

LIST LOC,SEQ,BIN
PEM

B 1

SEQ 0001

PRODUCT CODE: AC-F408A-MC

PRODUCT NAME: CZKWLA0 KW11-C DIAGNOSTIC

DATE: JANUARY 1980

AUTHOR(S): VIJAY ANANDWALA

MAINTAINER: C.S.S. DIAGNOSTIC GROUP

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

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

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

COPYRIGHT: 1980 DIGITAL EQUIPMENT CORP.
MAYNARD MASS.

1 .0 **ABSTRACT:**
 THE KW11-C DIAGNOSTIC PROGRAM PROVIDES A
 SERIES OF TESTS DESIGNED TO VERIFY THE INTEGRITY
 AND OPERABILITY OF THE KW11-C CLOCK. ONE OF THE
 TEST IS VISUAL ONLY, HOWEVER ALL TESTS WILL RUN WITH
 WITHOUT A DISPLAY CONNECTED TO THE SYSTEM.
 THE DIAGNOSTIC MODULE CONSISTS OF 17 TEST
 ELEMENTS PLUS VARIOUS SUPPORTING SUBROUTINES AND
 SEQUENCES.

2 .0 **REQUIREMENTS:**

2 .1 **EQUIPMENT**

PDP 11 PROCESSOR
 CONSOLE DEVICE (LA30, LA36, VT50, VT100, ETC)
 KW11-C CLOCK

DESIRED EQUIPMENT

 KW11-P OR KW11-L CLOCK

2 .2 **PROGRAM STORAGE**

PROGRAM REQUIRES 8 K WORDS OR MORE MEMORY

2 .3 **SOFTWARE**

5 .0 **ABSOLUTE LOADER OR OTHER INPUT MEDIUM
 SWITCH REGISTER ASSIGNMENTS:**

SW08(1)=LOOP ON TEST IN SWR<7:0>
 SW08(0)=DO ALL TESTS IN SEQUENCE

SW09(1)=LOOP ON ERROR
 SW09(0)=CONTINUE ON ERROR

SW10(1)=BELL ON ERROR
 SW10(0)=NO BELL ON ERROR

SW11(1)=INHIBIT ITERATIONS
 SW11(0)=ALLOW ITERATIONS

SW13(1)=INHIBIT ERROR TYPEOUTS
 SW13(0)=ALLOW ERROR TYPEOUTS

SW14(1)=LOOP ON TEST
 SW14(0)=DON'T LOOP ON TEST

SW15(1)=HALT ON ERROR
SW15(0)=CONTINUE ON ERROR

D 1

SEQ 0003

NOTE:

COMPUTERS WITHOUT A HARDWARE SWITCH REGISTER HAVE A SOFTWARE SWITCH REGISTER LOCATED IN MEMORY AT LOCATION 176 CALLED SWREG. THIS LOCATION CAN BE CHANGED EITHER MANUALLY OR BY TYPING THE CNTRL+G KEYS TOGETHER THEN RESPONDING TO THE TERMINAL DIALOGUE.
LOADING PROCEDURE:

4 .0

THE STANDARD PROCEDURE FOR LOADING ABSOLUTE BINARY TAPES SHOULD BE USED IF THE PROGRAM RESIDES ON A MASS STORAGE DEVICE SUCH AS MAGTAPE OR DISK REFER TO THE XXDP USER'S GUIDE.

5 .0

STARTING PROCEDURE:

5 .1

STARTING ADDRESSES

INITIAL STARTING ADDRESS IS 200
RESTART ADDRESS IS 200

5 .2

STARTING SEQUENCE

LOAD PROGRAM ACCORDING TO LOADING PROCEDURE
LOAD ADDRESS 200 AND START COMPUTER.

FIRST TIME PROGRAM PRINTS THE NAME OF THE PROGRAM, AND ALSO PRINTS HELP MESSAGE WHICH IS SELF EXPLANATORY. THE MONITOR PROMPTS WITH(?) FOR OPERATOR'S INPUT. CHARACTER TYPED MUST BE TERMINATED BY 'CR'. HELP MESSAGE PRINTS AS FOLLOWS

SELECT FUNCTION BELOW , TYPE CHARACTER CARRIAGE RETURN.
'T' TEST CLOCK (CONTROL C TO EXIT)
'S' TO SET CLOCK
'R' TO READ CLOCK
'M' TO MODIFY/READ ADDRESS AND VECTOR
'A' ACCURACY OF CLOCK
'G' CHANGE SOFTWARE SWITCH REGISTER
'E' EXIT HALTS CPU
'H' HELP PRINTS THIS MESSAGE.

WHEN MONITOR PROMPTS WITH(?) OPERATOR CAN TYPE ANY ONE OF ABOVE CHARACTER FOLLOWED BY A 'CR'.

E 1

IF YOU WANT TO TEST THE CLOCK TYPE 'T' 'CR'
PROGRAM START TESTING THE CLOCK. TO EXIT FROM ANY TEST TYPE
CONTROL C. WHEN PROGRAM RECOGNIZES CONTROL C CHARACTER,
PROGRAM PROMPTS WITH '?' TO INDICATE NOW IN MONITOR MODE
FROM WHICH YOU CAN AGAIN GO BACK TO TEST THE CLOCK OR
CAN SELECT THE FUNCTIONS DESCRIBED ABOVE.

HOW TO SET CLOCK

FIRST MAKE SURE YOU ARE IN A MONITOR MODE (?).
TYPE 'S' FOLLOWED BY A 'CR' TO SET CLOCK.

WHEN SETTING CLOCK, YOU MUST ENTER VALUES IN ALL FIELD IN
PROPER FORMAT. LETTERS IN FIELD INDICATES NUMBER OF DIGITS
TO BE TYPED FOR THE FIELD. TYPE 'CR' TO TERMINATE ENTERING
VALUE.

HOW TO READ CLOCK

MAKE SURE YOU ARE IN A MONITOR MODE(?).
TYPE 'R' AND 'CR', PROGRAM PRINTS YEAR, JULIAN DAY, MONTH,
DAY, HOUR, MINUTE, SECOND.

HOW TO MODIFY/READ ADDRESS, VECTOR AND BUS REQUEST LEVEL.

MAKE SURE YOU ARE IN A MONITOR MODE(?). TYPE
'M' AND 'CR' IF YOU WISH TO READ/MODIFY ADDRESSES. FIRST PROGRAM
PRINTS CURRENT DEVICE ADDRESS AND WAITS FOR OPERATOR'S RESPOND.
TYPE NEW ADDRESS IF YOU WISH TO CHANGE THE ADDRESS, OR JUST
'CR' FOR NO CHANGE IN CURRENT ADDRESS. NOW PROGRAM PRINTS THE
CURRENT VECTOR ADDRESS AND SIMILARLY WAITS FOR OPERATOR'S
RESPOND. DO AS ABOVE IF YOU WISH TO CHANGE THE ADDRESS OR IF
YOU DON'T. LAST PROGRAM PRINTS BUS REQUEST LEVEL. AFTER RESPON
DING TO CURRENT PRINT OUT YOU ARE BACK INTO MONITOR MODE.

NORMAL ADDRESSES ARE AS FOLLOWS
DEVICE ADDRESS = 760200
VECTOR ADDRESS = 300
BUS REQUEST LEVEL = 6

HOW TO TEST ACCURACY OF CLOCK

WHEN YOU ARE IN A MONITOR MODE TYPING 'A' FOLLOWED BY
'CR' PROGRAM GOES TO THE ROUTINE FOR TESTING ACCURACY
OF CLOCK.

FOLLOWING QUESTIONS WILL BE ASKED:

L-CLK (L) N ?
P-CLK (L) N ?

'TYPE TWO CHARACTERS AT 60 SEC APART'

ANSWER FIRST TWO QUESTIONS WITH <CR> IF ANSWER
IS AFFIRMATIVE AND WITH <N><CR> FOR NEGATIVE ANSWER.

F 1

IF BOTH ANSWER ARE NEGATIVE PROGRAM WILL ASK TO TYPE TWO CHARACTERS AT 60 SECONDS APART. PLEASE BE VERY ACCURATE IN TYPING TWO CHARACTERS AT EXACTLY 60 SECONDS APART. THE PERIOD BETWEEN TWO CHARACTERS TYPED WILL BE USED TO DETERMINE THE TIME BASE FOR THE KW11-C.

THE ERROR MESSAGE WILL BE PRINTED IF CLOCK IS NOT ACCURATE. THEN THE PROGRAM GOES TO MONITOR MODE(?).

IF THE CLOCK IS ACCURATE THERE WON'T BE ANY MESSAGE AND PROGRAM PROMPT WITH <?> INDICATE IN MONITOR MODE.

HOW TO CHANGE SOFTWARE SWITCH REGISTER

WHEN YOU ARE IN A MONITOR MODE(?), TYPE 'G' 'CR' PROGRAM PRINTS OLD CONTENT OF SOFTWARE SWITCH REGISTER AND WAITS FOR NEW VALUE. IF YOU DO NOT WANT TO CHANGE THE OLD VALUE IN SOFTWARE SWITCH REGISTER JUST TYPE 'CR'.
WHEN YOU ARE TESTING THE CLOCK 'T', TYPING CONTROL 'G' PRINTS OLD CONTENT OF SOFTWARE SWITCH REGISTER AND WAITS FOR NEW VALUE AS ABOVE. SO USE <CONTROL G> WHEN TESTING CLOCK.

HELP MESSAGE

IN MONITOR MODE TYPING 'H' 'CR' PRINTS

SELECT FUNCTION BELOW, TYPE CHARACTER CARRIAGE RETURN.
'T' TEST CLOCK (CONOTL C TO EXIT)
'S' TO SET CLOCK
'R' TO READ CLOCK
'M' TO MODIFY/READ ADDRESS AND VECTOR
'A' ACCURACY OF CLOCK
'G' CHECK/MODIFY SOFTWARE SWITCH REGISTER
'E' EXIT HALTS CPU
'H' HELP PRINTS THIS MESSAGE.

5 .0 PRELIMINARY OPERATIONS:

5 .1 DEVICE ADDRESSES AND VECTOR ADDRESSES

DEFAULT:
DEVICE ADDRESSES: 760200
760202
760204
CSR: 760206

VECTOR(0): 300
VECTOR(2): 304

BR: 6

5.3 PRELIMINARY PROGRAMS NEEDED

6.0 NONE
OPERATIONAL PROCEDURES:

7.0 NORMAL OPERATION IS WITH ALL SWITCHES ON 0.
ERRORS:

EACH ERROR SIGNATURE CONSISTS OF ONE OR MORE LINES OF TEXT, DESCRIBING THE ERROR AND ALSO INCLUDES EXPECTED VS RECEIVED DATA, CONTENT OF CSR AT TIME OF ERROR TEST NUMBER WHERE APPLICABLE. FOLLOWINGS ARE EXAMPLES OF ERROR PRINT OUT

EX1: 'READY BIT FAIL TO SET IN CSR'

ERROR	TEST#	CSR
PC 5432	7	00000

WHERE
 ERROR PC GIVES THE ADDRESS WHERE ERROR OCCURED.
 TEST# TELLS THE TEST NUMBER WHERE ERROR OCCURED.
 CSR TELLS THE CONTENT OF CSR AT TIME OF ERROR.

EX2: 'READ VALUE NOT MATCH WITH EXPECTED'

ERROR	TEST#	GOOD	BAD	CSR
PC 5432	21	00000	00073	00200

WHERE
 GOOD IS EXPECTED VALUE
 BAD IS ACTUAL VALUE READ.

OTHER FIELDS ARE SAME AS ABOVE.

EX3: 'HOLD BIT FAIL TO RESET'

ERROR	TEST#	PRESEC	CURSEC	CSR
PC 6010	11	20	25	000001

WHERE

H 1

SEQ 0007

PRESEC IS THE VALUE OF SECOND BEFORE HOLD BIT SET
CURSEC IS THE VALUE OF SECOND AT TIME OF ERROR.

EXECUTION TIME:

EXECUTION TIME VARY SOMEWHAT, DEPENDING
ON CPU TYPE. THE FOLLOWING ARE EXECUTION TIME OBSERVED ON
PDP11/34.

QUICK PASS: 6MIN, 20SEC
SUBSEQUENT PASS: 14MIN, 30SEC

-

```
11      167400      $SWR=167400
12      000001      $TN-1
13
14      .SBTTL BASIC DEFINITIONS
(1)
(1)      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)      001100      STACK= 1100
(1)      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
(1)      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)      ;*MISCELLANEOUS DEFINITIONS
(1)      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
(1)      000012      LF= 12      ;;CODE FOR LINE FEED
(1)      000015      CR= 15      ;;CODE FOR CARRIAGE RETURN
(1)      000200      CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1)      177776      PS= 177776   ;;PROCESSOR STATUS WORD
(1)      .EQUIV PS,PSW
(1)      177774      STKLM= 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= 00      ;;GENERAL REGISTER
(1)      000001      R1= 01      ;;GENERAL REGISTER
(1)      000002      R2= 02      ;;GENERAL REGISTER
(1)      000003      R3= 03      ;;GENERAL REGISTER
(1)      000004      R4= 04      ;;GENERAL REGISTER
(1)      000005      R5= 05      ;;GENERAL REGISTER
(1)      000006      R6= 06      ;;GENERAL REGISTER
(1)      000007      R7= 07      ;;GENERAL REGISTER
(1)      000006      SP= 06      ;;STACK POINTER
(1)      000007      PC= 07      ;;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
15 160006 ABASE=160006 ;:BASE CD BUS ADDRESS EQUATE
16 160300 AVECT1=160300 ;:BASE CD PRIORITY & VECTOR ADDRESS EQUATE
17 177546 LKS=177546 ;:CONTROL AND STATUS REG OF L.CLOCK
18
19 172540 PKCSR=172540 ;:CONTROL AND STATUS REG OF P.CLOCK
20 172542 PKBUF=172542 ;:BUFFER OF P.CLOCK
21 172544 PKCNT=172544 ;:COUNTER
22 001124 GOOD=$GDDAT
23 001126 BAD=$BDDAT
24 000001 REQ=BIT0
25 001122 BADA=$BDADR
26 .SBTTL MEMORY MANAGEMENT DEFINITIONS
(1)
(1) ;*KT11 VECTOR ADDRESS
(1)
(1) 000250 MMVEC= 250
(1)
(1) ;*KT11 STATUS REGISTER ADDRESSES
(1)
(1) 177572 SR0= 177572
(1) 177574 SR1= 177574
(1) 177576 SR2= 177576
(1) 172516 SR3= 172516
(1)
(1) ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
(1)
(1) 172300 KIPDR0= 172300
(1) 172302 KIPDR1= 172302
(1) 172304 KIPDR2= 172304
(1) 172306 KIPDR3= 172306
(1) 172310 KIPDR4= 172310
(1) 172312 KIPDR5= 172312
(1) 172314 KIPDR6= 172314
(1) 172316 KIPDR7= 172316
(1)
(1) ;*KERNEL 'I' PAGE ADDRESS REGISTERS
(1)
(1) 172340 KIPAR0= 172340
(1) 172342 KIPAR1= 172342
(1) 172344 KIPAR2= 172344
(1) 172346 KIPAR3= 172346
(1) 172350 KIPAR4= 172350
(1) 172352 KIPAR5= 172352
(1) 172354 KIPAR6= 172354
(1) 172356 KIPAR7= 172356
(1)
(1) .SBTTL TRAP CATCHER
(1)
(1) 000000 .=0
(1) ;*ALL UNUSED LOCATIONS OF THE VECTOR AREA CONTAIN
(1) ;*A '+2, IOT' SEQUENCE TO CATCH AND PROCESS ILLEGAL
(1) ;*TRAPS AND INTERRUPTS THAT MIGHT OCCUR.
(1) ;*THE IOT TRAP WHICH IS TAKEN ON THE ILLEGAL TRAP/INT
(1) ;*TRAPS TO THE $SCOPE ROUTINE WHICH (IF THE RETURN PC IS
(1) ;*LESS THAN 1002) JUMPS TO THE $ERROR ROUTINE.
```

```
(1) ;*THE $ERROR ROUTINE WILL REPORT THE ERROR AS FOLLOWS:
(1) ;* PC=YYYYYY UNEXPECTED TRAP TO XXX
(1) ;*AND RETURN TO THE PROGRAM AT PC=YYYYYY+2
(1) ;*WHERE XXX=LOCATION OF ILLEGAL TRAP
(1) ;* YYYYYY=PC AT TIME OF TRAP
(1) ;*NOTE: IF THE PROCESSOR IS NOT AN 11/05 THE PROGRAM
(1) ;* CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.
(1)
(1) 000000 000000 $40CAT: HALT ;:HALT
(1) 000002 000737 BR -100 ;:BRANCH TO 177700 & TIME OUT (NOT ON
(1) ;:11/05)
(1) 000004 003776 .WORD ROUT1 ;:VECTOR TO STARTING ADDRESS
(1) 000006 000340 .WORD 340 ;:WITH PRIORITY LEVEL 7
(1) 000174 000000 .WORD 0 ;:SOFTWARE DISPLAY REGISTER
(1) 000176 000000 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER
(1) 000200 000137 003776 .SBTTL STARTING ADDRESS(ES)
(1) JMP @ROUT1 ;:GO TO START OF PROGRAM
28
29
30 .SBTTL ACT11 HOOKS
(1)
(2) ;:*****
(1) ;:HOOKS REQUIRED BY ACT11
(1) 000204 $SVPC= ;:SAVE PC
(1) 000046 000046 .=46 ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1) 000052 000052 .=52 ;:2)SET LOC.52 TO ZERO
(1) 000052 000000 .WORD 0 ;:RESTORE PC
(1) 000204 .=$SVPC
(1) 001000 .=1000
31
32 .SBTTL APT PARAMETER BLOCK
(1)
(2) ;:*****
(1) ;:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2) ;:*****
(1) 001000 .SX= ;:SAVE CURRENT LOCATION
(1) 000024 000024 .=24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200 200 ;:FOR APT START UP
(1) 000044 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 001206 $MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000017 $STMT: .WORD 15 ;:RUN TIM OF LONGEST TEST
(1) 001006 000074 $PASTM: .WORD 60. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000000 $UNITM: .WORD 0 ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000052 .WORD $ETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
```

33

(1)

(2)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(3)

(3)

(3)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(2)

(2)

(2)

(3)

(2)

(2)

(2)

(2)

001100

001100

001100 000000

001102 000

001103 000

001104 000000

001106 000000

001110 000000

001112 000000

001114 000

001115 001

001116 000000

001120 000000

001122 000000

001124 000000

001126 000000

001130 000000

001132 000000

001134 000

001135 000

001136 000000

001140 177570

001142 177570

001144 177560

001146 177562

001150 177564

001152 177566

001154 000

001155 002

001156 012

001157 000

001160 000000

001162 000000

001164 000000

001166 000000

001170 000000

001172 000000

001174 000000

001176 177607

001202 077

001203 015

001204 000012

000377

.SBTTL COMMON TAGS

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

.=1100

\$CMTAG: .WORD 0 ::START OF COMMON TAGS

\$STNUM: .BYTE 0 ::CONTAINS THE TEST NUMBER

\$ERFLG: .BYTE 0 ::CONTAINS ERROR FLAG

\$ICNT: .WORD 0 ::CONTAINS SUBTEST ITERATION COUNT

\$LPADR: .WORD 0 ::CONTAINS SCOPE LOOP ADDRESS

\$LPERR: .WORD 0 ::CONTAINS SCOPE RETURN FOR ERRORS

\$ERTTL: .WORD 0 ::CONTAINS TOTAL ERRORS DETECTED

\$ITEMB: .BYTE 0 ::CONTAINS ITEM CONTROL BYTE

\$ERRMAX: .BYTE 1 ::CONTAINS MAX. ERRORS PER TEST

\$ERRPC: .WORD 0 ::CONTAINS PC OF LAST ERROR INSTRUCTION

\$GDADR: .WORD 0 ::CONTAINS ADDRESS OF 'GOOD' DATA

\$BDADR: .WORD 0 ::CONTAINS ADDRESS OF 'BAD' DATA

\$GDDAT: .WORD 0 ::CONTAINS 'GOOD' DATA

\$BDDAT: .WORD 0 ::CONTAINS 'BAD' DATA

.WORD 0 ::RESERVED--NOT TO BE USED

.WORD 0

\$AUTOB: .BYTE 0 ::AUTOMATIC MODE INDICATOR

\$INTAG: .BYTE 0 ::INTERRUPT MODE INDICATOR

.WORD 0

\$SWR: .WORD DSWR ::ADDRESS OF SWITCH REGISTER

\$DISPLAY: .WORD DDISP ::ADDRESS OF DISPLAY REGISTER

\$TKS: 177560 ::TTY KBD STATUS

\$TKB: 177562 ::TTY KBD BUFFER

\$TPS: 177564 ::TTY PRINTER STATUS REG. ADDRESS

\$TPB: 177566 ::TTY PRINTER BUFFER REG. ADDRESS

\$NULL: .BYTE 0 ::CONTAINS NULL CHARACTER FOR FILLS

\$FILLS: .BYTE 2 ::CONTAINS # OF FILLER CHARACTERS REQUIRED

\$FILLC: .BYTE 12 ::INSERT FILL CHARS. AFTER A 'LINE FEED'

