503  020051  042526
       023734  052103  051117  040440
       023742  042104  042522  051523
       023750  036040   000
 7484  023753   076  037440  000040  ENDOUT: .ASCIZ  /> ? /
 7485  023760  050200  047522  051107  FOUND1: .ASCIZ  <200>\PROGRAM DETECTED \
       023766  046501  042040  052105
       023774  041505  042524  020104
       024002   000
 7486  024003   040  047115  045503  FOUND2: .ASCIZ  \ MNCKW (CLOCK)'S    \
       024010  020127  041450  047514
       024016  045503  023451  020123
       024024  020040   000
 7487  024027   040  052073  052117  ERRTOT: .ASCIZ  \ ;TOTAL ERROR COUNT = \
       024034  046101  042440  051122
       024042  051117  041440  052517
       024050  052116  036440  000040
 7488  024056  035440  040502  020104  MESGD:  .ASCIZ  \ ;BAD UNITS \
       024064  047125  052111  020123
       024072   000
 7489  024073   200  047115  045503  VTMSG:  .ASCIZ  <200>/MNCKW (CLOCK) UNIT #/
       024100  020127  041450  047514
```

```
        024106  045503  020051  047125
        024114  052111  021440     000
 7490   024121     200  054105  042520  VTMSG3: .ASCIZ  <200>/EXPECTED INTERRUPT AT /
        024126  052103  042105  044440
        024134  052116  051105  052522
        024142  052120  040440  020124
        024150     000
 7491   024151     040  042522  042503  VTMSG1: .ASCIZ  / RECEIVED INTERRUPT AT /
        024156  053111  042105  044440
        024164  052116  051105  052522
        024172  052120  040440  020124
        024200     000
 7492   024201     200  046120  040505  VTMSG2: .ASCII  <200>/PLEASE CHECK VECTOR SWITCHES/<200>
        024206  042523  041440  042510
        024214  045503  053040  041505
        024222  047524  020122  053523
        024230  052111  044103  051505
        024236     200
 7493   024237     011  042522  052123          .ASCIZ  /       RESTARTING TEST/<200>
        024244  051101  044524  043516
        024252  052040  051505  100124
        024260     000
 7494
 7495           024262                          .EVEN
 7496
 7497   024262  001504  001116  001420  DT1:    .WORD   UNITBD,$ERRPC,ASR,$BDDAT,$GDDAT,0
        024270  001126  001124  000000
 7498   024276  001504  001116  000000  DT2:    .WORD   UNITBD,$ERRPC,0
 7499   024304  001504  001116  001422  DT3:    .WORD   UNITBD,$ERRPC,ABR,$BDDAT,$GDDAT,0
        024312  001126  001124  000000
 7500   024320  001504  001116  021572  DT4:    .WORD   UNITBD,$ERRPC,TRTO,TRFRO,0
        024326  021574  000000
 7501   024332  001504  001116  001420  DT12:   .WORD   UNITBD,$ERRPC,ASR,0
        024340  000000
 7502   024342  001504  001116  001420  DT20:   .WORD   UNITBD,$ERRPC,ASR,$BDDAT,$GDDAT,0
        024350  001126  001124  000000
 7503   024356  001504  001116  001420  DT22:   .WORD   UNITBD,$ERRPC,ASR,$BDDAT,$TMP0,0
        024364  001126  001446  000000
 7504   024372  001504  001116  001446  DT26:   .WORD   UNITBD,$ERRPC,$TMP0,0
        024400  000000
 7505
 7506   024402     000     000     000  DF0:    .BYTE   0,0,0,0,0,0,0
        024405     000     000     000
        024410     000
 7507
 7508           000001                          .END
```