\$TPFLG: .BYTE 0 ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)

\$TMP0: .WORD 0 ::USER DEFINED

\$TMP1: .WORD 0 ::USER DEFINED

\$TMP2: .WORD 0 ::USER DEFINED

\$TMP3: .WORD 0 ::USER DEFINED

\$TMP4: .WORD 0 ::USER DEFINED

\$TIMES: 0 ::MAX. NUMBER OF ITERATIONS

\$ESCAPE: 0 ::ESCAPE ON ERROR ADDRESS

\$BELL: .ASCIZ <207><377><377> ::CODE FOR BELL

\$QUES: .ASCII /?/ ::QUESTION MARK

\$CRLF: .ASCIZ <15> ::CARRIAGE RETURN

\$LF: .ASCIZ <12> ::LINE FEED

::*****

.SBTTL APT MAILBOX-ETABLE

::*****

.EVEN
 \$MAIL: ::APT MAILBOX
 \$MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL ::FATAL ERROR NUMBER

(2)	001212	000000	\$TESTN:	.WORD	ATESTN	::TEST NUMBER
(2)	001214	000000	\$PASS:	.WORD	APASS	::PASS COUNT
(2)	001216	000000	\$DEVCT:	.WORD	ADEVCT	::DEVICE COUNT
(2)	001220	000000	\$UNIT:	.WORD	AUNIT	::I/O UNIT NUMBER
(2)	001222	000000	\$MSGAD:	.WORD	AMSGAD	::MESSAGE ADDRESS
(2)	001224	000000	\$MSGLG:	.WORD	AMSGLG	::MESSAGE LENGTH
(2)	001226		\$ETABLE:			::APT ENVIRONMENT TABLE
(2)	001226	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
(2)	001227	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
(2)	001230	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
(2)	001232	000000	\$USWR:	.WORD	AUSWR	::USER SWITCHES
(2)	001234	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)	001236	000	\$MAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001237	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)	001240	000000	\$MADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001242	000	\$MAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001243	000	\$MTYP2:	.BYTE	AMTYP2	::MEM.TYPE,BLK#2
(2)	001244	000000	\$MADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001246	000	\$MAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001247	000	\$MTYP3:	.BYTE	AMTYP3	::MEM.TYPE,BLK#3
(2)	001250	000000	\$MADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001252	000	\$MAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001253	000	\$MTYP4:	.BYTE	AMTYP4	::MEM.TYPE,BLK#4
(2)	001254	000000	\$MADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001256	160300	\$VECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001260	000000	\$VECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001262	160006	\$BASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001264	000000	\$DEVN:	.WORD	ADEVN	::DEVICE MAP
(2)	001266	000000	\$CDW1:	.WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001270	000000	\$CDW2:	.WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
(2)	001272	000000	\$DDW0:	.WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
(2)	001274	000000	\$DDW1:	.WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
(2)	001276	000000	\$DDW2:	.WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
(2)	001300	000000	\$DDW3:	.WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
(2)	001302	000000	\$DDW4:	.WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
(2)	001304	000000	\$DDW5:	.WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
(2)	001306	000000	\$DDW6:	.WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
(2)	001310	000000	\$DDW7:	.WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
(2)	001312	000000	\$DDW8:	.WORD	ADDW8	::DEVICE DESCRIPTOR WORD#8
(2)	001314	000000	\$DDW9:	.WORD	ADDW9	::DEVICE DESCRIPTOR WORD#9
(2)	001316	000000	\$DDW10:	.WORD	ADDW10	::DEVICE DESCRIPTOR WORD#10
(2)	001320	000000	\$DDW11:	.WORD	ADDW11	::DEVICE DESCRIPTOR WORD#11
(2)	001322	000000	\$DDW12:	.WORD	ADDW12	::DEVICE DESCRIPTOR WORD#12
(2)	001324	000000	\$DDW13:	.WORD	ADDW13	::DEVICE DESCRIPTOR WORD#13
(2)	001326	000000	\$DDW14:	.WORD	ADDW14	::DEVICE DESCRIPTOR WORD#14

.MAIN. MACY11 30A(1052) 13-MAR-80 07:40 PAGE 1-6
CZKWL.A.P11 13-MAR-80 07:39 APT MAILBOX-ETABLE

SEQ 0014

(2) 001330 000000
(2)
(2)
(2) 001332
(2)

\$DDW15: .WORD ADDW15 ;;DEVICE DESCRIPTOR WORD#15

\$ETEND:

```
(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) $ERRTB:
(1) 001332 ;ERROR#1 EM1 ::ACTUAL VALUE DIDN'T MATCH WITH EXPECTED
34 ;ERROR#1 DH1
35 001332 024406 DT1
36 001334 027431 000
37 001336 027654 ;ERROR#2 EM2 ::ERROR: READY BIT DIDN'T SET
38 001340 000000 DH2
39 ;ERROR#2 DT2
40 001342 024454 000
41 001344 027517 ;ERROR#3 EM3 ::TIME INTERVAL COUNTER DIDN'T DECREMENT IN ORDER
42 001346 027670 DH1
43 001350 000000 DT1
44 ;ERROR#3 DT2
45 001352 024502 EM4 ::PROBABLE CAUSE OF AN ERROR:BIT#14 WAS
46 001354 027431 DH1 ::NOT SET IN DEVICE REG4
47 001356 027654 DT1
48 001360 000000 000
49 ;ERROR#4 EM4
50 001362 024563 DH2
51 ;ERROR#4 DT2
52 001364 027517 000
53 001366 027670 ;ERROR#5 EM5 ::ERROR:BIT #14 IN CSR FAIL TO RESET
54 001370 000000 DH2
55 ;ERROR#5 DT2
56 001372 024615 EM6 ::READY BIT DIDN'T SET;YEAR VALUE MAY BE INVALID
57 001374 027517 DH1
58 001376 027670 DT1
59 001400 000000 000
60 ;ERROR#6 EM6
61 001402 024651 DH1
62 001404 027431 DT1
63 001406 027654 000
64 001410 000000 ;ERROR#7 EM7 ::READY BIT DIDN'T SET;ONE OR BOTH VALUES OF MONTH AND D
65 ;ERROR#7 EM7
66 001412 024730 DH1
67 001414 027431 DT1
68 001416 027654 000
69 001420 000000 ;ERROR#10 EM10 ::READY BIT DIDN'T SET;ONE OR BOTH VALUES OF HOUR AND MI
70 ;ERROR#10 EM10
71 001422 025040 DH1
72 001424 027431 DT1
73 001426 027654 000
74 001430 000000
```

75			:ERROR#11	
76	001432	025152	EM11	::ERROR: BIT #15 (DE) FAIL TO SET IN CSR
77	001434	027517	DH2	
78	001436	027670	DT2	
79	001440	000000	000	
80			:ERROR#12	
81	001442	025212	EM12	::ERROR: CLOCK DIDN'T STOP
82	001444	027517	DH2	::CLOCK WAS SUPPOSE TO STOP WHILE
83	001446	027670	DT2	::SETTING MONTH+DAY VALUE
84	001450	000000	000	
85			:ERROR#13	
86	001452	025250	EM13	::ERROR: CLOCK IS NOT RUNNING
87				
88	001454	027517	DH2	::CLOCK WAS SUPPOSE TO RUN AFTER
89	001456	027670	DT2	::SETTING MONTH+DAY AND HOUR+MIN
90	001460	000000	000	
91			:ERROR#14	
92	001462	025304	EM14	::INTERRUPT DIDN'T OCCUR WHILE UPDATING THE CLOCK
93	001464	027517	DH2	
94	001466	027670	DT2	
95	001470	000000	000	
96			:ERROR#15	
97	001472	025437	EM15	::INTERRUPT DIDN'T OCCUR ON PRESET VALUE
98	001474	027517	DH2	
99	001476	027670	DT2	
100	001500	000000	000	
101			:ERROR#16	
102	001502	025572	EM16	::SOFT ERROR
103	001504	027431	DH1	
104	001506	027654	DT1	
105	001510	000000	000	
106			:ERROR#17	
107	001512	025605	EM17	::UNEXPECTED INTERRUPT AS SOFTWARE SWITCH
108				::WAS NOT SET
109	001514	027431	DH1	
110	001516	027654	DT1	
111	001520	000000	000	
112			:ERROR#20	
113	001522	025655	EM20	::UNEXPECTED INTERRUPT AS INTRPT WAS NOT ENABLED
114	001524	027431	DH1	
115	001526	027654	DT1	
116	001530	000000	000	
117			:ERROR#21	
118	001532	025733	EM21	::UNEXPECTED INTERRUPT AS (DE) BIT
119				:: DIDN'T SET IN CSR
120	001534	027517	DH2	
121	001536	027670	DT2	
122	001540	000000	000	
123			:ERROR#22	
124	001542	026004	EM22	::UNEXPECTED UNEXPLAINABLE INTERRUPT
125	001544	027517	DH2	
126	001546	027670	DT2	
127	001550	000000	000	
128			:ERROR#23	
129	001552	026053	EM23	::UNEXPECTED INTERRUPT AS (TI) BIT
130				::WAS NOT SET IN CSR

131	001554	027517	DH2	
132	001556	027670	DT2	
133	001560	000000	000	
134				
135	001562	026121	:ERROR#24	
136			EM24	::TIME FAIL TO HOLD WHILE
137	001564	027431	DH1	::WHILE HOLD BIT WAS SET IN CSR
138	001566	027654	DT1	
139	001570	000000	000	
140				
141	001572	026207	:ERROR+25	
142	001574	027431	EM25	::ERROR:READY BIT WASN'T SUPPOSE TO BE SET
143	001576	027654	DH1	
144	001600	000000	DT1	
145			000	
146	001602	026256	:ERROR+26	
147	001604	027517	EM26	::'M'BIT DID NOT SET IN CSR
148	001606	027670	DH2	
149	001610	000000	DT2	
150			000	
151	001612	026321	:ERROR+27	
152	001614	027517	EM27	::TIME INTERVAL INTERRUPT DIDN'T OCCUR
153	001616	027670	DH2	
154	001620	000000	DT2	
155			000	
156	001622	026376	:ERROR+30	
157	001624	027517	EM30	::'M' AND/OR 'DE' BIT NOT SET IN CSR
158	001626	027670	DH2	
159	001630	000000	DT2	
160			000	
161				
162	001632	026445	:ERROR+31	
163	001634	027562	EM31	::HOLD BIT FAIL TO RESET IN TIME
164	001636	027704	DH4	
165	001640	000000	DT4	
166			000	
167	001642	026505	:ERROR+32	
168	001644	027517	EM32	::HOLD BIT FAIL. TO RESET
169	001646	027670	DH2	
170	001650	000000	DT2	
171			000	
172				
173	001652	026545	:ERROR+33	
174	001654	027547	EM33	::CRYSTAL OSCILLATOR SEEMS TO BE INACCURATE
175	001656	027700	DH3	
176	001660	000000	DT3	
177			000	
178	001662	026626	:ERROR+34	
179	001664	027634	EM34	::GOT NO INTERRUPT
180	001666	027720	DH5	::GOT NO INTERRUPT WITH BPU AT LEVEL 0
181	001670	000000	DT5	
182			000	
183	001672	026674	:ERROR+35	
184	001674	027431	EM35	::INTERRUPT OCCUR AT WRONG LEVEL
185	001676	027654	DH1	
186	001700	000000	DT1	
			000	

```
187 ;ERROR+36
188 001702 026734 EM36 ;:'DE' BIT FAIL TO CLEAR IN CSR
189 001704 027517 DH2
190 001706 027670 DT2
191 001710 000000 000
192
193
194 ; KW11C BUS REGISTER ADDRESS POINTERS
195
196 ;CSR: 160006 ;COMMAND/STATUS REGISTER
197
198 ; KW11C VECTOR ADDRESS POINTERS
199
200 ;VECO: 300 ;NORMAL INTERRUPT VECTOR
201 ;VEC2: 304 ;
202
203 ; KW11C DEVICE LEVEL
204
205 ;BRLV: 6
206
207 ; COMMAND STATUS REGISTER BIT ASSIGNMENTS
208
209 ; COMMON PROGRAM LOCATION(S)
210
211 ;
212 ;.MACR RDYCK
213 JSR PC, RDYBIT ;CHECK FOR READY BIT SET
214 TST FLAG.1 ;IS FLAG SET?
215 .ENDM
216
217 ;.MACR ECHO
218 MOV (SP)+, R4 ;STORE CHARACTER
219 MOVB R4, ECHOB ;TO ECHO THE CHARACTER
220 TYPE , ECHOB ;TYPE THE CHARACTER
221 .ENDM
222
223 ;.MACR ECHO1
224 MOV (SP)+, R4 ;STORE CHARACTER
225 MOVB R4, ECHOB1 ;ECHO THE CHARACTER
226 TYPE , ECHOB1
227 .ENDM
228 ;.MACR DECHK
229 MOV #77777,COUNTR ;:DELAY LOOP SETUP
230 JSR PC, DMT ;:DEAD MAN TIMER ROUTINE
231 BIT #BIT15,@CSR ;:IS DE BIT SET
232 BEQ -12 ;:TRY AGAIN
233 .ENDM
234
235 ;.MACR DECLR
236 MOV #77777,COUNTR ;:DELAY LOOP SETUP
237 MOV #0, @CSR ;:CLEAR CSR
238 JSR PC, DMT ;:DEAD MAN TIMES ROUTINE
239 BIT #BIT15, @CSR ;:IS BIT15 CLR?
240 BNE -20 ;:NO:BR
241 .ENDM
242
243 ;.MACR COMPAR ?B
```

```
244          CMP    GOOD,  #74          ;; EXPECTED VALUE MORE THAN OR EQUALS 60.?
245          BLT    B          ;; NO:BR
246          SUB    #74,  GOOD          ;; EXPECTED SECOND CAN'T BE MORE THAN 59
247          B:    CMP    GOOD,  BAD          ;; EXPECTED=ACTUAL?
248          .ENDM
250
251
253          .SBTTL PROGRAM START
254 001712 START:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;; CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001712 012706 001100 MOV    # $CMTAG, R6          ;; FIRST LOCATION TO BE CLEARED
(1) 001716 005026 CLR    (R6)+          ;; CLEAR MEMORY LOCATION
(1) 001720 022706 001140 CMP    # SWR, R6 ;; DONE?
(1) 001724 001374 BNE    -6          ;; LOOP BACK IF NO
(1) 001726 012706 001100 MOV    # STACK, SP          ;; SETUP THE STACK POINTER
(1) ;; INITIALIZE A FEW VECTORS
(1) 001732 012737 023342 000020 MOV    # SCOPE, @ IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
(1) 001740 012737 000340 000022 MOV    # 340, @ IOTVEC+2 ;; LEVEL 7
(1) 001746 012737 022656 000030 MOV    # ERROR, @ EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
(1) 001754 012737 000340 000032 MOV    # 340, @ EMTVEC+2 ;; LEVEL 7
(1) 001762 012737 024324 000034 MOV    # STRAP, @ TRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
(1) 001770 012737 000340 000036 MOV    # 340, @ TRAPVEC+2; LEVEL 7
(1) 001776 012737 023634 000024 MOV    # SPURDN, @ PURVEC ;; POWER FAILURE VECTOR
(1) 002004 012737 000340 000026 MOV    # 340, @ PURVEC+2 ;; LEVEL 7
(1) 002012 013737 020114 020106 MOV    $ENDCT, $EOPCT ;; SETUP END-OF-PROGRAM COUNTER
(1) 002020 005037 001172 CLR    $TIMES          ;; INITIALIZE NUMBER OF ITERATIONS
(1) 002024 005037 001174 CLR    $ESCAPE          ;; CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002030 112737 000001 001115 MOV    # 1, $ERMAX          ;; ALLOW ONE ERROR PER TEST
(1) 002036 012737 002036 001106 MOV    # ., $SLPADR          ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002044 012737 002044 001110 MOV    # ., $LPERR          ;; SETUP THE ERROR LOOP ADDRESS
(2) ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;; EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002052 013746 000004 MOV    @ ERRVEC, -(SP) ;; SAVE ERROR VECTOR
(2) 002056 012737 002112 000004 MOV    # 64$, @ ERRVEC ;; SET UP ERROR VECTOR
(2) 002064 012737 177570 001140 MOV    # DSWR, SWR          ;; SETUP FOR A HARDWARE SWICH REGISTER
(2) 002072 012737 177570 001142 MOV    # DISP, DISPLAY ;; AND A HARDWARE DISPLAY REGISTER
(2) 002100 022777 177777 177032 CMP    # -1, @ SWR          ;; TRY TO REFERENCE HARDWARE SWR
(2) 002106 001012 BNE    66$          ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;; AND THE HARDWARE SWR IS NOT = -1
(2) 002110 000403 BR    65$          ;; BRANCH IF NO TIMEOUT
(2) 002112 012716 002120 64$: MOV    # 65$, (SP)          ;; SET UP FOR TRAP RETURN
(2) 002116 000002 RTI
(2) 002120 012737 000176 001140 65$: MOV    # SWREG, SWR          ;; POINT TO SOFTWARE SWR
(2) 002126 012737 000174 001142 MOV    # DISPREG, DISPLAY
(2) 002134 012637 000004 66$: MOV    (SP)+, @ ERRVEC ;; RESTORE ERROR VECTOR
(1)
(2) 002140 005037 001214 CLR    $PASS          ;; CLEAR PASS COUNT
(2) 002144 132737 000200 001227 BITB   # APTSIZE, $ENVM ;; TEST USER SIZE UNDER APT
(2) 002152 001403 BEQ    67$          ;; YES, USE NON-APT SWITCH
(2) 002154 012737 001230 001140 MOV    # $SWREG, SWR          ;; NO, USE APT SWITCH REGISTER
(2) 002162 67$:
255 .SBTTL TYPE PROGRAM NAME
(1) ;; TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002162 005227 177777 INC    # -1          ;; FIRST TIME?
(1) 002166 001045 BNE    68$          ;; BRANCH IF NO
```

```
(1) 002170 022737 020146 000042      CMP      #SENDAD,@#42      ;;ACT-11?
(1) 002176 001441                      BEQ      68$              ;;BRANCH IF YES
(1) 002200 104401 002246                      TYPE     ,69$            ;;TYPE ASCIZ STRING
(2)                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002204 005737 000042                      TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002210 001012                      BNE     70$              ;;BRANCH IF YES
(2) 002212 123727 001226 000001          CMPB    $ENV,#1          ;;ARE WE RUNNING UNDER APT?
(2) 002220 001406                      BEQ     70$              ;;BRANCH IF YES
(2) 002222 023727 001140 000176          CMP     SWR,#SWREG       ;;SOFTWARE SWITCH REG SELECTED?
(2) 002230 001005                      BNE     71$              ;;BRANCH IF NO
(2) 002232 104406                      GTSWR                      ;;GET SOFT-SWR SETTINGS
(2) 002234 000403                      BR      71$
(2) 002236 112737 000001 001134 70$:  MOVB    #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
(2) 002244 71$:
(1) 002244 000416                      BR      68$              ;;GET OVER THE ASCIZ
(1)                                ;;69$: .ASCIZ <CRLF>#MAINDEC-11-CZKWL.A0-KW11C#<CRLF>
(1)                                68$:
256 002302 005227 177777          FLOAT:  INC     #-1              ;;PRINT ONCE
257 002306 001402                      BEQ     6$
258 002310 000137 002714                      JMP     SETUP4
259 002314 012737 002452 000004 6$:  MOV     #TIMOUT,@#ERRVEC  ;;TIME OUT INTERRUPT ADDRESS
260 002322 005037 024212                      CLR     FLAG.2
261 002326 012704 002522                      MOV     #OCTMOD,R4        ;;POINTER TO THE OCTAL MODULE TABLE
262 002332 012700 160010                      MOV     #160010,R0        ;;STARTING ADDRESS OF DEV TABLE
263 002336 012702 002566                      MOV     #RANKTA,R2        ;;POINTER TO THE RANK TABLE
264
265 002342 022712 000021          1$:  CMP     #21,(R2)          ;;IS IT LAST ADDRESS?
266 002346 001415                      BEQ     5$              ;;BRANCH IF YES
267
268 002350 011005          2$:  MOV     (R0),R5          ;;TRY TO ADDRESS THE DEVICE ADDRESS
269 002352 000240          3$:  NOP
270 002354 000240                      NOP
271 002356 005737 024212                      TST     FLAG.2          ;;IS INTRPT FLAG SET?
272 002362 001403                      BEQ     4$              ;;BRANCH IF NOT SET
273 002364 005037 024212                      CLR     FLAG.2          ;;CLEAR INTRPT FLAG
274 002370 000764                      BR      1$              ;;ADDRESS NEXT ADDRESS
275 002372 061400          4$:  ADD     (R4),R0          ;;ADD CONTENT OF MOD TAB TO
276                                ;;GET NEXT ADDRESS
277                                ;;POP UP OCTMOD TAB
278                                ;;POP UP TANKTA
279                                ;;BRANCH
280 002400 000760          5$:  BR      1$
281 002402 010405                      MOV     R4,R5            ;;CURRENT POINTER
282 002404 162705 000002                      SUB     #2,R5            ;;BACK UP BY 2
283 002410 161500                      SUB     (R5),R0          ;;GET THE DESIRED ADDRESS
284 002412 010037 024064                      MOV     R0,RG0          ;;ADDRESS OF KW11C
285 002416 062700 000002                      ADD     #2,R0            ;;GET NEXT ADDRESS
286 002422 010037 024066                      MOV     R0,RG2          ;;NEXT DEV ADDRESS
287 002426 062700 000002                      ADD     #2,R0
288 002432 010037 024070                      MOV     R0,RG4
289 002436 062700 000002                      ADD     #2,R0
290 002442 010037 001262                      MOV     R0,$BASE
291 002446 000137 002632                      JMP     SETUP1
```

;;THIS IS TIMEOUT INTERRUPT ROUTINE

291
292
293
294

```
295  
296 002452 012737 000001 024212 TIMEOUT: MOV #1, FLAG.2 ;;SET INTRPT FLAG  
297 002460 005724 TST (R4)+ ;;INCREMENT OCTMOD POINTR  
298 002462 005722 TST (R2)+ ;;INCREMENT RANKTAB POINTR  
299 002464 021427 000020 CMP (R4), #20 ;;IS NEXT LOC AT EVEN ADDRESS  
300 002470 001403 BEQ 1$ ;;BRANCH IF YES  
301 002472 062700 000010 ADD #10, R0 ;;INDICATE NO DEVICE  
302 002476 000410 BR 3$ ;;RETURN FROM INTRPT  
303 002500 032700 000010 1$: BIT #10, R0 ;;IS IT ON EVEN BOUNDARY  
304 002504 001403 BEQ 2$ ;;BRANCH IF YES  
305 002506 062700 000010 ADD #10, R0 ;;MAKE IT EVEN BOUNDARY  
306 002512 000402 BR 3$ ;;RETURN FROM INTRPT  
307 002514 062700 000020 2$: ADD #20, R0 ;;INDICATES NO DEVICE PRESENT  
308 002520 000002 3$: RTI ;;RETURN FROM INTRPT  
309  
310 002522 000010 OCTMOD: 0010  
311 002524 000020 0020  
312 002526 000010 0010  
313 002530 000010 0010  
314 002532 000010 0010  
315 002534 000010 0010  
316 002536 000010 0010  
317 002540 000010 0010  
318 002542 000010 0010  
319 002544 000010 0010  
320 002546 000010 0010  
321 002550 000020 0020  
322 002552 000010 0010  
323 002554 000010 0010  
324 002556 000010 0010  
325 002560 000010 0010  
326 002562 000000 0000  
327 002564 000000 0000  
328  
329 002566 000001 RANKTA: 0001  
330 002570 000002 0002  
331 002572 000003 0003  
332 002574 000004 0004  
333 002576 000005 0005  
334 002600 000006 0006  
335 002602 000007 0007  
336 002604 000010 0010  
337 002606 000011 0011  
338 002610 000012 0012  
339 002612 000013 0013  
340 002614 000014 0014  
341 002616 000015 0015  
342 002620 000016 0016  
343 002622 000017 0017  
344 002624 000020 0020  
345 002626 000021 0021  
346 002630 000022 0022  
347  
348 002632 012737 017770 000004 SETUP1: MOV #TRAP4, @ERRVEC ;;ADDRESS OF TRAP ROUTINE  
349 002640 013700 001256 MOV $VECT1,R0 ;;GET PRIORITY LEVEL  
350 002644 000300 SWAB R0 ;;PUT IN LEFT HALF
```

```

351 002646 042700 177400      BIC    #177400,R0      ::CLEAR UNWANTED BITS
352 002652 010037 024104      MOV    RO,PRI         ::SAVE DEVICE LEVEL
353                               :
354                               : NOW SETUP BUS ADDRESS VALUES
355                               :
356 002656 013737 001262 024062 SETUP2: MOV    $BASE,CSR      ::STORE BUS ADDRESS
357                               :
358                               : NOW SETUP VECTOR ADDRESSES
359                               :
360 002664 012700 024072      MOV    #PCV0,R0       ::SETUP VECTOR POINTER
361 002670 013701 001256      MOV    $VECT1,R1      ::GET BASE VECTOR ADDR
362 002674 042701 177400      BIC    #-400,R1       ::CLEAR UNWANTED BITS
363 002700 010120      SETUP3: MOV    R1,(R0)+  ::PUT ADDR
364 002702 062701 000002      ADD    #2,R1          ::INCR. TO NEXT LOC.
365 002706 022700 024102      CMP    #PCV1+4,R0     ::DONE ALL LOCATIONS?
366 002712 001372      BNE    SETUP3         ::BR, IF NOT DONE
367 002714 005737 000042      SETUP4: TST    @#42    ::TEST CHAIN MODE UNDER XXDP
368 002720 001402      BEQ    $             ::BR, IF CHAIN MODE
369 002722 000137 002770      JMP    HELP
370 002726 012777 017554 021136 1$: MOV    #INTRPT,@PCV0   ::ADDRESS OF INTERRUPT ROUTINE
371 002734 013777 024104 021132  MOV    PRI,@PSV0      ::RISE PRIORITY
372 002742 012777 017662 021126  MOV    #TIMRTN,@PCV1  ::ADDRESS OF TIME INTERVAL INTERRUPT ROUTINE
373 002750 013777 024104 021122  MOV    PRI,@PSV1     ::BUS LEVEL
374 002756 005227 177777      INC    #-1           ::
375 002762 001402      BEQ    HELP          ::PROMPT FOR INPUT
376 002764 000137 003530      JMP    RESTR
377 002770
(1) 002770 104401 002776      HELP:  TYPE    ,65$      ::TYPE ASCIZ STRING
(1) 002774 000434      BR      64$          ::GET OVER THE ASCIZ
(1)                               ::65$: .ASCIZ <200>/SELECT FUNCTION BELOW. TYPE CHARACTER CARRIAGE RETURN/
(1) 003066      64$:
(1) 003072 104401 003074      TYPE    ,67$      ::TYPE ASCIZ STRING
(1) 003072 000424      BR      66$          ::GET OVER THE ASCIZ
(1)                               ::67$: .ASCIZ <200>/ 'T' TO TEST CLOCK(CONTROL C TO EXIT)/
(1) 003144      66$:
(1) 003150 104401 003152      TYPE    ,69$      ::TYPE ASCIZ STRING
(1) 003150 000413      BR      68$          ::GET OVER THE ASCIZ
(1)                               ::69$: .ASCIZ <200>/ 'S' TO SET CLOCK/
(1) 003200      68$:
(1) 003204 104401 003206      TYPE    ,71$      ::TYPE ASCIZ STRING
(1) 003204 000413      BR      70$          ::GET OVER THE ASCIZ
(1)                               ::71$: .ASCIZ <200>/ 'R' TO READ CLOCK/
(1) 003234      70$:
(1) 003240 104401 003242      TYPE    ,73$      ::TYPE ASCIZ STRING
(1) 003240 000424      BR      72$          ::GET OVER THE ASCIZ
(1)                               ::73$: .ASCIZ <200>/ 'M' MODIFY OR READ ADDRESS AND VECTOR/
(1) 003312      72$:
(1) 003316 104401 003320      TYPE    ,75$      ::TYPE ASCIZ STRING
(1) 003316 000414      BR      74$          ::GET OVER THE ASCIZ
(1)                               ::75$: .ASCIZ <200>/ 'A' ACCURACY OF CLOCK/
(1) 003350      74$:
(1) 003354 104401 003356      TYPE    ,77$      ::TYPE ASCIZ STRING
(1) 003354 000424      BR      76$          ::GET OVER THE ASCIZ
(1)                               ::77$: .ASCIZ <200>/ 'G' MODIFY SOFTWARE SWITCH REGISTER/
(1) 003426      76$:
384 003426 104401 003434      TYPE    ,79$      ::TYPE ASCIZ STRING
  
```

(1)	003432	000413			BR	78\$::GET OVER THE ASCIZ
(1)					79\$:	.ASCIZ	<200>/ 'E' EXIT	HALTS CPU/
(1)	003462				78\$:	TYPE	81\$::TYPE ASCIZ STRING
385	003462	104401	003470		BR	80\$::GET OVER THE ASCIZ
(1)	003466	000420			81\$:	.ASCIZ	<200>/ 'H' HELP	PRINTS THIS MESSAGE/
(1)					80\$:	RESTR:	TYPE	,\$\$RFL
(1)	003530						TYPE	,\$\$QUES
386	003530	104401	001203			CKSWR		
387	003534	104401	001202		MOV	#STACK, SP		::SET THE STACK POINTER
388	003540	104407			CLR	\$TIMES		::INITIALIZE NUMBER OF ITERATION
389	003542	012706	001100		RDLIN			
390	003546	005037	001172		MOV	@(SP)+, R4		::GET THE CHARACTER
391	003552	104411			CMP	#124, R4		::IS IT 'T'?
392	003554	013604			BNE	1\$::NO:BR
393	003556	022704	000124		JMP	LOGTST		::GO TEST THE CLOCK
394	003562	001002						
395	003564	000137	004002		1\$:	CMP	#123, R4	::IS IT 'S'?
396					BNE	2\$::NO:BR
397	003570	022704	000123		JSR	PC,NEWTIM		::GET NEW TIME
398	003574	001003			BR	RESTR		::WAIT FOR CHARACTER
399	003576	004737	013136					
400	003602	000752			2\$:	CMP	#122, R4	::IS IT 'R'?
401					BNE	3\$::NO:BR
402	003604	022704	000122		JSR	PC, READ		::GO READ THE CURRENT TIME
403	003610	001003			BR	RESTR		
404	003612	004737	014426					
405	003616	000744			3\$:	CMP	#115, R4	::IS IT 'M'?
406					BNE	4\$::NO:BR
407	003620	022704	000115		JSR	PC, DEVCOD		::GET NEW ADDRESS AND VECTOR
408	003624	001003			BR	RESTR		
409	003626	004737	017234					
410	003632	000736			4\$:	CMP	#101, R4	::IS IT 'A'?
411					BNE	5\$::NO:BR
412	003634	022704	000101		JSR	PC, A.CLK		::GO CHECK ACCURACY OF CLOCK
413	003640	001004			RESET			
414	003642	004737	014754		BR	RESTR		
415	003646	000005						
416	003650	000727			5\$:	CMP	#105, R4	::IS IT 'E'?
417					BNE	6\$::NO:BR
418	003652	022704	000105		HALT			::HALT CPU
419	003656	001003			JMP	HELP		::RESTART THE PROGRAM
420	003660	000000						
421	003662	000137	002770		6\$:	CMP	#110, R4	::IS IT 'H'?
422					BNE	7\$::NO:BR
423	003666	022704	000110		JMP	HELP		::GO TO PRINT HELP MESSAGE
424	003672	001002			7\$:	CMP	#107, R4	::IS IT 'G'?
425	003674	000137	002770		BNE	8\$::NO:BR
426	003700	022704	000107		GTSWR			
427	003704	001003			JMP	RESTR		
428	003706	104406						
429	003710	000137	003530		8\$:	TYPE	65\$::TYPE ASCIZ STRING
430	003714				BR	64\$::GET OVER THE ASCIZ
(1)	003714	104401	003722		65\$:	.ASCIZ	<200>/ 'ILLEGAL CHARACTER,TYPE 'H' FOR HELP'/	
(1)	003720	000424			64\$:			
(1)								
(1)	003772							

```

431 003772 000137 003530
432 003776 000137 001712      ROUT1:  JMP      RESTRT
433
434
435 004002 012703 000010      LOG13T:  MOV      #10,   R3          ;;LOOP COUNTER
436 004006 032777 000400 020046 3$:      BIT      #BIT8, @CSR          ;;CHECK BATTERY?
437 004014 001426                    BEQ      5$                    ;;BRANCH IF BATTERY IS O.K.
438 004016 104401 004024                    TYPE     ,65$                  ;;TYPE ASCIZ STRING
(1) 004022 000415                    BR       64$                  ;;GET OVER THE ASCIZ
(1) 004056                    ;;65$:  .ASCIZ <200>/'BATTERY IS DISCHARGING'/
439 004056 012777 000400 017776 64$:      MOV      #BIT8, @CSR          ;;RESET THE CLOCK
440 004064 005303                    DEC      R3                    ;;COUNT DOWN
441 004066 001347                    BNE     3$                    ;;BRANCH IF NOT ZERO
442 004070 000000                    HALT
443 004072 004737 016106 5$:      JSR     PC,   CNTLC          ;;IS IT CONTROL C?
444 004076 104407                    CKSWR                    ;;IS SOFTWARE SWITCH SELECTED?
445
446
447 ;THIS ROUTINE WILL SET AND RESET ALL WORKING BITS IN CSR
448 ;
449 ;*****
(3) ;*TEST 1 TEST SET/RESET BITS IN CSR
(3) ;*****
(2) 004100 000004      TST1:  SCOPE
(1) 004102 012737 000012 001172      MOV      #10, $TIMES          ;;DO 10. ITERATIONS
450 004110 012737 000340 177776      MOV      #PR7, PSW          ;;GET THE PRIORITY
451 004116 012700 004376      MOV      #BITAB, RO         ;;POINTER TO THE TABLE
452 004122 022700 004436 1$:      CMP      #BITEN, RO         ;;IS IT END OF THE TABLE?
453 004126 003002                    BGT     2$                    ;;NO:BR
454 004130 000137 004214                    JMP     4$                    ;;EXIT
455 004134 012077 017722 2$:      MOV      (RO)+, @CSR        ;;GET THE VALUE FROM THE TABLE
456 004140 012037 001124      MOV      (RO)+, GOOD        ;;EXPECTED VALUE IN CSR
457 004144 013737 001124 024214      MOV      GOOD, GOOD1        ;;EXPECTED VALUE IN GOOD1 TOO
458 004152 005137 024214      COM      GOOD1              ;;UNWANTED BITS IN GOOD1
459 004156 017737 017700 001126      MOV      @CSR, BAD          ;;READ CSR
460 004164 043737 024214 001126      BIC     GOOD1, BAD          ;;CLEAR UNWANTED BITS
461 004172 023737 001124 001126      CMP      GOOD, BAD          ;;ARE THEY EQUAL?
462 004200 001404                    BEQ     3$                    ;;YES:BR
463 004202 017737 017654 001122      MOV      @CSR, BADA         ;;READ STATUS OF CSR
464 004210 104001      ERROR+1 ;;THE BIT SET IN CSR NOT MATCH WITH EXPECTED.
465 004212 000743 3$:      BR       1$                    ;;GO GET NEXT VALUE
466 004214 012777 014356 017640 4$:      MOV      #14356, @CSR       ;;SET POSSIBLE BITS IN CSR
467 004222 005077 017634      CLR     @CSR                 ;;CLEAR ALL BITS IN CSR
468 004226 017737 017630 001126      MOV      @CSR, BAD          ;;READ CURRENT CSR
469 004234 012737 000000 001124      MOV      #0, GOOD           ;;EXPECTED VALUE
470 004242 013737 001124 024214      MOV      GOOD, GOOD1        ;;
471 004250 005137 024214      COM     GOOD1                ;;
472 004254 043737 024214 001126      BIC     GOOD1, BAD          ;;CLEAR UNWANTED BITS
473 004262 023737 001124 001126      CMP     GOOD, BAD           ;;ARE THEY EQUAL?
474 004270 001404                    BEQ     5$                    ;;YES:BR
475 004272 017737 017564 001122      MOV     @CSR, BADA         ;;READ CSR
476 004300 104001      ERROR+1
477 004302 012777 014356 017552 5$:      MOV     #14356, @CSR        ;;
478 004310 000005      RESET
479 004312 017737 017544 001126      MOV     @CSR, BAD          ;;READ CSR
  