```
ABASE = 171020          5654#  5664   5731   5732
ABR     001422          5732#  5896*  5897*  5951   6040*  6042*  6115*  6123   6140*  6148   6210*  6215   6302*
                        6306   6326*  6345   6358*  6372*  6401*  6421*  6472*  6474*  6476*  6478*  6480*  6482*
                        6488*  6498   6513*  6546*  6731*  6767*  6775   6786*  6797   6937*  6939*  6941*  6943*
                        6945*  6969*  6981   6990   7002   7019*  7031   7040   7075*  7105*  7202   7499
ACDW1 = 000000          5664
ACDW2 = 000000          5664
ACPUOP= 000000          5664
ADDW0 = 000000          5664
ADDW1 = 000000          5664
ADDW10= 000000          5664
ADDW11= 000000          5664
ADDW12= 000000          5664
ADDW13= 000000          5664
ADDW14= 000000          5664
ADDW15= 000000          5664
ADDW2 = 000000          5664
ADDW3 = 000000          5664
ADDW4 = 000000          5664
ADDW5 = 000000          5664
ADDW6 = 000000          5664
ADDW7 = 000000          5664
ADDW8 = 000000          5664
ADDW9 = 000000          5664
ADEVCT= 000000          5664
ADEVM = 000000          5664
ADROUT  023655          7255   7482#
AENV  = 000000          5664
AENVM = 000000          5664
AFATAL= 000000          5664
AMADR1= 000000          5664
AMADR2= 000000          5664
AMADR3= 000000          5664
AMADR4= 000000          5664
AMAMS1= 000000          5664
AMAMS2= 000000          5664
AMAMS3= 000000          5664
AMAMS4= 000000          5664
AMSGAD= 000000          5664
AMSGLG= 000000          5664
AMSGTY= 000000          5664
AMTYP1= 000000          5664
AMTYP2= 000000          5664
AMTYP3= 000000          5664
AMTYP4= 000000          5664
ANYKEY  015120          7143   7159   7172   7280#
ANY2    015166          7073   7103   7141   7157   7170   7189   7225   7282#
APASS = 000000          5664
APRIOR= 000200          5656#  5664   5737
APTCSU= 000040          7336   7337#
APTENV= 000001          7327   7336   7337#
APTSIZ= 000200          5775   7337#
APTSPO= 000100          7336   7337#
ASK     001500          5757#  5877*  7126   7130*  7131*  7132   7134   7136*
ASR     001420          5731#  5887*  5896   5949   5998*  5999*  6000*  6001*  6002*  6003*  6004*  6005*  6006*
                        6007*  6059*  6060*  6065   6085*  6087*  6091*  6097*  6099   6114*  6119*  6123*  6139*
```

```
                        6144* 6148* 6172* 6177  6232* 6233* 6234* 6236  6256* 6257  6264* 6265* 6266
                        6271* 6272  6279* 6280* 6281  6288  6293* 6301* 6303* 6304* 6306* 6325* 6330*
                        6332* 6345* 6355* 6371* 6374* 6376* 6378  6399* 6403* 6405* 6409  6420* 6422*
                        6424* 6426  6433* 6472* 6474* 6476* 6478* 6480* 6482* 6487* 6490* 6492* 6498*
                        6500  6508* 6512* 6514* 6519  6545* 6548* 6549* 6556* 6562* 6569* 6571* 6572*
                        6578* 6584* 6590* 6592* 6597* 6602* 6698* 6699* 6700* 6701* 6703  6706  6714*
                        6715* 6716* 6717* 6719  6721  6730* 6732* 6737  6739  6747* 6748* 6749* 6751*
                        6752* 6754  6762* 6766* 6770* 6771* 6775* 6781* 6785* 6787* 6788* 6791* 6797*
                        6804* 6937* 6939* 6941* 6943* 6945* 6968* 6970* 6971* 6974* 6979* 6990* 6999*
                        7001* 7002* 7018* 7020* 7021* 7024* 7029* 7040* 7049* 7050  7074* 7076* 7082
                        7104* 7106* 7111  7140* 7142* 7144  7156* 7158* 7160  7169* 7171* 7173  7190*
                        7196* 7200* 7229* 7497  7501  7502  7503
ASWREG= 000000          5664
ATESTN= 000000          5664
AUNIT = 000000          5664
AUSWR = 000000          5664
AVECT1= 000440          5655# 5664  5733  5734  5735  5736
AVECT2= 000000          5664
BADUNT  001452          5745# 5876* 7247  7327*
BASEXC  015024          5811  7255#
BIT0  = 000001          5652# 5875  6007  6119  6144  6303  6330  6374  6403  6409  6426  6472  6474
                        6476  6478  6480  6482  6490  6571  6707  6722  6747  6751  6752  6771  6788
BIT00 = 000001          5652# 6281
BIT01 = 000002          5652#
BIT02 = 000004          5652#
BIT03 = 000010          5652#
BIT04 = 000020          5652#
BIT05 = 000040          5652#
BIT06 = 000100          5652#
BIT07 = 000200          5652#
BIT08 = 000400          5652# 6716  6717  7329
BIT09 = 001000          5652# 7327  7329
BIT1  = 000002          5652# 6006  6192  6337  6368
BIT10 = 002000          5652# 6192  6228  6266  6271  6272  6722  7146  7161  7174  7327
BIT11 = 004000          5652# 6000  6192  6228  6770  6787  6937  6939  6941  6943  6945  7329
BIT12 = 010000          5652# 6192  6228  6703  6707  6719  6722  6737  6756
BIT13 = 020000          5652# 5999  6279  6288  6368  7190  7327
BIT14 = 040000          5652# 5998  6571  6592  7329
BIT15 = 100000          5652# 5765  5867  6256  6257  6345  6707  6751  6990  7002  7040  7146  7161
                        7174  7196
BIT2  = 000004          5652# 6005  6248
BIT3  = 000010          5652# 6004  6388  6771  6788
BIT4  = 000020          5652# 6003  6303  6330  6374  6403
BIT5  = 000040          5652# 6002  6303  6330  6374  6403
BIT6  = 000100          5652# 5799  6001  6175  6213  6482  6483  7218  7238
BIT7  = 000200          5652#
BIT8  = 000400          5652# 6123  6148  6265  6304  6306  6332  6376  6405  6424  6472  6474  6476
                        6478  6480  6482  6492  6498  6549  6592  6775  6791  6797  6937  6939  6941
                        6943  6945  6974  7001  7024
BIT9  = 001000          5652# 6123  6148  6234  6280  6306  6345  6472  6474  6476  6478  6480  6482
                        6498  6572  6700  6701  6748  6749  6775  6797  6937  6939  6941  6943  6945
                        6979  6990  7002  7029  7040
BPTVEC= 000014          5652#
CKSWR = 104410          7327  7329  7444#
CR    = 000015          5652# 7336
CRLF  = 000200          5652# 5786  7336
```