```


TEST SET/RESET BITS IN CSR

```
480 004320 012737 000000 001124 MOV #0, GOOD ;; EXPECTED VALUE
481 004326 013737 001124 024214 MOV GOOD, GOOD1 ;;
482 004334 005137 024214 COM GOOD1 ;;
483 004340 043737 024214 001126 BIC GOOD1, BAD ;; CLEAR UNWANTED BITS
484 004346 023737 001124 001126 CMP GOOD, BAD ;; ARE THEY EQUAL?
485 004354 001403 BEQ 6$, ;; YES:BR
486 004356 017737 017500 001122 MOV @CSR, BADA ;; RAD STATUS OF CSR
487 004364 004737 016106 6$: JSR PC, CNTLC
488 004370 104407 CKSWR
489 004372 000137 004440 JMP BITEN+2 ;; EXIT
490
491 004376 000001 BITAB: 000001
492 004400 000001 000001
493 004402 000002 000002
494 004404 000002 000002
495 004406 000004 000004
496 004410 000004 000004
497 004412 000010 000010
498 004414 000010 000010
499 004416 000040 000040
500 004420 000040 000040
501 004422 000100 000100
502 004424 000100 000100
503 004426 100000 100000
504 004430 004000 004000
505 004432 040000 040000
506 004434 002000 002000
507 004436 000000 BITEN: 00000
508
509
510
(3)
::*****
:*TEST 2 TEST TIMEINTERVAL COUNTER DECREMENTS IN PROPER ORDER
(3)
::*****
(2) 004440 000004 TST2: SCOPE
(1) 004442 012737 000012 001172 MOV #10., $TIMES ;; DO 10. ITERATIONS
511 004450 004737 016106 JSR PC, CNTLC
512 004454 104407 CKSWR
513 004456 005037 024046 CLR NUMBR
514 004462 012702 047777 1$: MOV #47777, R2 ;; TIMEINT COUNT IN R2
515 004466 012737 007777 024050 MOV #7777, KOUNT ;; ACTUAL VALUE IN REG4
516 004474 005077 017362 CLR @CSR ;; CLEAR CSR
517 004500 010277 017364 MOV R2, @RG4 ;; TIMINT COUNT INTO DEV REG
518 004504 012777 000010 017350 11$: MOV #BIT3, @CSR ;; READ TIME INTERVAL
519 004512 005337 024050 DEC KOUNT ;; DECREMENT COUNT BY ONE
520 004516 012737 077777 024052 2$: MOV #77777, COUNTR ;; INITIALIZE THE COUNTER
521 004524 004737 017206 3$: JSR PC, DMT ;; DEAD MAN TIMER
522 004530 027737 017334 024050 CMP @RG4, KOUNT ;; IS TIME INTERVAL CHANGES?
523 004536 001372 BNE 3$ ;; BRANCH IF NOT CHANGED
524 004540 005737 024210 TST FLAG.1 ;; IS ERROR FLAG SET
525 004544 001413 BEQ 4$ ;; BRANCH IF NOT
526 004546 017737 017316 001126 MOV @RG4, BAD ;; READ TIME INTERVAL
527 004554 013737 024050 001124 MOV KOUNT, GOOD ;; EXPECTED
528 004562 017737 017274 001122 MOV @CSR, BADA ;; STATUS OF CSR
529 004570 104003 ERROR+3 ;; DEAD MAN TIMER FAILED
530 004572 000733 BR 1$ ;; TRY AGAIN
531 004574 005237 024046 4$: INC NUMBR ;; JUST FOR CHECK
```

TEST TIMEINTERVAL COUNTER DECREMENTS IN PROPER ORDER

532	004600	005337	024050		DEC	KOUNT		::CHECK NEXT DECREMENT
533	004604	001344			BNE	Z\$::BRANCH IF NOT ZERO YET
534	004606	012737	007776	001124	MOV	#7776,	GOOD	::EXPECTED VALUE IN GOOD
535	004614	013737	024046	001126	MOV	NUMBER,	BAD	::ACTUAL VALUE IN BAD
536	004622	023737	001124	001126	CMP	GOOD,	BAD	::IS IT ORIGINAL COUNT?
537	004630	001404			BEQ	5\$::BRANCH IF YES
538	004632	017737	017224	001122	MOV	@CSR,	BADA	::STATUS OF CSR
539	004640	104001			ERROR+1			::VALUE DIDN'T MATCH WITH EXPCT
540	004642	012702	077777		5\$:	MOV	#77777,	R2
541	004646	032777	040000	017206	6\$:	BIT	#BIT14,	@CSR
542	004654	001006			BNE	7\$::IS (TI) BIT SET IN CSR ?
543	004656	005302			DEC	R2		::BRANCH IF BIT#14 SET
544	004660	001372			BNE	6\$::DECREMENT COUNTER
545	004662	017737	017174	001122	MOV	@CSR,	BADA	::BRANCH IF NOT ZERO
546	004670	104004			ERROR+4			::CONTENT OF CSR
547	004672	000005			7\$:	RESET		::BIT #14 DIDN'T SET
548	004674	000240			NOP			
549	004676	000240			NOP			
550	004700	032777	040000	017154	BIT	#BIT14,	@CSR	::(TI) BIT STILL SET?
551	004706	001405			BEQ	8\$::BRANCH IF CLEARED
552	004710	017737	017146	001122	MOV	@CSR,	BADA	::STATUS OF CSR
553	004716	104005			ERROR+5			::BIT#14 FAIL TO RESET
554	004720	000660			BR	1\$::TRY AGAIN
555	004722	004737	016106		8\$:	JSR	PC,	CNTLC
556	004726	104407			CKSWR			

557								
558								
559								
560								
561								
562								
(3)								
(3)								
(2)	004730	000004						
(1)	004732	012737	000012	001172	TST3:	SCOPE		
563	004740	004737	016106		MOV	#10,,	\$TIMES	::DO 10. ITERATIONS
564	004744	104407			JSR	PC,	CNTLC	::IS IT CONTROL C?
565	004746	012737	000300	024054	1\$:	CKSWR		
566	004754	005077	017102		MOV	#300,	CONST	::ASSUME TIME CONSTANT
567	004760	012777	047777	017102	11\$:	CLR	@CSR	::CLEAR CSR
568	004766	012777	000011	017066	MOV	#47777,	@RG4	::TIME INTERVAL INTO REG
569	004774	017737	017070	024202	MOV	#BIT3+BIT0,	@CSR	::READ TIME INTERVAL SETTING HOLD BIT
570	005002	012704	000500		MOV	@RG4,	EXPCT	::READ TIMEINTERVAL
571					MOV	#500,	R4	::TRY SO MANY TIME
572	005006	005304			2\$:	DEC	R4	::DECREMENT COUNTER
573	005010	001460			BEQ	7\$::EXIT IF ZERO
574	005012	005337	024202		DEC	EXPCT		::DECREMENT READ INTVAL BY ONE
575								
576	005016	042777	000001	017036	3\$:	BIC	#BIT0,	@CSR
577	005024	017737	017040	024056	MOV	@RG4,	READA	::ACTUAL TIME INTERVAL
578	005032	023737	024202	024056	CMP	EXPCT,	READA	::IS IT EQUAL TO EXPECTED
579	005040	001366			BNE	3\$::TRY AGAIN
580	005042	013703	024054		MOV	CONST,	R3	::GET CONSTANT
581								
582	005046	005303			4\$:	DFC	R3	::DECIDES TIME CONST
583	005050	001376			BNE	4\$::DECREMENT TILL ZERO

```

584 005052 017737 017012 024056 MOV @RG4, READA ;;VALUE READ
585 005060 013737 024056 024060 MOV READA, READB ;; STORE IN OTHER LOCATION
586 ;;TO BE USED LATER
587 005066 163737 024202 024056 SUB EXPCT, READA ;;TIME INTERVAL HAS CHANGED?
588 005074 001004 BNE 5$ ;;BRANCH IF YES
589 005076 005237 024054 INC CONST ;;INCREMENT ASSUMED CONST VALUE
590 005102 000137 005006 JMP 2$ ;;TRY NEXT
591 005106 023727 024056 000002 5$: CMP READA, #2 ;;DIFF GREATER THAN 2?
592 005114 002007 BGE 6$ ;;YES BRANCH
593 005116 013737 024060 024202 MOV READB, EXPCT ;;CURRENT VALUE IN EXPECT
594 005124 005337 024054 DEC CONST ;;TO GET CORRECT CONST
595 005130 000137 005006 JMP 2$ ;;GO BACK
596 005134 013737 024060 024202 6$: MOV READB, EXPCT ;;EXPCT = READB
597 005142 006237 024054 ASR CONST ;;DIVIDE BY 2
598 005146 000137 005006 JMP 2$
599 005152 000005 7$: RESET
600 005154 000240 NOP
601 005156 000240 NOP
602 005160 004737 016106 JSR PC, CNTLC
603 005164 104407 CKSWR

```

```

;THIS ROUTINE WRITES TIMEINTERVAL VALUES IN DEV RG4
;AND READS THE VALUE IN RG4 AND THEN COMPARES WITH EXPECTED VALUE
;

```

```

;*****
;*TEST 4 CHECK TIMEINTERVAL WRITES AND READS CORRECTLY
;*****

```

```

(3)
(2) 005166 000004 TST4: SCOPE
(1) 005170 012737 000012 001172 MOV #10, $TIMES ;;DO 10. ITERATIONS
611 005176 012737 000340 177776 TIME1: MOV #PR7, PSW
612 005204 004737 016106 JSR PC, CNTLC ;;IS IT CONTROL C
613 005210 104407 CKSWR
614 005212 012700 005332 1$: MOV #TIMTAB,RO ;;POINTER TO THE TABLE
615 005216 022700 005354 2$: CMP #TIMEND,RO ;;IS IT END OF TABLE?
616 005222 001436 BEQ 4$ ;;EXIT
617 005224 005077 016632 CLR @CSR ;;CLEAR CSR
618 005230 012037 001124 MOV (R0)+, GOOD ;;VALUE TO BE WRITTEN IN RG4
619 005234 013777 001124 016626 MOV GOOD, @RG4 ;;IN DEV RG4
620 005242 052777 000010 016612 3$: BIS #BIT3, @CSR ;;SET BIT TO READ TIME INTERVAL
621 005250 017737 016614 001126 MOV @RG4, BAD ;;ACTUAL READ VALUE
622 005256 042737 040000 001124 BIC #40000, GOOD ;;CLEAR TIME INTERVAL ENABLE BIT
623 005264 013737 001124 024214 MOV GOOD, GOOD1 ;;KEEP GOOD FOR PRINT OUT
624 005272 163737 001126 024214 SUB BAD, GOOD1 ;;ARE THEY EQUAL?
625 005300 023727 024214 000001 CMP GOOD1, #1 ;;IS DIFFERENCE GREATER THAN 1
626 005306 003743 BLE 2$ ;;BRANCH IF NOT
627 005310 017737 016546 001122 MOV @CSR, BADA ;;GET STATUS OF CSR
628 005316 104016 ERROR+16 ;;ERROR:SOFT ERROR TIME INTERVAL
629 ;;SET DIDN'T MATCH WITH ACTUAL
630 005320 004737 016106 4$: JSR PC, CNTLC ;;IS IT CONTROL C?
631 005324 104407 CKSWR
632 005326 000137 005356 JMP TIMEND+2 ;;GO TO NEXT ROUTINE
633 005332 047777 TIMTAB: 47777
634 005334 046004 46004
635 005336 040030 40030

```

636 005340 046754 46754
637 005342 040004 40004
638 005344 042003 42003
639 005346 047502 47502
640 005350 042000 42000
641 005352 043254 43254
642 005354 000000
643
644
645
646
647
648

TIMEND: 00000

THIS ROUTINE CHECKS IF INTERRUPT OCCURS AT GIVEN VECTOR AND BR LEVEL

*TEST 5 TEST INTERRUPT OCCURS AT GIVEN VECTOR AND BR LEVEL

(3)
(3)
(2) 005356 000004
(1) 005360 012737 000001 001172
649 005366 012737 000340 024106
650 005374 012737 000005 024226
651 005402 012737 000007 024110
652 005410 162737 000040 024106
653 005416 013737 024106 177776
654 005424 005037 024212
655 005430 005077 016426
656 005434 012777 000040 016420
657 005442 012777 040077 016420
658 005450 012737 000002 016030
659 005456 004737 015726
660
661 005462 005737 024212
662
663 005466 001005
664 005470 005337 024110
665 005474 100345
666 005476 104034
667 005500 000417
668
669 005502 000005
670 005504 000240
671 005506 000240
672 005510 000240
673 005512 013737 024102 001124
674 005520 013737 024110 001126
675 005526 023737 001124 001126
676 005534 001401
677 005536 104035
678
679 005540
680
681
682
683
684
685

TST5: SCOPE
MOV #1, \$TIMES ;; DO 1 ITERATION
MOV #340, DPRI ;;
CHKINT: MOV #5, INTREG ;; INTERRUPT FLAG
MOV #7, LEVEL ;; SET LEVEL
1\$: SUB #40, DPRI ;; LOWER PRIORITY
MOV DPRI, PSW ;;
CLR FLAG.2 ;; CLEAR INTERRUPT FLAG
CLR @CSR ;; CLEAR CSR
MOV @BITS, @CSR ;; ENABLE INTERRUPT
MOV #40077, @RG4 ;; TIME INTERVAL VALUE
MOV #2, TIMES ;; 2 SECOND INTERVAL
JSR PC, INTRTN ;; WAIT FOR INTERRUPT
TST FLAG.2 ;; DID INTERRUPT OCCUR?
BNE 2\$;; YES:BR
DEC LEVEL ;;
BPL 1\$;;
ERROR+34 ;; GOT NO INTERRUPT WITH BPU AT LEVEL 0
BR 3\$;; EXIT
2\$: RESET
NOP
NOP
NOP
MOV BRLV, GOOD ;; EXPECTED BRLV
MOV LEVEL, BAD ;; ACTUAL LEVEL
CMP GOOD, BAD ;;
BEQ 3\$;; BRANCH IF EQUAL
ERROR+35 ;; INTERRUPTED AT WRONG LEVEL
3\$:

THIS ROUTINE IS TO INTERRUPT AT PARTICULAR TIME INTERVAL

*TEST 6 TEST TIME INTERVAL INTERRUPT

(3)
(3)

```

(2) 005540 000004          TST:  SCOPE
(1) 005542 012737 000012 001172      MOV  #10.,$TIMES      ;;DO 10. ITERATIONS
686 005550 013737 024106 177776 TIMINT: MOV  DPRI,  PSW      ;;GET PRIORITY
687 005556 004737 016106          JSR  PC,    CNTLC    ;;IS IT CONTROL C?
688 005562 104407          CKSWR
689 005564 012737 000005 024226      MOV  #5,    INTREG   ;;JUST FLAG FOR TIMINTERVAL
690          ;;INTERRUPT
691 005572 012700 005766          MOV  #TIMTA1,R0     ;;POINTER TO THE TABLE
692          .....
693 005576 022700 006002          1$:  CMP  #TIMEN1,R0   ;;IS IT END OF TABLE
694 005602 001455          BEQ  6$           ;;EXIT
695 005604 005037 024212          2$:  CLR  FLAG.2     ;;CLEAR FLAG
696 005610 005077 016246          CLR  @CSR        ;;CLEAR CSR
697 005614 012777 000050 016240      MOV  #BIT5+BIT3,@CSR ;;INTERRUPT ENABLE?
698 005622 012037 001124          MOV  (R0)+,  GOOD   ;;VALUE TO BE INTERRUPTED AT
699 005626 013777 001124 016234      MOV  GOOD,   @RG4   ;;IN REG4
700 005634 012737 000005 016030      MOV  #5,    TIMES   ;;WAIT FOR APPRO 5 SEC
701 005642 004737 015726          JSR  PC,    INTRTN  ;;WAIT FOR INTERRUPT
702 005646 005737 024212          TST  FLAG.2     ;;DID INTERRUPT OCCURED?
703 005652 001424          BEQ  5$           ;;BRANCH IF NOT
704 005654 005037 024212          CLR  FLAG.2     ;;CLEAR INTERRUPT FLAG
705 005660 042737 040000 001124      BIC  #40000,  GOOD  ;;CLEAR TIME INTERVAL ENABLE BIT
706 005666 013737 001124 024214      MOV  GOOD,   GOOD1  ;;KEEP GOOD TO PRINT ERKOR MSG
707 005674 163737 001126 024214      SUB  BAD,    GOOD1  ;;TIME LEFT IN RG4
708 005702 023727 024214 000001      CMP  GOOD1,  #1     ;;IS DIFF GREATER THAN ONE?
709 005710 003732          BLE  1$         ;;NO: GET NEXT VALUE
710 005712 013737 024120 001122      MOV  ACSR,   BADA   ;;READ STATUS OF CSR
711 005720 104001          ERROR+1
712 005722 000725          BR   1$         ;;GET NEXT VALUE
713 005724 017737 016132 001122  5$:  MOV  @CSR,   BADA   ;;GET STATUS OF CSR
714 005732 104027          ERROR+27      ;;TIME INTERVAL INTERRUPT DIDN'T OCCUR
715 005734 000720          BR   1$         ;;GET NEXT VALUE
716 005736 042777 040000 016124  6$:  BIC  #40000,  @RG4  ;;CLEAR INTRPT FLAG
717 005744 005037 024226          CLR  INTREG     ;;CLEAR THE FLAG
718 005750 005077 016106          CLR  @CSR      ;;CLEAR CSR
719 005754 004737 016106          JSR  PC,    CNTLC  ;;
720 005760 104407          CKSWR
721 005762 000137 006004          JMP  TIMEN1+2    ;;EXIT
722 005766 047777          TIMTA1: 47777
723 005770 040700          40700
724 005772 046003          46003
725 005774 040200          40200
726 005776 045065          45065
727 006000 040100          40100
728 006002 000000          TIMEN1: 00000
729 006004 004737 016324          JSR  PC,    GETRDY  ;;INITIALIZE READY BIT
730
731
732
733
734          ;;*****
(3)          ;;*TEST 7      TEST HOLD BIT SET/RESET IN TIME
(3)          ;;*****
(2) 006010 000004          TST:  SCOPE
(1) 006012 012737 000012 001172      MOV  #10.,$TIMES      ;;DO 10. ITERATIONS
735 006020 004737 016106          JSR  PC,    CNTLC

```

```

736 006024 104407          CKSWR
737 006026 012777 000001 016026  MOV  #BIT0, @CSR      ;;SET HOLD BIT IN CSR
738 006034 004737 016226          JSR  PC,   WAIT1     ;;WAIT FOR HOLD BIT TO RESET
739 006040 032777 000001 016014  BIT  #BIT0, @CSR     ;;IS HOLD BIT RESET?
740 006046 001404          BEQ  1$,           ;;YES:BR
741 006050 017737 016006 001122  MOV  @CSR,  BADA     ;;STATUS OF CSR
742 006056 104032          ERROR+32           ;;HOLD BIT FAIL TO RESET
743 006060 012777 000001 015774 1$:  MOV  #BIT0, @CSR     ;;SET HOLD BIT AGAIN
744 006066 005077 015770          CLR  @CSR          ;;FORCE TO RESET
745 006072 004737 016226          JSR  PC,   WAIT1     ;;WAIT
746 006076 032777 000001 015756  BIT  #BIT0, @CSR     ;;IS HOLD BIT RESET?
747 006104 001404          BEQ  2$,           ;;YES:BR
748 006106 017737 015750 001122  MOV  @CSR,  BADA     ;;STATUS OF CSR
749 006114 104032          ERROR+32           ;;HOLD BIT FAIL TO RESET WHEN FORCED TO RESET.
750 006116 004737 016106          JSR  PC,   CNTLC     ;;
751 006122 104407          CKSWR
752
753
754
755
756
757
758                                     ;:READ SECOND HERE
759                                     ;:
760                                     ;:*****
761 (3) *TEST 10      TEST SECOND VALUE IS CHANGING
762 (3) *****
763 (2) 006124 000004          TST10: SCOPE
764 (1) 006126 012737 000001 001172  MOV  #1,$TIMES      ;;DO 1 ITERATION
765 761 006134 004737 016404          JSR  PC,   GET2     ;;SET NEW YEAR,MONTH,DAY,HOUR,MIN
766 762 006140 017701 015724          MOV  @RG4,  R1     ;;READ SECOND
767 763 006144 027701 015720          1$:  CMP  @RG4,  R1     ;;DOES SECOND VALUE CHANGES?
768 764 006150 001775          BEQ  1$,           ;;NO:BR
769 765 006152 027701 015712          CMP  @RG4,  R1     ;;CHECK AGAIN
770 766 006156 001772          BEQ  1$,           ;;GO BACK
771 767 006160          2$:
772
773                                     ;:THIS ROUTINE CHECK HOLD BIT IN CSR
774                                     ;:
775                                     ;:*****
776 (3) *TEST 11      TEST HOLD BIT IN CSR HOLDS TIME WHILE SET
777 (3) *****
778 (2) 006160 000004          TST11: SCOPE
779 (1) 006162 012737 000012 001172  MOV  #10.,$TIMES   ;;DO 10. ITERATIONS
780 775 006170 005077 015666          HOLTST: CLR  @CSR      ;;CLEAR CSR
781 776 006174 012777 047777 015666  MOV  #47777, @RG4   ;;SET TIME INTERVAL
782 777 006202 012777 000001 015652  MOV  #BIT0, @CSR     ;;SET HOLD BIT
783 778 006210 017704 015654          MOV  @RG4,  R4     ;;READ SECOND VALUE
784 779 006214 032777 000001 015640 1$:  BIT  #BIT0, @CSR     ;;IS HOLD BIT SET?
785 780 006222 001414          BEQ  2$,           ;;BRANCH IF HOLD BIT IS RESET
786 781 006224 027704 015640          CMP  @RG4,  R4     ;;DID SECOND VALUE CHANGEd?
787 782 006230 001771          BEQ  1$,           ;;NO:BR
788 783 006232 032777 000001 015622  BIT  #BIT0, @CSR     ;;IS HOLD BIT RESET?

```

784	006240	001405				BEQ	2\$::YES:BR
785	006242	004737	016522			JSR	PC,	GETTIM	::READ ALL PRESENT VALUE
786	006246	104031				ERROR+31			::ERROR:HOLD BIT FAIL TO RESET
787	006250	000137	006774			JMP	10\$::EXIT
788	006254	004737	016522		2\$:	JSR	PC,	GETTIM	::READ CURRENT VALUE OF SECOND
789	006260	062704	000002			ADD	#2,	R4	::EXPECTED VALUE OF SECOND AFTER
790									::HOLD BIT RESET
791	006264	010437	001124			MOV	R4,	GOOD	::EXPECTED VALUE IN GOOD
792	006270	013737	016550	001126		MOV	SECOND,	BAD	::ACTUAL VALUE
793	006276	023727	001124	000074		CMP	GOOD,	#74	::EXPECTED VALUE MORE THAN OR EQUALS 60.?
(1)	006304	002403				BLT	64\$::NO:BR
(1)	006306	162737	000074	001124		SUB	#74,	GOOD	::EXPECTED SECOND CAN'T BE MORE THAN 59
(1)	006314	023737	001124	001126	64\$:	CMP	GOOD,	BAD	::EXPECTED=ACTUAL?
794	006322	001403				BEQ	3\$::YES THEY ARE EQUAL BRANCH
795	006324	104001				ERROR+1			::READ VALUE OF SECOND NOT MATCH
796									::WITH EXPECTED VALUE.
797	006326	000137	006774			JMP	10\$::JUMP
798	006332	012777	000011	015522	3\$:	MOV	#BIT0+BIT3,	@CSR	::SET HOLD BIT AGAIN
799	006340	017700	015524			MOV	@RG4,	R0	::READ TIME INTERVAL VALUE
800	006344	027700	015520		4\$:	CMP	@RG4,	R0	::ARE THEY EQUAL?
801	006350	001413				BEQ	5\$::YES:BR
802	006352	010037	001124			MOV	R0,	GOOD	::EXPECTED VALUE
803	006356	017737	015506	001126		MOV	@RG4,	BAD	::READ VALUE OF TIME INTERVAL
804	006364	017737	015472	001122		MOV	@CSR,	BADA	::READ STATUS OF CSR
805	006372	104001				ERROR+1			::EXPECTED VALUE NOT MATCH WITH ACTUAL VALUE
806	006374	000137	006774			JMP	10\$::EXIT
807	006400	032777	040000	015454	5\$:	BIT	#BIT14,	@CSR	::IS BIT14 SET IN CSR
808	006406	001756				BEQ	4\$::NO:BR
809	006410	005077	015446		6\$:	CLR	@CSR		::CLR CSR
810	006414	012777	000001	015440		MOV	#BIT0,	@CSR	::SET HOLD BIT
811	006422	013737	001126	001124		MOV	BAD,	GOOD	::GET PREVIOUS SECOND VALUE
812	006430	017737	015434	001126		MOV	@RG4,	BAD	::READ CURRENT SECOND
813	006436	023727	001124	000074		CMP	GOOD,	#74	::EXPECTED VALUE MORE THAN OR EQUALS 60.?
(1)	006444	002403				BLT	65\$::NO:BR
(1)	006446	162737	000074	001124		SUB	#74,	GOOD	::EXPECTED SECOND CAN'T BE MORE THAN 59
(1)	006454	023737	001124	001126	65\$:	CMP	GOOD,	BAD	::EXPECTED=ACTUAL?
814	006462	001406				BEQ	7\$::YES:BR
815	006464	017737	015372	001122		MOV	@CSR,	BADA	::READ CURRENT STATUS OF CSR
816	006472	104001				ERROR+1			::PREVIOUS SECOND NOT MATCH WITH
817									::CURRENT SECOND WHILE HOLD BIT
818									::WAS SET
819	006474	000137	006774			JMP	10\$::EXIT
820	006500	005077	015356		7\$:	CLR	@CSR		::FORCE HOLD BIT TO RESET
821	006504	004737	016152			JSR	PC,	DELAY	::WAIT FOR 10MSEC
822	006510	013737	001126	001124		MOV	BAD,	GOOD	::READ PREVIOUS SEC
823	006516	017737	015346	001126		MOV	@RG4,	BAD	::CURRENT SECOND
824	006524	023727	001124	000074		CMP	GOOD,	#74	::EXPECTED VALUE MORE THAN OR EQUALS 60.?
(1)	006532	002403				BLT	66\$::NO:BR
(1)	006534	162737	000074	001124		SUB	#74,	GOOD	::EXPECTED SECOND CAN'T BE MORE THAN 59
(1)	006542	023737	001124	001126	66\$:	CMP	GOOD,	BAD	::EXPECTED=ACTUAL?
825	006550	001406				BEQ	8\$::BR IF EQUAL
826	006552	017737	015304	001122		MOV	@CSR,	BADA	::READ CURRENT STATUS OF CSR
827	006560	104001				ERROR+1			::ACTUAL VALUE NOT MATCH WITH EXPECTED.
828	006562	000137	006774			JMP	10\$::EXIT
829	006566	012777	000001	015266	8\$:	MOV	#BIT0,	@CSR	::SET HOLD BIT NOW
830	006574	012737	043720	016224		MOV	#43720,	TCNT	::TIME INTERVAL VALUE FOR 1 SECOND

```
831 006602 004737 016204 JSR PC, WAIT2 ;;WAIT FOR A SECOND
832 006606 005077 015250 CLR @CSR ;;FORCE HOLD BIT TO CLEAR
833 006612 004737 016152 JSR PC, DELAY ;;WAIT FOR 10MSEC
834 006616 013737 001126 001124 MOV BAD, GOOD ;;READ PREVIOUS SECOND
835 006624 005237 001124 INC GOOD ;;CURRENT VALUE SHOULD BE ONE MORE
836 ;;THAN PREVIOUS VALUE
837 006630 017737 015234 001126 MOV @RG4, BAD ;;READ SECOND
838 006636 023727 001124 000074 CMP GOOD, #74 ;;EXPECTED VALUE MORE THAN OR EQUALS 60.?
(1) 006644 002403 BLT 67$ NO:BR
(1) 006646 162737 000074 001124 SUB #74, GOOD ;;EXPECTED SECOND CAN'T BE MORE THAN 59
(1) 006654 023737 001124 001126 67$: CMP GOOD, BAD ;;EXPECTED=ACTUAL?
839 006662 001406 BEQ 9$ YES:BR
840 006664 017737 015172 001122 MOV @CSR, BADA ;;STATUS OF CSR
841 006672 104001 ERROR+1
842 006674 000137 006774 JMP 10$ ;;EXIT
843 006700 005077 015156 9$: CLR @CSR ;;CLEAR BIT14 IN CSR
844 006704 012737 043720 016224 MOV #43720, TCNT ;;TIME INTERVAL VALUE FOR A SEC
845 006712 004737 016204 JSR PC, WAIT2 ;;WAIT FOR A SECOND
846 006716 013737 001126 001124 MOV BAD, GOOD ;;PREVIOUS SECOND
847 006724 017737 015140 001126 MOV @RG4, BAD ;;CURRENT SECOND
848 006732 005237 001124 INC GOOD ;;EXPECTED VALUE AFTER ONE SECOND
849 006736 023727 001124 000074 CMP GOOD, #74 ;;EXPECTED VALUE MORE THAN OR EQUALS 60.?
(1) 006744 002403 BLT 68$ NO:BR
(1) 006746 162737 000074 001124 SUB #74, GOOD ;;EXPECTED SECOND CAN'T BE MORE THAN 59
(1) 006754 023737 001124 001126 68$: CMP GOOD, BAD ;;EXPECTED=ACTUAL?
850 006762 001404 BEQ 10$ YES:BR
851 006764 017737 015072 001122 MOV @CSR, BADA ;;READ CSR
852 006772 104001 ERROR+1
853 006774 004737 016106 10$: JSR PC, CNTLC
854
855
856 .SBTTL CHECK DISPLAY ON THE CLOCK
857
(3) *****
;*TEST 12 TEST DIGITS IN DISPLAY WORKS O.K.
(3) *****
(2) 007000 000004 TST12: SCOPE
(1) 007002 012737 000001 001172 MOV #1,$TIMES ;;DO 1 ITERATION
858 007010 004737 016106 DISP: JSR PC, CNTLC
859 007014 104407 CKSWR
860 007016 005037 024152 CLR DHOUR ;;INITIALIZE HOUR+MIN
861 007022 012777 100000 015032 DISPLY: MOV #BIT15, @CSR ;;SET TIME
862 007030 012777 000401 015026 MOV #401, @RG0 ;;SET MONTH+DAY
863 007036 004737 017000 JSR PC, RDYBIT ;;CHECK FOR READY BIT SET
(1) 007042 005737 024210 TST FLAG.1 ;;IS FLAG SET?
864 007046 001407 BEQ 1$ ;;BRANCH IF READY BIT IS SET
865 007050 017737 015006 001122 MOV @CSR, BADA ;;STATUS OF CSR
866 007056 104002 ERROR+2 ;;READY BIT DIDN'T SET
867 007060 004737 016324 JSR PC, GETRDY ;;GET READY BIT SET
868 007064 000756 BR DISPLY
869 007066 012777 100000 014766 1$: MOV #BIT15, @CSR ;;SET HOUR + MIN
870 007074 013777 024152 014764 MOV DHOUR, @RG2 ;;SET HOUR+MIN
871 007102 004737 017000 JSR PC, RDYBIT ;;CHECK FOR READY BIT SET
(1) 007106 005737 024210 TST FLAG.1 ;;IS FLAG SET?
872 007112 001407 BEQ 2$ ;;BRANCH IF READY BIT SET
873 007114 017737 014742 001122 MOV @CSR, BADA ;;READ STATUS OF CSR
874 007122 104002 ERROR+2 ;;READY BIT NOT SET
```



```
875 007124 004737 016324 JSR PC, GETRDY ::GET READY BIT SET
876 007130 000727 BR DISP
877 007132 012737 000001 016104 2$: MOV #1, CONT3 ::WAIT FOR SECOND
878 007140 004737 016032 JSR PC, WAIT ::WAIT FOR SOME TIME TO
879 ::SEE DIGITS IN DISPLY
880 ::CHANGES
881 007144 005237 024152 INC DHOURL ::INCREMENT MINUTES
882 007150 122737 000073 024152 CMPB #73,DHOURL ::IS MINUTES=59
883 007156 002321 BGE DISPLY ::BRANCH IF NOT
884 007160 062737 000377 024152 ADD #377, DHOURL ::INCREMENT DHOURL
885 007166 022737 013473 024152 CMP #13473, DHOURL ::IS IT 23HOUR-59MIN
886 007174 002312 BGE DISPLY ::BRANCH
887 007176 004737 016106 JSR PC, CNTLC
888 007202 000005 RESET
889 007204 000240 NOP
890 007206 000240 NOP
891 007210 000240 NOP
892 007212 000137 007216 JMP TABDRV
893
894
895 ;THIS IS TABLE DRIVEN ROUTINE,IT TESTS INVALID VALUE OF MONTH+DAY
896 ;
897 ;
898 ;
899 ;
900 007216 TABDRV:
901
902 ;*****
903 ;*TEST 13 TEST CHECKS FOR VALID/INVALID VALUES OF YEAR
904 ;*****
905 (3)
906 (3)
907 (2) 007216 000004 TST13: SCOPE
908 (1) 007220 012737 000012 001172 MOV #10, $TIMES ::DO 10. ITERATIONS
909 007226 004737 016106 1$: JSR PC, CNTLC ::IS IT CONTROL C?
910 007232 013737 024106 177776 MOV DPRI, PSW
911 007240 104407 CKSWR ::SOFTWARE SWITCH SELECTED?
912 007242 012700 007524 MOV #TABLE1,RO ::POINTER TO THE TABLE1
913 007246 022700 007556 2$: CMP #TABEN1,RO ::IS IT END OF TABLE1
914 007252 003002 BGT 22$ ::BRANCH IF NOT
915 007254 000137 JMP TABEN1+2 ::EXIT ON END OF TABLE1
916 007260 005077 014576 22$: CLR @CSR ::CLEAR CSR
917 007264 012037 001124 MOV (RO)+, GOOD ::YEAR VALUE IN GOOD
918 007270 012777 040000 014564 MOV #BIT14,@CSR ::SET (SY) BIT IN CSR TO WRITE A
919 ::A YEAR VALUE
920 007276 013777 001124 014560 MOV GOOD, @RGO ::YEAR IN DEVICE REG
921 007304 004737 017000 JSR PC, RDYBIT ::CHECK FOR READY BIT SET
922 (1) 007310 005737 024210 TST FLAG.1 ::IS FLAG SET?
923 007314 001443 BEQ 33$ ::BRANCH IF READY BIT IS SET
924 007316 012737 077777 024052 MOV #77777,COUNTR ::DELAY LOOP SETUP
925 (1) 007324 004737 017206 JSR PC, DMT ::DEAD MAN TIMER ROUTINE
926 (1) 007330 032777 100000 014524 BIT #BIT15,@CSR ::IS DE BIT SET
927 (1) 007336 001772 BEQ -12 ::TRY AGAIN
928 007340 005737 024210 TST FLAG.1 ::IS ERROR OCCURED?
929 007344 001405 BEQ 33$ ::BRANCH IF NO ERROR
930 007346 017737 014510 001122 MOV @CSR, BADA ::READ CURRENT STATUS OF CSR
931 007354 104011 ERROR+11 ::(DE) BIT FAIL TO SET IN CSR
932 007356 000733 BR 2$ ::TRY NEXT VALUE
```

```

923 007360 012777 000004 014474 33$: MOV #BIT2, @CSR      ;;READ YEAR VALUE
924 007366 017737 014472 001126      MOV @ARGO, BAD      ;;ACTUAL READ YEAR VALUE
925 007374 012737 000000 001124      MOV #0, GOOD       ;;EXPECTED VALUE
926 007402 023737 001126 001124      CMP BAD, GOOD      ;;BECAUSE INVALID YEAR VALUE
927 007410 001716                BEQ 2$              ;;GO GET ANOTHER VALUE
928 007412 017737 014444 001122      MOV @CSR, BADA     ;;READ CSR
929 007420 104001                ERROR+1           ;;YEAR VALUE DIDN'T MATCH WITH EXPECTED
930 007422 000711                BR 2$              ;;TRY NEXT VALUE

```

```

931
932 007424                3$:
(1) 007424 012737 077777 024052      MOV #77777,COUNTR  ;;DELAY LOOP SETUP
(1) 007432 004737 017206                JSR PC, DMT        ;;DEAD MAN TIMER ROUTINE
(1) 007436 032777 100000 014416      BIT #BIT15,@CSR   ;;IS DE BIT SET
(1) 007444 001772                BEQ -12            ;;TRY AGAIN
933 007446 005737 024210                TST FLAG.1        ;;IS ERROR ?
934 007452 001405                BEQ 4$             ;;BRANCH IF NOT
935 007454 017737 014402 001122      MOV @CSR, BADA     ;;GET CSR STATUS
936 007462 104011                ERROR+11          ;;(DE)BIT FAIL TO SET
937 007464 000670                BR 2$              ;;GET NEXT VALUE
938 007466 012777 000004 014366 4$: MOV #BIT2, @CSR   ;;READ YEAR
939 007474 017737 014364 001126      MOV @ARGO, BAD     ;;ACTUAL READ VALUE
940 007502 023737 001124 001126      CMP GOOD, BAD     ;;ARE THEY EQUAL?
941 007510 001656                BEQ 2$             ;;GO GET MORE
942 007512 017737 014344 001122      MOV @CSR, BADA     ;;READ STATUS OF CSR
943 007520 104001                ERROR+1           ;;READ YEAR VALUE NOT MATCH WITH EXPECTED
944 007522 000651                BR 2$              ;;GO CHECK FOR END OF TABLE

```

- TABLE1: 17073
- 16455
- 16454
- 17076
- 15474
- 17000
- 17477
- 17400
- 16430
- 17074
- 17077
- 20000
- 17075
- TABEN1: 00000

```

945
946 007524 017073
947 007526 016455
948 007530 016454
949 007532 017076
950 007534 015474
951 007536 017000
952 007540 017477
953 007542 017400
954 007544 016430
955 007546 017074
956 007550 017077
957 007552 020000
958 007554 017075
959 007556 000000
960
961
962
963
(3)
(3)
(2) 007560 000004
(1) 007562 012737 000010 001172
964 007570
965 007570 000240
966 007572 004737 016106
967 007576 104407
968 007600 004737 017000
(1) 007604 005737 024210
969 007610 001406

```

```

:
:*****
: *TEST 14 TEST VALID & INVALID VALUES OF MONTH+DAY
:*****
TST14: SCOPE
MOV #10,$TIMES ;;DO 10 ITERATIONS
TABDR2:
NOP
JSR PC, CNTLC ;;IS IT CONTROL C?
CKSWR
JSR PC, RDYBIT ;;CHECK FOR READY BIT SET
TST FLAG.1 ;;IS FLAG SET?
BEQ 1$ ;;BRANCH IF READY BIT SET

```

```

970 007612 017737 014244 001122      MOV @CSR, BADA      ;;CURRENT CSR
971 007620 104002                      ERROR+2             ;;READY BIT FAIL TO SET
972 007622 004737 016324              JSR PC, GETRDY     ;;GET READY BIT SET
973 007626 012700 010042              1$: MOV #TABLE2,R0  ;;POINTER TO THE TABLE2
974 007632 022700 010146              2$: CMP #TABEN2,R0  ;;IS END OF TABEN2
975 007636 003002                      BGT 22$            ;;BRANCH IF NOT
976 007640 000137 010150              JMP TABEN2+2       ;;EXIT
977 007644 012037 001124              22$: MOV (R0)+, GOOD  ;;MONTH+DAY VALUE FROM TABLE2
978 007650 005077 014206              CLR @CSR           ;;CLEAR CSR
979 007654 012777 100000 014200      MOV #BIT15,@CSR    ;;SET BIT #15 IN CSR TO WRITE MONTH+DAY
980 007662 013777 001124 014174      MOV GOOD, @RGO     ;;MONTH+DAY IN DEV REGISTER
981 007670 012737 077777 024052      MOV #77777, COUNTR ;;INITIALIZE COUNTER
982 007676 004737 017206              3$: JSR PC, DMT     ;;WAIT LOOP
983 007702 022777 100200 014152      CMP #BIT15+BIT7,@CSR ;;IS RDY & ERROR FLAG SET
984 007710 001372                      BNE 3$             ;;BRANCH BACK IF NOT
985 007712 017737 014152 024144      MOV @RG4, SEC      ;;READ SECOND
986 007720 012737 000001 016104      MOV #1, CONT3      ;;WAIT FOR 1 SECOND
987 007726 004737 016032              JSR PC, WAIT       ;;WAIT FOR COUPLE OF SECOND
988 007732 027737 014132 024144      CMP @RG4, SEC      ;;READ SEC AGAIN TO SEE ANY CHANGE
989 007740 001401                      BEQ 4$             ;;BRANCH IF CLOCK IS STOPPED
990 007742 104012                      ERROR+12           ;;CLOCK WAS NOT SUPPOSED TO BE RUNNING
991 007744 005737 024210              4$: TST FLAG.1     ;;IS ERROR FLAG SET
992 007750 001416                      BEQ 5$             ;;BRANCH IF NO ERROR
993 007752 017737 014106 001126      MOV @RGO, BAD      ;;ACTUAL VALUE READ
994 007760 012037 001124              MOV (R0)+, GOOD    ;;EXPECTED VALUE
995 007764 023737 001124 001126      CMP GOOD, BAD      ;;ARE THEY EQUAL?
996 007772 001717                      BEQ 2$             ;;YES, GO GET MORE VALUE
997 007774 017737 014062 001122      MOV @CSR, BADA     ;;GET STATUS OF CSR
998 010002 104001                      ERROR+1            ;;READ MONTH+DAY NOT MATCH