```
CTRCHA  015336          7291* 7292* 7293  7297  7301#
CTRLCG  015260          7250  7288# 7327  7329
DDISP = 177570          5652# 5664  5775
DF0     024402          5674  5679  5684  5689  5694  5699  5705  5710  5715  5720  5725  5727  7506#
DH1     022604          5672  5677  5692  5713  5723  7461#
DH12    022743          5697  5703  7465#
DH2     022635          5727  7462#
DH20    022765          5708  7466#
DH26    023025          5718  7467#
DH3     022651          5682  7463#
DH4A    022702          5687  7464#
DISPLA  001142          5664# 5775* 7327* 7329*
DISPRE  000174          5640# 5775
DOT     023565          5805  7480#
DR      001436          5739# 5903  6193  6248  6337  6388
DR2     001440          5740# 5902* 6192* 6228* 6368*
DSWR  = 177570          5652# 5664  5775
DT1     024262          5673  5678  5693  5724  7497#
DT12    024332          5698  5704  7501#
DT2     024276          5727  7498#
DT20    024342          5709  7502#
DT22    024356          5714  7503#
DT26    024372          5719  7504#
DT3     024304          5683  7499#
DT4     024320          5688  7500#
DWARF   001476          5755# 5765* 5770* 5789* 5800* 5814* 5857  5900  6189  6225  6245  6334  6364
                        6385  7066  7097  7122  7181  7220
EMTVEC= 000030          5652# 5775*
EM1     021662          5671  7448#
EM11    022213          5712  7454#
EM12    022250          5696  7455#
EM13    022315          5722  7456#
EM16    022361          5702  7457#
EM2     021730          5676  7449#
EM20    022430          5707  7458#
EM26    022475          5717  7459#
EM3     021772          5681  7450#
EM4     022034          5686  7451#
EM5     022073          5691  7452#
EM6     022134          5727  7453#
ENDOUT  023753          7258  7266  7484#
ENDP    014512          7124  7183  7216#
ERCNT   001466          5751# 5879*
ERRTOT  024027          7241  7487#
ERRVEC= 000004          5652# 5775* 5784* 5785* 5835* 5853* 5854* 5949* 5951* 7329*
EVER    001460          5748# 5803* 5855  5866* 5867* 5869  5871  7227
EXTMSG  014740          7236  7238#
FIXADR  003016          5889# 7231
FOUND1  023760          5859  7485#
FOUND2  024003          5862  7486#
GNS   = ****** U        5786  7070  7071  7072  7101  7102  7128  7138  7139  7155  7168  7186  7187
                        7188  7281  7282  7285  7444
GTSWR = 104407          5786  5819  7299  7444#
HT    = 000011          5652# 7336
IOTRD   021316          5638  5784  5853  7361#
IOTVEC= 000020          5652# 5775*
```