```


1011	010046	000020	0020
1012	010050	000000	0000
1013	010052	000400	0400
1014	010054	000400	0400
1015	010056	000402	0402
1016	010060	000402	0402
1017	010062	000432	0432
1018	010064	000432	0432
1019	010066	001035	1035
1020	010070	001000	1000
1021	010072	001440	1440
1022	010074	001400	1400
1023	010076	002037	2037
1024	010100	002000	2000
1025	010102	002440	2440
1026	010104	002400	2400
1027	010106	003037	3037
1028	010110	003000	3000
1029	010112	003440	3440
1030	010114	003400	3400
1031	010116	004040	4040
1032	010120	004000	4000
1033	010122	004437	4437
1034	010124	004400	4400
1035	010126	005040	5040
1036	010130	005000	5000
1037	010132	005437	5437
1038	010134	005400	5400
1039	010136	006040	6040
1040	010140	006000	6000
1041	010142	006024	6024
1042	010144	006024	6024
1043	010146	000000	0000

TABEN2: 0000

```

: THIS ROUTINE CHECKS VALID AND INVALID HOUR+MIN.
: IF HOUR+MIN ARE INVALID IT WRITES ANOTHER VALUE OF HOUR+MIN UNTIL
: CORRECT VALUE OF HOUR+MINUTE IS WRITTEN. ONCE CORRECT VALUE IS
: WRITTEN YOU CAN'T WRITE ANOTHER VALUE OF HOUR+MIN WITHOUT WRITING
: VALID VALUE IN MONTH AND DAY REGISTER

```

TABDR3:

1051	010150								
1052	010150	004737	016106			JSR	PC,	CNTLC	::
1053	010154	104407				CKSWR			
1054	010156	004737	017000			JSR	PC,	RDYBIT	::CHECK FOR READY BIT SET
(1)	010162	005737	024210			TST	FLAG.1		::IS FLAG SET?
1055	010166	001406				BEQ	11\$::BRANCH IF RDYBIT SET
1056	010170	017737	013666	001122		MOV	@CSR,	BADA	::GET STATUS OF CSR
1057	010176	104002				ERROR+2			::RDYBIT FAIL TO SET
1058	010200	004737	016324			JSR	PC,	GETRDY	::GET READY BIT SET
1059	010204	012700	010456		11\$:	MOV	#TABLE3,RO		::POINTER TO THE TABLE
1060	010210	022700	010516		1\$:	CMP	#TABEN3,RO		::IS IT END OF TABLE
1061	010214	003023				BGT	2\$::BRANCH IF NOT END OF TABLE
1062	010216	012777	100000	013636		MOV	#BIT15,@CSR		::SET HOUR+MIN
1063	010224	012777	000000	013634		MOV	#000000,@RG2		::LAST VALUE OF HOUR+MIN BEFORE
1064									::EXIT
1065	010232	004737	017000			JSR	PC,	RDYBIT	::CHECK FOR READY BIT SET


```
1120 010512 006007          6007          ::HOURL=12,MIN=7
1121 010514 006007          6007          ::EXPECTED HOUR=12,MIN=7
1122 010516 000000          TABEN3: 00000
1123
1124
1125 ;THIS ROUTINE WRITES VALUES IN DEV REGISTER RANDOMLY
1126 ;THAT IS INSTEAD OF WRITING IN MONTH+DAY REGISTER
1127 ;WRITE IN HOUR+MIN REGISTER & INSTEAD OF WRITING IN
1128 ;HOUR+MIN REG,WRITE IN SECOND REGISTER
1129 ;*****
(3) ;*TEST 15 TEST RANDOMLY WRITE VALUE IN DEVICE REGISTERS
(3) ;*****
(2) 010520 000004          TST15: SCOPE
(1) 010522 012737 000012 001172          MOV #10.,$TIMES ;:DO 10. ITERATIONS
1130 010530 004737 016106          RANDOM: JSR PC, CNTLC
1131 010534 104407          CKSWR
1132 010536 005077 013320          CLR @CSR ;:CLEAR CSR
1133 010542 012777 100000 013312          MOV #BIT15,@CSR ;:SET BIT #15 IN CSR TO WRITE
1134 ;:MONTH+DAY
1135 010550 012777 005005 013310          MOV #5005, @RG2 ;:HOUR+MIN REGISTER
1136 010556 004737 017000          JSR PC, RDYBIT ;:WAIT FOR READY BIT SET
1137 010562 017737 013302 024144          MOV @RG4, SEC ;:READ PRESENT SECOND VALUE
1138 010570 012737 000002 016104          MOV #2, CONT3 ;:WAIT FOR A SECOND
1139 010576 004737 016032          JSR PC, WAIT ;:WAIT FOR 2 SECOND
1140 010602 027737 013262 024144          CMP @RG4, SEC ;:ARE THEY EQUAL?
1141 010610 001001          BNE 1$
1142 010612 104012          ERROR+12 ;:CLOCK WAS SUPPOSE TO BE RUNNING
1143 010614 005737 024210 1$: TST FLAG.1 ;:IS ERROR FLAG SET?
1144 010620 001004          BNE 11$
1145 010622 017737 013234 001122          MOV @CSR, BADA
1146 010630 104025          ERROR+25 ;:RDY BIT WAS NOT SUPPOSE TO BE SET
1147 010632 012777 004010 013230 11$: MOV #4010, @RG4 ;:WRITE HOUR+MIN VALUE IN SEC REG
1148 010640 004737 017000          JSR PC, RDYBIT ;:CHECK FOR READY BIT SET
(1) 010644 005737 024210          TST FLAG.1 ;:IS FLAG SET?
1149 010650 001404          BEQ 2$
1150 010652 017737 013204 001122          MOV @CSR, BADA ;:
1151 010660 104002          ERROR+2 ;:ERROR AS FLAG DIDN'T SET
1152 010662 012777 100000 013172 2$: MOV #BIT15,@CSR ;:WRITE MONTH+DAY
1153 010670 012777 006004 013166          MOV #6004, @RG0 ;:BUT WRITE IN RG0(MONTH+DAY)REGISTER
1154 010676 004737 017000          JSR PC, RDYBIT ;:CHECK FOR READY BIT SET
(1) 010702 005737 024210          TST FLAG.1 ;:IS FLAG SET?
1155 010706 017737 013156 024144          MOV @RG4, SEC ;:READ SECOND VALUE
1156 010714 012737 000002 016104          MOV #2, CONT3 ;:WAIT FOR SEC
1157 010722 004737 016032          JSR PC, WAIT ;:WAIT FOR 1 SECOND
1158 010726 027737 013136 024144          CMP @RG4, SEC ;:ARE THEY EQUAL?
1159 010734 001401          BEQ 3$
1160 010736 104012          ERROR+12 ;:CLOCK WAS NOT SUPPOSE TO BE RUNNING
1161 010740 012777 100000 013114 3$: MOV #BIT15, @CSR ;:SET BIT TO WRITE HOUR+MIN
1162 010746 012777 006004 013112          MOV #6004, @RG2 ;:NOW WRITE IN VALID REGISTER
1163 010754 004737 017000          JSR PC, RDYBIT ;:CHECK FOR READY BIT SET
(1) 010760 005737 024210          TST FLAG.1 ;:IS FLAG SET?
1164 010764 017737 013100 024144          MOV @RG4, SEC ;:READ SECOND
1165 010772 012737 000002 016104          MOV #2, CONT3 ;:WAIT FOR SEC
1166 011000 004737 016032          JSR PC, WAIT ;:WAIT FOR 1 SEC
1167 011004 027737 013060 024144          CMP @RG4, SEC ;:ARE THEY EQUAL?
1168 011012 001001          BNE 4$ ;:YES, MEANS CLOCK RUNNING
```

```

1169 011014 104012          ERROR+12          ;;ERROR CLOCK WAS SUPPOSE TO BE RUNNING
1170 011016 005737 024210 4$:  TST      FLAG.1
1171 011022 001406          BEQ      5$
1172 011024 017737 013032 001122  MOV     @CSR,  BADA
1173 011032 104002          ERROR+2
1174 011034 004737 016324 5$:  JSR     PC,    GETRDY
1175 011040
1176
1177
1178
(3)
;*****
;*TEST 16      TEST ROUTINE TO CHECK CLOCK KEEPS CORRECT TIME
;*****
(3)
(2) 011040 000004          TST16: SCOPE
(1) 011042 012737 000012 001172  MOV     #10, $TIMES      ;;DO 10. ITERATIONS
1179 011050 004737 016106  BASIC: JSR     PC,      CNTLC      ;;
1180 011054 104407          CKSWR
1181 011056 012700 011220  MOV     #BASTAB,R0      ;;POINTER TO THE TABLE
1182 011062 022700 011422 1$:  CMP     #BASEND,R0      ;;IS IT END OF TABLE?
1183 011066 003002          BGT     2$              ;;BRANCH IF NOT END OF TABLE
1184 011070 000137 011424  JMP     BASEND+2        ;;EXIT
1185 011074 005077 012762 2$:  CLR     @CSR            ;;CLEAR CSR
1186 011100 012002          MOV     (R0)+, R2       ;;MASK FOR CSR
1187 011102 012004          MOV     (R0)+, R4       ;;ADDRESS OF CSR
1188 011104 010274 000000  MOV     R2,    @(R4)    ;;MASK IN CSR
1189 011110 012002          MOV     (R0)+, R2       ;;VALUE TO BE SET IN DEV REG
1190 011112 012004          MOV     (R0)+, R4       ;;ADDRESS OF DEV REG
1191 011114 010274 000000  MOV     R2,    @(R4)    ;;VALUE IN DEV REG
1192 011120 004737 017000  JSR     PC,      RDYBIT  ;;CHECK FOR READY BIT SET
(1) 011124 005737 024210  TST     FLAG.1          ;;IS FLAG SET?
1193 011130 001403          BEQ     12$            ;;BRANCH IF SET
1194 011132 104002          ERROR+2
1195 011134 004737 016324  JSR     PC,      GETRDY  ;;GO GET READY BIT
1196 011140 022027 177777 12$:  CMP     (R0)+, #-1     ;;IS IT END OF SUBSET?
1197 011144 001346          BNE     1$            ;;GET NEXT VALUE FROM TABLE
1198
1199 011146 005077 012710 3$:  CLR     @CSR            ;;CLEAR CSR
1200 011152 012002          MOV     (R0)+, R2       ;;MASK TO READ DEV REG
1201 011154 012004          MOV     (R0)+, R4       ;;ADDRESS OF CSR
1202 011156 010274 000000  MOV     R2,    @(R4)    ;;SET BIT IN CSR TO READ
1203 011162 012037 001124  MOV     (R0)+, GOOD     ;;EXPECTED VALUE IN GOOD
1204 011166 012004          MOV     (R0)+, R4       ;;ADDRESS OF DEV REG
1205 011170 017437 000000  MOV     @{R4}, BAD      ;;ACTUAL VALUE READ
1206 011176 023737 001124 001126  CMP     GOOD,  BAD      ;;EXPECTED=ACTUAL?
1207 011204 001401          BEQ     4$            ;;BRANCH IF EQUAL
1208 011206 104001          ERROR+1
1209 011210 022027 177777 4$:  CMP     (R0)+, #-1     ;;IS IT END OF SUBSET?
1210 011214 001354          BNE     3$            ;;BRANCH IF NOT
1211 011216 000721          BR      1$            ;;IS THERE ANY MORE VALUES
1212
1213 011220 040000          BASTAB: BIT14          ;;BIT TO SET YEAR VALUE
1214 011222 024062          CSR            ;;ADDRESS OF CSR
1215 011224 017073          17073          ;;YEAR = 1979
1216 011226 024064          RGO            ;;ADDRESS OF DEVICE REGISTER
1217 011230 000000          000000        ;;NOT END OF SUBSET
1218
1219 011232 100000          BIT15          ;;BIT TO SET MONTH+DAY

```


1220	011234	024062	CSR	::ADDRESS OF CSR
1221	011236	006037	6037	::DECEMBER 31
1222	011240	024064	RG0	::ADDRESS OF DEV REG
1223	011242	000000	000000	::NOT END OF SUBSET
1224				
1225	011244	100000	BIT15	::BIT TO SET HOUR+MIN
1226	011246	024062	CSR	::ADDRESS OF CSR
1227	011250	006000	6000	::HOUR=12,MIN=0
1228	011252	024066	RG2	::ADDRESS OF REGISTER
1229	011254	177777	177777	::END OF SUBSET
1230				
1231	011256	000004	BIT2	::BIT TO READ YEAR VALUE
1232	011260	024062	CSR	::ADDRESS OF CSR
1233	011262	017073	17073	::EXPECTED YEAR VALUE
1234	011264	024064	RG0	::ADDRESS OF DEV REG
1235	011266	000000	000000	::NOT END OF SUBSET
1236				
1237	011270	000002	BIT1	::TO READ JULIAN DAY
1238	011272	024062	CSR	::ADDRESS OF CSR
1239	011274	002455	2455	::EXPECTED JULIAN DAY
1240	011276	024064	RG0	::ADDRESS OF DEV REG
1241	011300	000000	000000	::NOT END OF SUBSET
1242				
1243	011302	000001	BIT0	::TO READ HOR+MIN
1244	011304	024062	CSR	::ADDRESS OF CSR
1245	011306	006000	6000	::EXPECTED HOUR+MIN
1246	011310	024066	RG2	::ADDRESS OF DEV REG
1247	011312	177777	177777	::END OF SUBSET
1248				
1249	011314	040000	BIT14	::SET YEAR
1250	011316	024062	CSR	::ADDRESS OF CSR
1251	011320	017074	17074	::1980
1252	011322	024064	RG0	::ADDRESS OF DEVICE REG
1253	011324	000000	000000	::NOT END OF SUBSET
1254	011326	100002	BIT15+BIT1	::SET JULIAN DAY
1255	011330	024062	CSR	
1256	011332	000001	001	::JULIAN DAY (JAN 1)
1257	011334	024064	RG0	::ADDRESS OF DEV REG
1258	011336	000000	000000	::NOT END OF SUBSET
1259				
1260	011340	100000	BIT15	::SET HOUR+MIN
1261	011342	024062	CSR	
1262	011344	000000	000000	::0HOUR,0MIN
1263	011346	024066	RG2	::ADDRESS OF REG
1264	011350	177777	177777	::END OF SUBSET
1265				
1266	011352	000002	BIT1	::READ JULIAN DAY
1267	011354	024062	CSR	
1268	011356	000001	1	::EXPECTED VALUE
1269	011360	024064	RG0	::ADDRESS OF DEV REG
1270	011362	000000	000000	::NOT END OF SUBSET
1271				
1272	011364	000001	BIT0	::READ MONTH+DAY
1273	011366	024062	CSR	
1274	011370	000401	401	::JAN 1 31
1275	011372	024064	RG0	::ADDRESS OF DEV REG

```
1276 011374 000000 000000 ::NOT END OF SUBSET
1277
1278 011376 000001 BIT0 ::READ HOUR+MIN
1279 011400 024062 CSR
1280 011402 000000 000000 ::EXPECTED HOUR+MIN
1281 011404 024066 RG2
1282 011406 000000 000000 ::NOT END OF SUBSET
1283
1284 011410 000001 BIT0 ::READ SECOND
1285 011412 024062 CSR ::ADD OF CSR
1286 011414 000000 000000 ::EXPECTED SECOND VALUE
1287 011416 024070 RG4 ::ADDRESS OF SECOND REG
1288 011420 177777 177777 ::END OF SUBSET
1289 011422 000000 BASEND: 000000 ::END OF TABLE
1290
1291
1292
1293
1294
```

;;THIS ROUTINE CHECKS IF INTERRUPT OCCURS PROPERLY

```
1295
1296
1297
1298
(3)
(3)
(2) 011424 000004
(1) 011426 012737 000012 001172
1299 011434 013737 024106 177776 TABDR4: MOV #10, $TIMES ;;DO 10. ITERATIONS
1300 011442 004737 016106 JSR DPRI, PSW
1301 011446 104407 JSR PC, CNTLC
1302 011450 012700 012152 CKSWR
1303 011454 022700 012212 1$: MOV #TABLE4, R0 ;;POINTER TO THE STARTING ADDRESS OF THE TABLE4
1304 011460 003002 CMP #TABEN4, R0 ;;IS IT END OF TABEN5?
1305 011462 000137 012114 BGT 2$ ;;BRANCH IF NOT
1306 JMP 8$ ;;EXIT
1307 011466 012737 000001 024226 2$: MOV #1, INTREG ;;FLAG TO INDICATE UPDATE INTRPT
1308 011474 012037 001124 MOV (R0)+, GOOD ;;UPDATE VALUE
1309 011500 005077 012356 CLR @CSR ;;CLEAR CSR
1310 011504 005037 024212 CLR FLAG.2 ;;CLEAR FLAG
1311 011510 012777 100100 012344 MOV #BIT15+BIT6, @CSR ;;INTERRUPT ENABLED
1312 011516 013777 001124 012340 MOV GOOD, @RG0 ;;MONTH+DAY VALUE IN DEV REG
1313 011524 005037 024120 CLR ACSR ;;CLR LOCATION
1314 011530 012737 000005 016030 MOV #5, TIMES ;;WAIT FOR APPROX 5 SEC
1315 011536 004737 015726 JSR PC, INTRTN ;;WAIT FOR INTERRUPT
1316 011542 005737 024212 TST FLAG.2 ;;IS INTERRUPT OCCURED?
1317 011546 001011 BNE 3$ ;;BRANCH ON INTERRUPT
1318 011550 017737 012306 001122 MOV @CSR, BADA ;;GET STATUS OF CSR
1319 011556 104014 ERROR+14 ;;INTERRUPT DIDN'T OCCUR WHILE
1320 ;;UPDATING CLOCK WITH INTENABLE
1321 011560 005077 012276 CLR @CSR ;;DISABLE INTERRUPT
1322 011564 005720 TST (R0)+ ;;POP UP PONITR
1323 011566 000137 011454 JMP 1$ ;;GET NEXT VALUE
1324 011572 005037 024212 3$: CLR FLAG.2 ;;CLEAR INTERRUPT FLAG
1325 011576 004737 017000 JSR PC, RDYBIT ;;CHECK FOR READY BIT SET
(1) 011602 005737 024210 TST FLAG.1 ;;IS FLAG SET?
1326 011606 001417 BEQ 4$ ;;BRANCH IF YES
```