```
KWL     001416          5729#   6482*   6483*
LCNT    001474          5754#
LF    = 000012          5652#   7336
LOGIC   002464          5791    5816    5826    5832#   7251
MASKNM  001450          5744#   5833*   5871*   5875*   7233*   7244    7327
MDEVCT  001470          5752#   5878*
MESGD   024056          7246    7488#
MTEST   002260          5796#   5822
MTEST1  002270          5793    5798#   5820    5828    7272    7296    7331
PIRQ  = 177772          5652#
PIRQVE= 000240          5652#
PRIADR  003002          5804    5874    5887#   7271
PRIME0  023056          5796    7224    7468#
PRIME1  023231          5797    7471#
PRIOR   001434          5737#
PR0   = 000000          5652#
PR1   = 000040          5652#
PR2   = 000100          5652#
PR3   = 000140          5652#
PR4   = 000200          5652#
PR5   = 000240          5652#
PR6   = 000300          5652#
PR7   = 000340          5652#
PS    = 177776          5652#
PSW   = 177776          5652#
PWRVEC= 000024          5652#   5775*   7339*
RDCHR = 104411          7129    7283    7331    7444#
RDLIN = 104412          5806    7332    7444#
RDOCT = 104413          7259    7267    7444#
RESTRT= 001550          5649    5772#
RESVEC= 000010          5652#
ROTATE  001462          5749#   5880*   5881
RUNIT   002462          5807*   5808*   5809    5812    5817    5821    5823    5829#
STACK = 001100          5652#   5775
START = 001530          5648    5767#
STKLMT= 177774          5652#
SWR     001140          5664#   5775*   5786    5841    6482    7327    7329    7331*   7339*
SWREG   000176          5641#   5775    5786    7331
SW0   = 000001          5652#
SW00  = 000001          5652#
SW01  = 000002          5652#
SW02  = 000004          5652#
SW03  = 000010          5652#
SW04  = 000020          5652#
SW05  = 000040          5652#
SW06  = 000100          5652#
SW07  = 000200          5652#
SW08  = 000400          5652#
SW09  = 001000          5652#
SW1   = 000002          5652#
SW10  = 002000          5652#   6482
SW11  = 004000          5652#
SW12  = 010000          5652#   5841
SW13  = 020000          5652#
SW14  = 040000          5652#
SW15  = 100000          5652#
```

```
SW2    = 000004            5652#
SW3    = 000010            5652#
SW4    = 000020            5652#
SW5    = 000040            5652#
SW6    = 000100            5652#
SW7    = 000200            5652#
SW8    = 000400            5652#
SW9    = 001000            5652#
TBITVE= 000014             5652#
TEMP1   001502             5758#  5763*  5769*  5773*  5792
TESTER= 001506             5650   5762#
TKVEC  = 000060            5652#  7331*
TPVEC  = 000064            5652#
TRAPVE= 000034             5652#  5775*
TRFRO   021574             7372*  7442#  7500
TRTO    021572             7361*  7362*  7364   7424   7432   7441#  7500
TRTVEC= 000014             5652#
TSCLC   001442             5741#  7191*  7193*  7194   7197*  7198   7201*
TSCLD   001444             5742#  7192*
TSTCNT  001472             5753#  5764*  5768*  5790*  5815*  5825*  5843
TST1    003106             5883   5899#  7234
TST10   003714             6002#
TST11   004012             6003#
TST12   004110             6004#
TST13   004206             6005#
TST14   004304             6006#
TST15   004402             6007#
TST16   004500             6040#
TST17   004570             6042#
TST2    003156             5901   5907   5949#
TST20   004660             6057#
TST21   004732             6083#
TST22   005014             6112#
TST23   005142             6137#
TST24   005270             6169#
TST25   005366             6187   6190   6191   6194   6207#
TST26   005450             6226   6227   6230#
TST27   005550             6244   6247   6254   6258   6263#
TST3    003252             5951#
TST30   005616             6273   6278#
TST31   005664             6299#
TST32   006002             6323#
TST33   006216             6336   6343   6357   6362#
TST34   006330             6366   6384   6386   6387   6389   6397#
TST35   006372             6418#
TST36   006440             6472#
TST37   006570             6474#
TST4    003324             5998#
TST40   006720             6476#
TST41   007050             6478#
TST42   007200             6480#
TST43   007330             6482#
TST44   007504             6482   6485#
TST45   007646             6510#
TST46   007714             6541#
TST47   010040             6565#
```