1327	011610	012037	001124		MOV	(R0)+,	GOOD		:: EXPECTED VALUE IN GOOD
1328	011614	017737	012244	001126	MOV	@RG0,	BAD		:: READ MONTH+DAY
1329	011622	023737	001124	001126	CMP	GOOD,	BAD		:: ARE THEY EQUAL?
1330	011630	001711			BEQ	1\$:: YES, BRANCH
1331	011632	013737	024120	001122	MOV	ACSR,	BADA		:: GET STATUS OF CSR
1332	011640	104001			ERROR+1				:: READ VALUE NOT MATCH WITH EXPECTED
1333	011642	000137	011454		JMP	1\$:: GET MORE VALUE
1334	011646	012037	001124		4\$: MOV	(R0)+,	GOOD		:: EXPECTED VALUE
1335	011652	017737	012206	001126	MOV	@RG0,	BAD		:: ACTUAL VALUE IN BAD
1336	011660	023737	001124	001126	CMP	GOOD,	BAD		:: ARE THEY EQUAL?
1337	011666	001406			BEQ	5\$:: BRANCH IF EQUAL
1338	011670	013737	024120	001122	MOV	ACSR,	BADA		:: GET STATUS OF CSR
1339	011676	104001			ERROR+1				:: READ VALUE NOT MATCH WITH
1340									:: EXPECTED
1341	011700	000137	011454		JMP	1\$:: GET NEXT VALUE
1342	011704	012037	001124		5\$: MOV	(R0)+,	GOOD		:: HOUR+MIN IN GOOD
1343	011710	005077	012146		CLR	@CSR			:: CLEAR CSR
1344	011714	005037	024212		CLR	FLAG.2			::
1345	011720	012777	100100	012134	MOV	#BIT15+BIT6,@CSR			:: INTERRUPT ENABLED
1346	011726	013777	001124	012132	MOV	GOOD,	@RG2		:: HOUR+MIN IN DEV REG
1347	011734	005037	024120		CLR	ACSR			::
1348	011740	012737	000005	016030	MOV	#5,	TIMES		:: 5 SEC
1349	011746	004737	015726		JSR	PC,	INTRTN		:: WAIT FOR INTERRUPT
50	011752	005737	024212		TST	FLAG.2			:: IS INTERRUPT OCCURED?
51	011756	001011			BNE	6\$:: BRANCH IF YES
1352	011760	017737	012076	001122	MOV	@CSR,	BADA		:: READ STATUS OF CSR
1353	011766	104014			ERROR+14				:: INTERRUPT DIDN'T OCCUR WHILE
1354									:: UPDATING CLOCK
1355	011770	005077	012066		CLR	@CSR			:: UNABLE INTERRUPT
1356	011774	005720			TST	(R0)+			:: POP UP POINTER
1357	011776	000137	011454		JMP	1\$::
1358	012002	005037	024212		6\$: CLR	FLAG.2			:: CLEAR INTERRUPT FLAG
1359	012006	004737	017000		JSR	PC,	RDYBIT		:: CHECK FOR READY BIT SET
(1)	012012	005737	024210		TST	FLAG.1			:: IS FLAG SET?
1360	012016	001417			BEQ	7\$:: BRANCH IF YES
1361	012020	012037	001124		MOV	(R0)+,	GOOD		:: EXPECTED VALUE OF HOUR+MIN
1362	012024	017737	012036	001126	MOV	@RG2,	BAD		:: READ ACTUAL HOUR+MIN
1363	012032	023737	001124	001126	CMP	GOOD,	BAD		:: ARE THEY EQUAL?
1364	012040	001721			BEQ	5\$:: GO GET MORE VALUES
1365	012042	013737	024120	001122	MOV	ACSR,	BADA		:: GET CSR STATUS
1366	012050	104001			ERROR+1				:: READ VALUE NOT MATCH WITH EXPECTED
1367	012052	000137	011704		JMP	5\$:: GET MORE VALUE
1368	012056	012037	001124		7\$: MOV	(R0)+,	GOOD		:: EXPECTED HOUR+MIN
1369	012062	017737	012000	001126	MOV	@RG2,	BAD		:: READ HOUR+MIN
1370	012070	023737	001124	001126	CMP	GOOD,	BAD		:: ARE THEY EQUAL?
1371	012076	001404			BEQ	11\$:: GET NEXT VALUE
1372	012100	013737	024120	001122	MOV	ACSR,	BADA		:: GET STATUS OF CSR
1373	012106	104001			ERROR+1				:: READ VALUE NOT MATCH WITH EXPECTED
1374	012110	000137	011454		11\$: JMP	1\$::
1375	012114	005037	024226		8\$: CLR	INTREG			:: CLR UPDATE INT FLAG
1376	012120	004737	017000		JSR	PC,	RDYBIT		:: CHECK FOR READY BIT SET
(1)	012124	005737	024210		TST	FLAG.1			:: IS FLAG SET?
1377	012130	001002			BNE	9\$:: BRANCH IF READY DIDN'T SET
1378	012132	000137	012214		JMP	TABEN4+2			:: EXIT TO NEXT ROUTINE
1379	012136	017737	011720	001122	9\$: MOV	@CSR,	BADA		:: GET STATUS OF CSR
1380	012144	104001			ERROR+1				:: READ VALUE NOT MATCH WITH EXPCTED

```

1381 012146 000137 012214      JMP      TABEN4+2      ;;EXIT
1382 012152 000401      TABLE4: 00401
1383 012154 000401      00401
1384 012156 000000      00000
1385 012160 000000      00000
1386 012162 006002      06002
1387 012164 006002      06002
1388 012166 002004      02004
1389 012170 002004      02004
1390 012172 004000      04000
1391 012174 004000      04000
1392 012176 004002      04002
1393 012200 004002      04002
1394 012202 014400      14400
1395 012204 000000      00000
1396 012206 006020      06020
1397 012210 006020      06020
1398 012212 000000      TABEN4: 00000
1399
1400
1401
1402
1403
1404
1405
1406
1407
  
```

```

: THIS ROUTINE PRE-SETS INTERRUPT FOR
: OHOUR,OMINUTE,0 SECOND OF THE
: FIRST DAY OF THE MONTH
:
:
: *****
: *TEST 20      TEST PRE-SET INTERRUPT
: *****
  
```

```

(3)
(3)
(2) 012214 000004      TST20: SCOPE
(1) 012216 012737 000001 001172      MOV      #1,STIMES      ;;DO 1 ITERATION
1408 012224 013737 024106 177776      PRESET: MOV      DPRI, PSW
1409 012232 004737 016106      JSR      PC, CNTLC      ;;IS IT CONROL C?
1410 012236 104407      CKSWR
1411 012240 012700 013064      MOV      #TABLE5,R0      ;;PONTER TO THE STARTING ADDRESS OF THE TABLE
1412 012244 012703 013134      MOV      #TABEN5,R3      ;;R3 CONTAINS END ADDRESS OF TABLE
1413 012250 012737 000002 024226      MOV      #2, INTREG      ;;FLAG TO INDIACATE PRESET INTRPT
1414 012256 004737 016560      JSR      PC, INIPRE      ;;GET 'M' BIT SET
1415 012262 012777 000000 011572 1$: MOV      #0, @CSR      ;;CLEAR CSR
1416 012270 012777 040000 011564      MOV      #BIT14,@CSR      ;;SET (SY) BIT IN CSR TO WRITE A YEAR VALUE
1417 012276 012777 017070 011560      MOV      #17070,@RGO      ;;YEAR 1976 IN DEV REG
1418 012304 004737 017040      JSR      PC, RDYCK1
1419 012310 005737 024210      TST      FLAG.1
1420 012314 001406      BEQ      2$,      ;;BRANCH IF NOT
1421 012316 017737 011540 001122      MOV      @CSR, BADA
1422 012324 104006      ERROR+6      ;;READY BIT DIDN'T SET PROBABLE
1423      ;;CAUSE, YEAR VALUE MAY BE INVALID
1424 012326 004737 016324      JSR      PC, GETRDY      ;;GET READY BIT SET
1425 012332 012777 000000 011522 2$: MOV      #0, @CSR
1426 012340 012077 011516      MOV      (R0)+, @CSR      ;;GET MASK TO SET IN CSR
1427 012344 012077 011514      MOV      (R0)+, @RGO      ;;VALUE IN MONTH + DAY REG
1428 012350 004737 017040      JSR      PC, RDYCK1
1429 012354 005737 024210      TST      FLAG.1
1430 012360 001411      BEQ      3$,      ;;BRANCH IF READY BIT SET
1431 012362 017737 011474 001122      MOV      @CSR, BADA      ;;READ CSR
1432 012370 104007      ERROR+7      ;;ERROR: READY BIT DIDN'T SET
  
```

1433										::PROBABLE CAUSE OF AN ERROR
1434										::ONE OR BOTH VALUES OF MONTH
1435										::+DAY MAY BE INVALID
1436	012372	005720								
1437	012374	005720								
1438	012376	005720								
1439	012400	000137	013024							::ADJUST POINTER
1440	012404	012777	000000	011450	3\$:	TST	(R0)+			
1441	012412	012777	100000	011442		JMP	11\$			
1442	012420	012777	013473	011440		MOV	#0, @CSR			
1443	012426	004737	017040			MOV	#BIT15, @CSR			::WRITE HOUR+MIN
1444	012432	005737	024210			MOV	#13473, @ARG2			::HOUR+MIN IN REG2
1445	012436	001404				JSR	PC, RDYCK1			
1446	012440	017737	011416	001122		TST	FLAG.1			
1447	012446	104010				BEQ	4\$::BRANCH IF NOT
1448						MOV	@CSR, BADA			::GET CSR STATUS
1449						ERROR+10				::ERROR:ONE OR BOTH VALUES
1450	012450	012777	000000	011404	4\$:	MOV	#0, @CSR			::OF HOUR+MINUTE MAY BE
1451	012456	012077	011400			MOV	(R0)+, @CSR			::INVALID
1452										
1453	012462	012077	011376			MOV	(R0)+, @ARG0			::CLEAR CSR, OR SET INTRPT
1454	012466	004737	017120			JSR	PC, RDYCK			::FOR JULIAN DAY IF BIT1 IS SET
1455	012472	005737	024210			TST	FLAG.1			::PRE-SET INTERRUPT
1456	012476	001406				BEQ	5\$::GO CHECK FOR READY AND DE BIT
1457	012500	017737	011356	001122		MOV	@CSR, BADA			::IS 'DE' AND 'RDY' BIT SET?
1458	012506	104011				ERROR+11				::BRANCH IF (DE AND RDY)BIT SETS
1459	012510	005720				TST	(R0)+			::GET CSR STATUS
1460	012512	000544				BR	11\$::(DE) BIT FAIL TO SET
1461	012514	012777	000000	011340	5\$:	MOV	#0, @CSR			::ADJUST POINTER
1462	012522	012777	100200	011336		MOV	#100200, @ARG2			::EXIT
1463	012530	004737	017120			JSR	PC, RDYCK			::PRE-SET INTERRUPT FOR HOUR+MIN
1464	012534	005737	024210			TST	FLAG.1			::CHECK FOR RDY AND DE BIT
1465	012540	001406				BEQ	6\$::IS ('DE' AND 'RDY') SET?
1466	012542	017737	011314	001122		MOV	@CSR, BADA			::BRANCH IF SET
1467	012550	104011				ERROR+11				::GET CSR
1468	012552	005720				TST	(R0)+			::(DE) BIT FAIL TO SET
1469	012554	000523				BR	11\$::ADJUST POINTER
1470	012556	012777	000000	011276	6\$:	MOV	#0, @CSR			::GET NEXT VALUE
1471	012564	012777	100000	011276		MOV	#100000, @ARG4			::PRE-SET INTERRUPT FOR 0 SEC
1472	012572	004737	017040			JSR	PC, RDYCK1			
1473	012576	005737	024210			TST	FLAG.1			::IS FLAG BIT SET?
1474	012602	001406				BEQ	7\$::BRANCH IF (DE,RDY,MBIT) BIT SET
1475	012604	017737	011252	001122		MOV	@CSR, BADA			::GET CSR STATUS
1476	012612	104026				ERROR+26				::PROBABLE CAUSE OF AN ERROR:
1477										::'M'BIT FAIL TO SET CHECK
1478										::CSR TO FIND EXACT CAUSE OF AN ERROR
1479	012614	005720				TST	(R0)+			::ADJUST POINTER
1480	012616	000502				BR	11\$::GET NEXT VALUE
1481	012620	005077	011236		7\$:	CLR	@CSR			
1482	012624	005037	024212			CLR	FLAG.2			::CLEAR INTERRUPT FLAG
1483	012630	005037	024120			CLR	ACSR			::
1484	012634	012777	000100	011220		MOV	#BIT6, @CSR			::NOW ENABLE INTERRUPT
1485	012642	012737	000120	016030		MOV	#120, TIMES			::APPRO 80 SEC
1486	012650	004737	015726			JSR	PC, INTRTN			::WAIT FOR INTERRUPT
1487	012654	005737	024212			TST	FLAG.2			::IS INTERRUPT OCCURED?
1488	012660	001006				BNE	8\$::BRANCH IF YES

1489	012662	017737	011174	001122		MOV @CSR, BADA	::GET STATUS OF CSR
1490	012670	104015				ERROR+15	::INTERRUPT DIDN'T OCCUR ON
1491							::PRESET VALUE
1492	012672	005077	011164			CLR @CSR	::CLEAR INTERRUPT ENABLE
1493	012676	005037	024212		8\$:	CLR FLAG.2	::CLEAR FLAG.2
1494	012702	017737	011156	001126		MOV @RG0, BAD	::READ MONTH+DAY
1495	012710	012037	001124			MOV (R0)+, GOOD	::EXPECTED HOUR+MIN
1496	012714	023737	001124	001126		CMP GOOD, BAD	::ARE THEY EQUAL?
1497	012722	001404				BEQ 9\$::YES, BRANCH
1498	012724	013737	024120	001122		MOV ACSR, BADA	::GET STATUS OF CSR AT TIME
1499							::OF INTERRUPT
1500	012732	104001				ERROR+1	::READ VALUE NOT MATCH
1501							::WITH EXPECTED VALUE OF
1502							::MONTH+DAY
1503	012734	017737	011126	001126	9\$:	MOV @RG2, BAD	::READ HOUR+MIN
1504	012742	012737	000000	001124		MOV #00, GOOD	::EXPECTED HOUR+MIN
1505	012750	023737	001124	001126		CMP GOOD, BAD	::EQUAL?
1506	012756	001404				BEQ 10\$::GO TO READ SECOND
1507	012760	013737	024120	001122		MOV ACSR, BADA	::GET CSR STATUS AT TIME OF INTERRUPT
1508	012766	104001				ERROR+1	::READ VALUE OF HOUR+MINUTE
1509							::DIDN'T MATCH WITH EXPECTED VALUE
1510	012770	017737	011074	001126	10\$:	MOV @RG4, BAD	::READ SECOND
1511	012776	012737	000000	001124		MOV #0, GOOD	::EXPECTED SECOND VALUE
1512	013004	023737	001124	001126		CMP GOOD, BAD	::ARE THEY EQUAL?
1513	013012	001404				BEQ 11\$::GET MORE VALUE
1514	013014	013737	024120	001122		MOV ACSR, BADA	::GET CSR AT TIME OF INTERRUPT
1515	013022	104001				ERROR+1	::READ VALUE NOT EQUAL TO EXPECTED
1516							::VALUE OF SECOND
1517	013024	004737	016106		11\$:	JSR PC, CNTLC	::IS IT CONTROL C?
1518	013030	020003				CMP R0, R3	::IS IT END OF TABLE?
1519	013032	002412				BLT 12\$::BRANCH IF NOT
1520	013034	005037	024226			CLR INTREG	::CLEAR INTERRUPT SWITCH
1521	013040	012777	000000	011016		MOV #0, @RG0	::JUST TO CLEAR 'M' BIT
1522	013046	012737	004002	020160		MOV #LOGTST, \$RTNAD	::
1523	013054	000137	020060			JMP \$EOP	::
1524	013060	000137	012332		12\$:	JMP 2\$::GET MORE VALUES
1525							
1526							
1527	013064	100002			TABLE5:	100002	::MASK TO SET JULIAN DAY
1528	013066	000074				000074	:: SET FEB-29
1529	013070	000002				0000002	::JULIAN BIT SET
1530	013072	100075				100075	::MARCH-1 (INTRPT ENBLE)
1531	013074	001401				001401	::MARCH-1
1532							
1533	013076	100000				100000	
1534	013100	004436				004436	::SEP-30
1535	013102	000000				000000	
1536	013104	105201				105201	::OCT-1 (INTRPT ENABLE)
1537	013106	005001				005001	::OCT-1
1538							
1539	013110	100002				100002	::SET JULIAN DAY
1540	013112	002456				002456	::DEC-31
1541	013114	000002				000002	::SET JULIAN BIT
1542	013116	100001				100001	::JAN-1 (INTRPT ENABLE)
1543	013120	000401				000401	::JAN-1
1544							

```

1545 013122 100000          100000
1546 013124 001034          001034          ::FEB-28
1547 013126 000002          000002
1548 013130 100074          100074          ::MARCH-1(INTRPT ENABLE)
1549 013132 001401          001401          ::MARCH-1
1550 013134 000000          TABENS: 000000          ::END OF TABLES
1551
1552          .SBTTL SET TIME OF THE CLOCK
1553
1554
1555
1556          ;THIS ROUTINE SETS NEW TIME IN THE CLOCK
1557          ;
1558 013136          ;
1559 (1) 013136 104401 013144          NEWTIM:
1560 (1) 013142 000444          TYPE      .65$          ::TYPE ASCIZ STRING
1561 (1)          BR      64$          ::GET OVER THE ASCIZ
1562 (1)          ;:65$: .ASCIZ <200>/TYPE "CR" AT THE END OF THE FOLLOWING LINE TO END ENTERING TIME VA
1563 (1)          64$:          TYPE      .67$          ::TYPE ASCIZ STRING
1564 (1) 013254 104401 013262          BR      66$          ::GET OVER THE ASCIZ
1565 (1) 013260 000403          ;:67$: .ASCIZ <200>/NEW/
1566 (1)          66$:          TYPE      .69$          ::TYPE ASCIZ STRING
1567 (1) 013270 104401 013276          BR      68$          ::GET OVER THE ASCIZ
1568 (1) 013274 000412          ;:69$: .ASCIZ <200>/YYYY,MO,DD,HH,MI/<200>
1569 (1) 013322          68$:
1570 013322 012702 024254          1$:  MOV      #DIGIT, R2          ::POINTER TO DIGIT TABLE
1571 013326 005037 024122          CLR      OCTNUM          ::INITIALIZE OCTNUM
1572 013332 005037 024124          CLR      COMMA          ::CLEAR COMMA COUNT
1573 013336 005037 024210          CLR      FLAG.1          ::CLEAR FLAG
1574 013342 005037 024126          CLR      DIGCNT          ::INITIALIZE DIGIT COUNT
1575 013346 005037 024230          CLR      SOFSWT          ::CLEAR SOFTWARE SWITCH
1576 013352 005037 024172          CLR      SWITCH1          ::CLEAR SWITCH1
1577
1578 013356 104410          2$:  RDCHR          (SP)+, R4          ::STORE CHARACTER
1579 013360 012604          MOV      R4, ECHOB1          ::
1580 013362 110437 024250          TYPE          ECHOB1          ::ECHO THE CHARACTER
1581 013366 104401 024250          CMP      #54, R4          ::IS IT COMMA?
1582 013372 022704 000054          BNE      3$          ::BRANCH IF NOT
1583 013376 001007          MOV      DIGCNT, (R2)+          ::DIGIT COUNTER ON THE STACK
1584 013400 013722 024126          CLR      DIGCNT          ::TO USE AGAIN
1585 013404 005037 024126          INC      COMMA          ::KEEP TRACK OF COMMAS
1586 013410 005237 024124          BR      2$
1587 013414 000760          3$:  CMP      #25, R4          ::IS CONTROL U?
1588 013416 022704 000025          BEQ      NEWTIM          ::GO BACK
1589 013422 001645          CMP      #15, R4          ::IS IT CR?
1590 013424 022704 000015          BNE      4$          ::BRANCH IF NOT
1591 013430 001026          MOV      DIGCNT, (R2)+          ::GET DIGIT COUNT
1592 013432 013722 024126          CLR      DIGCNT
1593 013436 005037 024126          CMP      #DIGEND, R2          ::IS IT END OF DIGIT TABLE?
1594 013442 022702 024316          BNE      6$          ::ALL VALUES ARE NOT ENTERED SO BR
1595 013446 001060          CMP      #4, COMMA          ::
1596 013450 022737 000004 024124          BNE      6$          ::BRANCH IF NOT ENOUGH COMMAS
1597 013456 001054          MOV      #12, ECHOB          ::FOR LINE FEED
1598 013460 112737 000012 024244
  