```
TST5     003422            5999#
TST50    010156            6587#
TST51    010300            6697#
TST52    010354            6713#
TST53    010430            6720    6728#
TST54    010514            6746#
TST55    010606            6764#
TST56    010742            6783#
TST57    011074            6937#
TST6     003520            6000#
TST60    011230            6937    6939#
TST61    011364            6939    6941#
TST62    011520            6941    6943#
TST63    011654            6943    6945#
TST64    012010            6945    6966#
TST65    012266            7003    7016#
TST66    012476            7038    7047    7051    7065#
TST67    013016            7067    7084    7096#
TST7     003616            6001#
TST70    013254            7098    7113    7118#
TST71    013626            7147    7151#
TST72    013744            7162    7165#
TST73    014120            7177    7180#
TST74    014502            7209    7213#
TYPBN  = 104406            7248    7444#
TYPDS  = 104405            5861    7236    7243    7328    7416    7444#
TYPE   = 104401            5786    5796    5797    5805    5827    5859    5862    7070    7071    7072    7101    7102    7128
                           7138    7139    7155    7168    7186    7187    7188    7224    7236    7241    7246    7249    7255
                           7258    7263    7266    7281    7282    7285    7309    7310    7312    7327    7328    7331    7336
                           7339    7413    7419    7423    7427    7444#
TYPOC  = 104402            7257    7265    7328    7331    7444#
TYPON  = 104404            7444#
TYPOS  = 104403            7421    7425    7444#
UNITBD   001504            5759#   7327*   7497    7498    7499    7500    7501    7502    7503    7504
UTEST    001464            5750#   5881*
VADDR    001454            5746#   5837    7229
VECOUT   023713            7263    7483#
VECTP    001426            5734#   5890*   5891*   6563    7428    7430*   7434*   7435*
VECT1    001424            5733#   5888*   5889*   5890    5892    6550*   6563*   7230*   7420    7428*   7432*   7433*   7434
                           7436
VECT2    001430            5735#   5892*   5893*   5894    6570*   6585*   6591*   6603*   7429*   7436*   7437*   7438
VECT2P   001432            5736#   5894*   5895*   6585    6603    7429    7431*   7438*   7439*
VTMSG    024073            7413    7489#
VTMSG1   024151            7423    7491#
VTMSG2   024201            7427    7492#
VTMSG3   024121            7419    7490#
VVECTR   001456            5747#   7230
$APTHD   001000            5663#
$ASTAT=  ****** U          7337
$ATYC    020716            7337#
$ATY1    020672            7337#
$ATY3    020700            7336    7337#
$ATY4    020710            7327    7337#
$AUTOB   001134            5664#   5786*   5787    7331
$BASE    001250            5664#   5832    5887    7256    7262*
$BDADR   001122            5664#
```

```
$BDDAT  001126  5664# 5832* 5836  5837* 5903* 5904* 5906  5998* 5999* 6000* 6001* 6002* 6003*
                6004* 6005* 6006* 6007* 6040* 6042* 6065* 6067  6070  6099* 6100  6102  6123*
                6125  6148* 6150  6177* 6215* 6306* 6345* 6348  6472* 6474* 6476* 6478* 6480*
                6482* 6498* 6706* 6721* 6739* 6754* 6756  6775* 6797* 6937* 6939* 6941* 6943*
                6945* 6981* 6982  6990* 6991  7002* 7031* 7032  7040* 7050* 7082* 7083  7111*
                7112  7144* 7146  7160* 7161  7173* 7175* 7176  7202* 7205  7497  7499  7502
                7503
$BELL   001164  5664# 7327  7331
$BIN    015640  7310#*
$CDW1   001254  5664#
$CHARC  020666  7336#*
$CKSWR  017316  7331#  7444
$CMTAG  001100  5664#  5775
$CM3  = 000000  5664#
$CNTLC  020244  7331#
$CNTLG  020256  7331#
$CNTLU  020251  7331#
$CPUOP  001222  5664#
$CRLF   001171  5664#  7327  7328  7331  7336
$DBLK   016056  7312#
$DEVCT  001204  5664#  5783*
$DEVM   001252  5664#
$DOAGN  014714  7236#
$DTBL   016046  7312#
$ENDAD  014704  5661  7236#
$ENDCT  014652  7236#
$ENDMG  014723  7236#
$ENULL  014720  7236#  7249
$ENV    001214  5664#  5786  5839  7327  7336  7337
$ENVM   001215  5664#  5775  7336  7337
$EOP    014616  5852  5865  5949  5951  7228  7236#
$EOPCT  014644  7236#
$ERFLG  001103  5664#  7327* 7329*
$ERMAX  001115  5664#  5775* 7329*
$ERROR  016066  5775  7327#
$ERRPC  001116  5664#  7327* 7328  7497  7498  7499  7500  7501  7502  7503  7504
$ERRTB  001256  5664#  7328
$ERRTY  016322  7327  7328#
$ERTTL  001112  5664#  5802* 7239  7242  7327*
$ESCAP  001162  5664#  5775* 7327  7329*
$ETABL  001214  5664#
$ETEND  001256  5663  5664#
$FATAL  001176  5664#  7337*
$FFLG   021136  7337#*
$FILLC  001156  5664#  7336
$FILLS  001155  5664#  7336
$GDADR  001120  5664#
$GDDAT  001124  5664#  5905* 5906  5998* 5999* 6000* 6001* 6002* 6003* 6004* 6005* 6006* 6007*
                6040* 6042* 6067* 6068* 6100* 6101* 6121* 6125  6146* 6150  6171* 6209* 6327*
                6346* 6347  6348  6358  6359  6503* 6707* 6722* 6740* 6755* 6768* 6980* 6982
                6991  7000* 7030* 7032  7039* 7052* 7081* 7083  7110* 7112  7145* 7174* 7176
                7203* 7204  7497  7499  7502
$GET42  014674  7236#
$GTSWR  017406  7331#  7444
$HD   = 000001  5616
$HIBTS  001000  5663#
```

```
$HIOCT  020406          7332#*
$ICNT   001104          5664#   7329*
$ILLUP  021300          7339#
$INTAG  001135          5664#   7331*
$ITEMB  001114          5664#   7327*   7328
$LF     001172          5664#   7327    7331    7336
$LFLG   021135          7337#*
$LPADR  001106          5664#   5775*   5949*   7069*   7099*   7153*   7167*   7185*   7329*   7440
$LPERR  001110          5664#   5775*   5949*   6328*   7068*   7100*   7152*   7166*   7184*   7327    7329*
$MADR1  001226          5664#
$MADR2  001232          5664#
$MADR3  001236          5664#
$MADR4  001242          5664#
$MAIL   001174          5663    5664#   5775    5786    7327    7329    7336
$MAMS1  001224          5664#
$MAMS2  001230          5664#
$MAMS3  001234          5664#
$MAMS4  001240          5664#
$MBADR  001002          5663#
$MFLG   021134          7337#*
$MNEW   020274          7331#
$MSGAD  001210          5664#   7337*
$MSGLG  001212          5664#   7337*
$MSGTY  001174          5664#   7337*
$MSWR   020263          7331#
$MTYP1  001225          5664#
$MTYP2  001231          5664#
$MTYP3  001235          5664#
$MTYP4  001241          5664#
$MXCNT  016760          7329#
$NULL   001154          5664#   7336
$NWTST= 000001          5899#   5949#   5951#   5998#   5999#   6000#   6001#   6002#   6003#   6004#   6005#   6006#   6007#
                        6040#   6042#   6057#   6083#   6112#   6137#   6169#   6207#   6230#   6263#   6278#   6299#   6323#
                        6362#   6397#   6418#   6472#   6474#   6476#   6478#   6480#   6482#   6485#   6510#   6541#   6565#
                        6587#   6697#   6713#   6728#   6746#   6764#   6783#   6937#   6939#   6941#   6943#   6945#   6966#
                        7016#   7065#   7096#   7118#   7151#   7165#   7180#   7213#
$OCNT   015562          7309#*
$OMODE  015564          7309#*
$OVER   016744          7329#
$PASS   001202          5664#   5775*   5801*   6356    7120    7222    7236*   7329    7411
$PASTM  001006          5663#
$POWER  021306          7339#
$PWRDN  021140          5775    7339#
$PWRMG  021274          7339#
$PWRUP  021212          7339#
$QUES   001170          5664#   5827    7327    7331    7336
$RDCHR  017660          7331#   7444
$RDDEC= ******  U       7444
$RDLIN  017750          7331#   7444
$RDOCT  020306          7332#   7444
$RDSZ = 000040          7331#
$RTNAD  014716          7236#
$R2A  = ******  U       7444
$SAVRE= ******  U       7444
$SAVR6  021304          7339#*
$SCOPE  016476          5775    7329#
```

```
$SETUP= 000117  5761#  5775  5786  7236  7327  7329  7331
$STUP = 177777  5761#
$SVLAD  016710  7329#
$SVPC = 000214  5661#
$SWR  = 167400  5558#  5616  5618  5658#  5664  5775  5899  5949  5951  5998  5999  6000  6001
                6002  6003  6004  6005  6006  6007  6040  6042  6057  6083  6112  6137  6169
                6207  6230  6263  6278  6299  6323  6362  6397  6418  6472  6474  6476  6478
                6480  6482  6485  6510  6541  6565  6587  6697  6713  6728  6746  6764  6783
                6937  6939  6941  6943  6945  6966  7016  7065  7096  7118  7151  7165  7180
                7213  7236  7327  7329  7339
$SWREG  001216  5664#  5775
$SWRMK= 000000  5618  7329
$TFSTN  001200  5664#  7329*
$TIMES  001160  5664#  5775*  5949*  5998*  5999*  6000*  6001*  6002*  6003*  6004*  6005*  6006*  6007*
                6057*  6083*  6112*  6137*  6169*  6207*  6299*  6323*  6472*  6474*  6476*  6478*  6480*
                6482*  6485*  6510*  6541*  6565*  6587*  6728*  6764*  6937*  6939*  6941*  6943*  6945*
                6966*  7016*  7065*  7096*  7180*  7213*  7236*  7329*
$TKB    001146  5664#  7280  7291  7331
$TKCNT  016762  7331#*
$TKINT  017030  5781  7331#
$TKQEN= 017030  7331#
$TKQIN  016764  7331#*
$TKQOU  016766  7331#*
$TKQSR  016770  7331#
$TKS    001144  5664#  5799*  6175*  6213*  7218*  7238*  7289  7331*
$TKSRV  017100  7331#
$TMPO   001446  5743#  5949*  5951*  6347*  6937*  6939*  6941*  6943*  6945*  7503  7504
$TN   = 000075  5616#  5659#  5899#  5901  5907  5949#  5951#  5998#  5999#  6000#  6001#  6002#  6003#
                6004#  6005#  6006#  6007#  6040#  6042#  6057#  6083#  6112#  6137#  6169#  6187  6190
                6191  6194  6207#  6226  6227  6230#  6244  6247  6254  6258  6263#  6273  6278#
                6299#  6323#  6336  6343  6357  6362#  6366  6384  6386  6387  6389  6397#  6418#
                6472#  6474#  6476#  6478#  6480#  6482#  6485#  6510#  6541#  6565#  6587#  6697#  6713#
                6720  6728#  6746#  6764#  6783#  6937#  6939#  6941#  6943#  6945#  6966#  7003  7016#
                7038  7047  7051  7065#  7067  7084  7096#  7098  7113  7118#  7147  7151#  7162
                7165#  7177  7180#  7209  7213#
$TPB    001152  5664#  7336*
$TPFLG  001157  5664#  7336
$TPS    001150  5664#  7336
$TRAP   021576  5775  7444#
$TRAP2  021620  7444#
$TRP  = 000014  7444#
$TRPAD  021632  7444#
$TSTM   001004  5663#
$TSTNM  001102  5664#  7232*  7236*  7327  7329*  7375
$TTYIN  020204  7331#
$TYPBN  015566  7310#  7444
$TYPDS  015642  7312#  7444
$TYPE   020410  5777*  5778*  5779*  5780*  7336#  7337  7444
$TYPEC  020622  7331  7336#
$TYPEX  020670  7336#
$TYPOC  015364  7309#  7444
$TYPON  015400  7309#  7444
$TYPOS  015340  7309#  7444
$UNIT   001206  5664#  5834*  5838*  5843  5847  5860  5863  5866  5869  5873*  7226*  7227  7414
$UNITM  001010  5663#
$USWR   001220  5664#
```

```
$VECT1  001244            5664#   5888    7264    7270*
$VECT2  001246            5664#
$XTSTR  016514            7329#
$$GET4= 000000            7236#
$OFILL  015563            7309#*
$40CAT= ****** U          7327    7329
.       = 024411          5622#   5633    5634#   5637#   5639#   5642#   5647#   5661#   5662#   5663#   5664#   5762    5767
                          5772    5775    5786#   7072#   7102#   7139#   7168#   7186#   7187#   7188#   7236    7281#   7282#
                          7312#   7327    7328#   7329    7331#   7336    7337#   7339    7495#
.$ASTA= ****** U          7337
.$X     = 001000          5663#
```

```
        6697    6713    6728    6746    6764    6783    6937    6939    6941    6943    6945    6966    7016    7065    7096
        7118    7151    7165    7180    7213    7236    7309    7310    7312    7327    7328    7329    7331    7332    7336
        7337    7339    7444
SWRSU   1416#   5652#   5775#
TRMTRP  7444#
TYPBIN  2039#   5652#
TYPDEC  2009#   5652#   7236    7328
TYPNAM  1779#   5652#   5786
TYPNUM  1976#   5652#
TYPOCS  1929#   5555#   5652#
TYPOCT  1892#   5652#   7328    7331
TYPTXT  1846#   5652#   7070    7071    7072    7101    7102    7128    7138    7139    7155    7168    7186    7187    7188
        7281    7282    7285
ZP21    6048#   6057    6083
ZP25    6162#   6169
ZP25A   6199#   6207
ZZZ     7057#   7065    7089#   7096
ZZ1     5952#   5998    5999    6000    6001    6002    6003    6004    6005    6006    6007    6314#   6323    6472#   6474#
        6476#   6478#   6480#   6482#   6947#   6966    7008#   7016
$$CMRE  5664#
$$CMTM  5664#
$$ESCA  1667#   5652#
$$NEWT  1621#   5652#   5899    5949    5951    5998    5999    6000    6001    6002    6003    6004    6005    6006    6007
        6040    6042    6057    6083    6112    6137    6169    6207    6230    6263    6278    6299    6323    6362    6397
        6418    6472    6474    6476    6478    6480    6482    6485    6510    6541    6565    6587    6697    6713    6728
        6746    6764    6783    6937    6939    6941    6943    6945    6966    7016    7065    7096    7118    7151    7165
        7180    7213
$$SET   7444#
$$SETM  5775#
$$SKIP  1701#   5652#   5901    5907    6187    6190    6191    6194    6226    6227    6244    6247    6254    6258    6273
        6336    6343    6357    6366    6384    6386    6387    6389    6482    6720    6937    6939    6941    6943    6945
        7003    7038    7047    7051    7067    7084    7098    7113    7147    7162    7177    7209
.EQUAT   176#   5555#   5652
.HEADE    50#   5554#   5616
.KT11    319#
.SETTR  5554#
.SETUP  1180#   5554#   5761
.SWRHI    92#   5555#   5618
.SWRLO  5618#
.TRMTR  5554#
.$ACT1  4961#   5557#   5661
.$APTB  5005#   5664#
.$APTH  5261#   5557#   5663
.$APTY  5436#   5557#   7337
.$ASTA  5307#
.$CATC   905#   5555#
.$CMTA  1016#   5555#   5664
.$DB2D  4591#
.$DB20  4714#
.$DIV   4494#
.$EOP   2162#   5556#   7236
.$ERRO  2643#   5556#   7327
.$ERRT  2838#   5556#   7328
.$MULT  4431#
.$POWE  4143#   5555#   7339
.$RAND  4218#
```

```
.$RDDE   3814#
.$RDOC   3723#   5554#   7332
.$READ   3328#   5556#   7331
.$R2AZ   4858#
.$SAVE   3889#
.$SB2D   4675#
.$SB2O   4776#
.$SCOP   2397#   5556#   7329
.$SIZE   4271#
.$SUPR   4814#
.$TRAP   3991#   5554#   7444
.$TYPB   3221#   5554#   7310
.$TYPD   3144#   5556#   7312
.$TYPE   2925#   5556#   7336
.$TYPO   3048#   5555#   7309
.$40CA    944#
.1170     498#


. ABS.  024411     000


 ERRORS DETECTED:  0

 CVMNCB,CVMNCB/CRF=CVMNCB.SML,CVMNCB.P11
 RUN-TIME: 20 26 1 SECONDS
 RUN-TIME RATIO: 435/47-9.0
 CORE USED:  36K  (71 PAGES)
```