```

```

1591 013466 104401 024244          TYPE      ECHOB
1592 013472 004737 014224          JSR      PC,   DECOCT
1593 013476 004737 013702          JSR      PC,   YERTN
1594                                     ::
1595 013502 000137 013700          JMP      8$
1596 013506 022704 000177          4$:     CMP      #177, R4      ::DECIMAL TO OCTAL CONVERSION ROUTINE
1597 013512 001005                                     ::ROUTINE TO ENTER VALUES INTO
1598 013514 104401 001202          BNE     9$      ::DEVICE REGISTER
1599 013520 104401 001203          TYPE    ,SQUES  ::EXIT
1600 013524 000714                                     ::IS IT RUBOUT?
1601 013526 020427 000060          9$:     CMP      R4,   #60     ::NO BR
1602 013532 002426                                     ::TYPE A '?'
1603 013534 020427 000071          BLT     6$      ::'CARRIAGE RETURN' AND 'LINE FEED'
1604 013540 003023          BGT     6$      ::READ NEW CHARACTERS
1605 013542 010422          MOV     R4,   (R2)+  ::IS NUMBER BELOW ZERO ?
1606 013544 005237 024126          INC     DIGCNT  ::BRANCH IF YES
1607 013550 005737 024124          TST     COMMA   ::IS NUMBER ABOVE NINE ?
1608 013554 001005          BNE     5$      ::BRANCH IF YES
1609 013556 023727 024126 000004    CMP     DIGCNT,#4  ::VALID NUMBER ON DIG STACK
1610 013564 003674          BLE     2$      ::COUNT NUMBER OF DIGITS BEFORE COMMA
1611 013566 000410          BR     6$      ::IS IT FIRST NUMBER ?
1612 013570 022737 000004 024124 5$:     CMP     #4,   COMMA  ::BRANCH IF NOT
1613 013576 002404          BLT     6$      ::MAXIMUM DIGITS IN A YEAR
1614 013600 023727 024126 000002    CMP     DIGCNT,#2  ::GET MORE DIGITS IF NOT FOUR YET
1615 013606 003663          BLE     2$      ::ERROR IF MORE THAN FOUR
1616                                     ::MAX POSSIBLE COMMAS
1617                                     ::IF MORE BRANCH
1618 013610 020227 024254          6$:     CMP     R2,   #DIGIT  ::MAX DIG FOR MONTH-DAY-HOUR-MIN
1619 013614 002402                                     ::GET MORE DIGITS
1620 013616 005042          CLR     -(R2)   ::TYPE ERROR MESSAGE
1621 013620 000773          BR     6$      ::CLEAR DIGIT STACK
1622 7$:
1623 (1) 013622 104401 013630          TYPE    ,71$    ::TYPE ASCIZ STRING
1624 (1) 013626 000422          BR     70$     ::GET OVER THE ASCIZ
1625 (1)                                     ::71$: .ASCIZ <200>/*****SYNTAX ERROR*****/
1626 (1) 013674                                     70$:
1627 013674 000137 013136          JMP     NEWTIM
1628 013700 000207          8$:     RTS     PC      ::EXIT FROM THE ROUTINE
1629
1630                                     .SBTTL THE TIME VALUE IN THE DEVICE REGISTER
1631
1632 YERTN: RESET
1633 MOV     YEAR, R0      ::YEAR VALUE
1634 MOV     R0,   R1      ::ALSO IN R1
1635 BIC     #177700,R0   ::CLEAR ALL BUT #77
1636 BIC     #170077,R1   ::CLEAR ALL BUT #7700
1637 ASL     R1          ::SHIFT LEFT
1638 ASL     R1          ::SHIFT LEFT
1639 BIS     R1,   R0      ::R0 CONTAINS YEAR VALUE IN DESIRE FORMAT
1640 BIS     #BIT14, @CSR  ::SET (SY) YEAR BIT
1641 MOV     R0,   @RGO   ::VALUE IN THE DEVICE REGISTER
1642 JSR     PC,   RDBYBIT ::CHECK FOR READY BIT SET
1643 (1) 013746 005737 024210          TST     FLAG.1   ::IS FLAG SET?
    
```


1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750

014224 012705 014416
014230 005037 024156
014234 005037 024160
014240 005037 024222
014244 005037 024174
014250 005037 024174
014254 005037 024122
014260 005037 024164
014264 014237 024174
014270 005012
014272 012537 024164
014276 014237 024160
014302 005012
014304 162737 000060 024160
014312 022737 000000 024160
014320 001406
014322 063737 024164 024156
014330 005337 024160
014334 001372
014336 063737 024156 024122
014344 005037 024156
014350 005337 024174
014354 001346
014356 013746 024122
014362 020227 024254
014366 001316
014370 012637 024132
014374 012637 024134
014400 012637 024136
014404 012637 024140
014410 012637 024142
014414 000207
014416 000001
014420 000012
014422 000144
014424 001750
014426

```
.SBTTL ROUTINE TO CONVERT DECIMAL TO OCTAL

:
:DECIMAL TO OCTAL CONVERSION ROUTINE
:STACK CONTAINS ASCII NUMBER
:POINTER TABLE CONTAINS WEIGH
:
:
DECOCT: MOV #POINTR,R5 ;:R5 CONTAINS STARTING ADDRESS OF THE WEIGH TABL
1$: CLR STR ;:CLEAR STR
CLR NUM ;:CLEAR NUM
CLR FLAG ;:CLEAR FLAG
CLR CNT ;:CLEAR CNT
CLR CNT ;:CLEAR CNT
CLR OCTNUM ;:CLEAR OCTAL NUM
CLR COUNT ;:CLEAR COUNT
MOV -(R2),CNT ;:#OF DIGITS IN NUM
CLR (R2) ;:CLEAR DIGIT STACK
2$: MOV (R5)+,COUNT ;:DECIMAL WEIGHT
MOV -(R2),NUM ;:DECIMAL NUM
CLR (R2) ;:CLEAR FROM DIG STACK
SUB #60, NUM ;:
CMP #0, NUM ;:IS IT ZERO
BEQ 4$
3$: ADD COUNT, STR ;:
DEC NUM ;:
BNE 3$
4$: ADD STR, OCTNUM ;:
CLR STR ;:
DEC CNT ;:
BNE 2$
MOV OCTNUM, -(SP) ;:
CMP R2, #DIGIT ;:
BNE DECOCT ;:
MOV (SP)+, YEAR ;:YEAR VALUE IN OCTAL
MOV (SP)+, MONTH ;:MONTH IN OCTAL
MOV (SP)+, DAY ;:DAY IN OCTAL
MOV (SP)+, HOUR ;:HOUR VALUE IN OCTAL
MOV (SP)+, MIN ;:MIN IN OCTAL
RTS PC ;:EXIT

POINTR: 1
12
144
1750

.SBTTL READ TIME OF THE CLOCK

:
:THIS ROUTINE READ CURRENT TIME OF THE CLOCK
:
READ:
```

```

(1) 014426 104401 014434 TYPE .65$ ::TYPE ASCIZ STRING
(1) 014432 000426 BR 64$ ::GET OVER THE ASCIZ
(1) ::65$: .ASCIZ <200>/ YR JD MO DA HR MI SE/
64$:
1751 014510 104401 001203 TYPE .$CRLF
1752 014514 012777 000004 007340 MOV #BIT2, @CSR ::READ YEAR
1753 014522 017700 007336 MOV @ARG0, R0 ::READ VALUE IN R0
1754 014526 010004 MOV R0, R4 ::ALSO IN R4
1755 014530 042700 177700 BIC #177700,R0 ::CLEAR ALL BUT #77
1756 014534 042704 160077 BIC #160077,R4 ::CLEAR ALL BUT #17700
1757 014540 006204 ASR R4 ::SHIFT RIGHT
1758 014542 006204 ASR R4 ::SHIFT RIGHT
1759 014544 050400 BIS R4, R0 ::IN OCTAL
1760 014546 010046 MOV R0, -(SP) ::NUMBER TO BE PRINTED ON STACK
1761 014550 104405 TYPDS
1762 014552 012777 000002 007302 MOV #BIT1, @CSR ::READ JULIAN DAY
1763 014560 017700 007300 MOV @ARG0, R0 ::READ VALUE IN R0
1764 014564 010004 MOV R0, R4 ::ALSO IN R4
1765 014566 042700 177700 BIC #177700,R0 ::CLEAR ALL BUT #77
1766 014572 042704 160077 BIC #160077,R4 ::CLEAR ALL BUT #17700
1767 014576 006204 ASR R4 ::SHIFT RIGHT
1768 014600 006204 ASR R4 ::SHIFT RIGHT
1769 014602 050400 BIS R4, R0 ::IN OCTAL
1770 014604 010046 MOV R0, -(SP) ::NUMBER ON STACK
1771 014606 104405 TYPDS
1772 014610 005077 007246 CLR @CSR ::CLEAR CSR
1773 014614 017700 007244 MOV @ARG0, R0 ::READ MONTH+DAY
1774 014620 010001 MOV R0, R1 ::ALSO IN R1
1775 014622 042700 170077 BIC #170077,R0 ::CLR ALL BUT 7700
1776 014626 006200 ASR R0 ::SHIFT
1777 014630 006200 ASR R0 ::SHIFT
1778 014632 006200 ASR R0 ::SHIFT
1779 014634 006200 ASR R0 ::SHIFT
1780 014636 006200 ASR R0 ::SHIFT
1781 014640 006200 ASR R0 ::SHIFT
1782 014642 006200 ASR R0 ::SHIFT
1783 014644 006200 ASR R0 ::SHIFT
1784 014646 010046 MOV R0, -(SP)
1785 014650 104405 TYPDS ::OCTAL NUM ON STACK
1786 014652 042701 177700 BIC #177700,R1 ::CLEAR ALL BUT #77
1787 014656 010146 MOV R1, -(SP) ::OCTAL NUM ON STACK
1788 014660 104405 TYPDS
1789 014662 005077 007174 CLR @CSR
1790 014666 017700 007174 MOV @ARG2, R0 ::HOUR VALUE
1791 014672 010001 MOV R0, R1 ::FOR MINUTE
1792 014674 042700 160377 BIC #160377,R0 ::CLEAR UNWANTED BITS
1793 014700 006200 ASR R0 ::SHIFT
1794 014702 006200 ASR R0 ::SHIFT
1795 014704 006200 ASR R0 ::SHIFT
1796 014706 006200 ASR R0 ::SHIFT
1797 014710 006200 ASR R0 ::SHIFT
1798 014712 006200 ASR R0 ::SHIFT
1799 014714 006200 ASR R0 ::SHIFT
1800 014716 006200 ASR R0 ::SHIFT
1801 014720 010046 MOV R0, -(SP) ::OCTAL NUM ON STACK
1802 014722 104405 TYPDS

```

```

1803 014724 042701 177700      BIC    #177700,R1      ;;
1804 014730 010100      MOV    R1, R0         ;;
1805 014732 010046      MOV    R0, -(SP)     ;;OCT ON STACK
1806 014734 104405      TYPDS
1807 014736 017700 007126      MOV    @RG4, R0      ;;READ SECOND
1808 014742 042700 017700      BIC    #17700, R0    ;;CLEAR ALL BUT #77
1809 014746 010046      MOV    R0, -(SP)     ;;OCTAL NUM ON STACK
1810 014750 104405      TYPDS
1811 014752 000207      RTS    PC
1812
1813
1814
1815
1816      ; THIS ROUTINE TESTS THE ACCURACY OF CLOCK.
1817      ; IF THERE IS AN ERROR, IT WILL PRINT ERROR
1818      ; MESSAGE
1819      ;
1820      A.CLK:
1821 014754 104401 014762      TYPE   ,65$          ;;TYPE ASCIZ STRING
(1) 014760 000411      BR     64$          ;;GET OVER THE ASCIZ
(1)      ;;65$: .ASCIZ <200>/L-CLK (L) N?/
(1) 015004      64$:
1822 015004 104411      1$: RDLIN          ;;IS THERE ANY CHAR
1823 015006 013604      MOV    @ (SP)+, R4   ;;GET CHARACTER
1824 015010 022704 000116      CMP    #116, R4     ;;IS IT 'N'?
1825 015014 001412      BEQ    3$           ;;BRANCH IF L-CLK NOT THERE
1826 015016 022704 000131      CMP    #131, R4     ;;IS IT 'Y'
1827 015022 001403      BEQ    2$           ;;BRANCH IF 'Y'
1828 015024 022704 000000      CMP    #0, R4       ;;IS IT 'CR'?
1829 015030 001365      BNE    1$           ;;NO:BR
1830 015032 004737 015360      2$: JSR    PC, L.CLK ;;COMPARE THE CLK WITH L.CLK
1831 015036 000137 015216      JMP    7$           ;;JUMP TO RETURN
1832
1833      3$:
(1) 015042 104401 015050      TYPE   ,67$          ;;TYPE ASCIZ STRING
(1) 015046 000412      BR     66$          ;;GET OVER THE ASCIZ
(1)      ;;67$: .ASCIZ <200>/P-CLOCK (L) N?/
(1) 015074      66$:
1834 015074 104411      4$: RDLIN          ;;GET CHARACTER IN R4
1835 015076 013604      MOV    @ (SP)+, R4   ;;IS IT 'N'?
1836 015100 022704 000116      CMP    #116, R4     ;;GO GET MANUALLY TIME BASE
1837 015104 001411      BEQ    6$           ;;IS IT 'Y'?
1838 015106 022704 000131      CMP    #131, R4     ;;YES:BR
1839 015112 001403      BEQ    5$           ;;IS IT 'CR'?
1840 015114 022704 000000      CMP    #0, R4       ;;NO:BR
1841 015120 001365      BNE    4$           ;;USE P.CLK AS TIME BASE
1842 015122 004737 015220      5$: JSR    PC, P.CLK ;;BRANCH TO RETURN
1843 015126 000433      BR
1844 015130      6$:
(1) 015130 104401 015136      TYPE   ,69$          ;;TYPE ASCIZ STRING
(1) 015134 000426      BR     68$          ;;GET OVER THE ASCIZ
(1)      ;;69$: .ASCIZ <200>/'TYPE TWO CHARACTERS AT 60 SECONDS APART'/
(1) 015212      68$:
1845 015212 004737 015554      JSR    PC, NO.CLK   ;;
1846 015216 000207      7$: RTS    PC       ;;RETURN
1847

```

1848
1849
1850
1851
1852
1853
1854
1855
1856

: THIS ROUTINE COMPARES KW11C CLOCK WITH
: KW11-P CLOCK FOR ACCURACY OF KW11-C CLOCK.
: IT PRINTS HARD ERROR IF CRYSTAL OSCILLATOR
: NOT ACCURATE.

```

1857 015220 010046 P.CLK: MOV R0, -(SP) ;; STORE R0
1858 015222 010246 MOV R2, -(SP) ;; R2
1859 015224 010446 MOV R4, -(SP) ;; R4
1860 015226 012704 015350 MOV #P.TAB, R4 ;; POINTER TO P.TABLE
1861 015232 012702 000002 MOV #2, R2 ;; COUNTER IN R2
1862 015236 012777 000010 006616 1$: MOV #10, @CSR ;; SET BIT READ TIME INTERVAL
1863 015244 012737 000144 172542 MOV #144, PKBUF ;; COUNTER EQUIV TO 100.
1864 015252 012777 047777 006610 MOV #47777, @RG4 ;; SET TIME INTERVAL COUNTER
1865 015260 012737 000003 172540 MOV #3, PKCSR ;; SET RATE SELECT AND RUN BIT
1866 015266 032737 000200 172540 2$: BIT #BIT7, PKCSR ;; IS DONE BIT SET?
1867 015274 001774 BEQ 2$, ;; NO: LOOP AROUND
1868 015276 017700 006566 MOV @RG4, R0 ;; READ DELTAT REGS
1869 015302 005302 DEC R2 ;;
1870 015304 001354 BNE 1$, ;; BRANCH IF NOT ZERO
1871 015306 010037 015346 MOV R0, P.CNT ;; GET DELAT T COUNTER
1872 015312 023724 015346 3$: CMP P.CNT, (R4)+ ;; COMPARE WITH EXPECTED
1873 015316 001405 BEQ 4$, ;;
1874 015320 020427 015356 CMP R4, #P.END ;; END OF TABLE?
1875 015324 003772 BLE 3$, ;; NO: BR
1876 015326 104033 ERROR+33 ;; HARD ERROR: CRYSTAL SEEMS TO BE
1877 015330 000402 BR 5$, ;;
1878 015332 104401 027353 4$: TYPE ,PASMSG ;; THE CLOCK PASS THE RELATIVE ACCURACY TEST
1879 015336 012604 5$: MOV (SP)+, R4 ;;
1880 015340 012602 MOV (SP)+, R2 ;;
1881 015342 012600 MOV (SP)+, R0 ;;
1882 015344 000207 RTS PC ;;

```

```

1883
1884 015346 000000 P.CNT: 00
1885 015350 007753 P.TAB: 7753
1886 015352 007752 7752
1887 015354 007754 7754
1888 015356 177777 P.END: 177777
1889

```

1890
1891
1892
1893
1894
1895

: THIS ROUTINE COMPARES KW11-C CLOCK
: WITH L-CLOCK FOR ACCURACY.
: IT PRINTS AN ERROR IF CRYSTAL OSCILLATOR
: IS NOT ACCURATE.

```

1896 015360 010046 L.CLK: MOV R0, -(SP) ;;
1897 015362 010246 MOV R2, -(SP) ;;
1898 015364 010446 MOV R4, -(SP) ;;
1899 015366 012704 015534 MOV #L.TAB, R4 ;; POINTER TO THE TABLE
1900 015372 012702 000002 MOV #2, R2 ;; COUNTER
1901 015376 012777 000010 006456 1$: MOV #10, @CSR ;; SET BIT TO READ DELTA ( INTVL
1902 015404 012777 047777 006456 MOV #47777, @RG4 ;; SET COUNTER IN DELTA T
1903 015412 005037 177546 CLR LKS ;; CLEAR STATUS OF L.CLK

```

```

1904 015416 032737 000200 177546 2$: BIT #BIT7, LKS ;;FLAG SET?
1905 015424 001774 BEQ 2$ ;;NO:BR
1906 015426 017700 006436 MOV @RG4, R0 ;;READ DELTA T COUNTER
1907 015432 005302 DEC R2 ;;DEC R2
1908 015434 001360 BNE 1$ ;;BR IF NOT ZERO
1909 015436 010037 015532 MOV R0, L_CNT ;;GET DELTA-T
1910 015442 022737 000062 024242 3$: CMP #62, HZ ;;WHAT FREQ?
1911 015450 001410 BEQ 4$ ;;BR IF 50HZ
1912 015452 023724 015532 CMP L_CNT, (R4)+ ;;COMPARE WITH TABLE FOR 60HZ
1913 015456 001417 BEQ 6$ ;;EXIT IF EQUAL
1914 015460 020427 015542 CMP R4, #L.END1 ;;IS IT END OF TAB1
1915 015464 003766 BLE 3$ ;;NO:BR
1916 015466 104033 ERROR+33 ;;CRYSTAL OSCILLATOR NOT ACCURATE
1917 015470 000414 BR 7$ ;;EXIT
1918 015472 012704 015544 4$: MOV #L.TAB2, R4 ;;PONTER TO THE TABLE
1919 015476 023724 015532 5$: CMP L_CNT, (R4)+ ;;
1920 015502 001405 BEQ 6$ ;;EXIT
1921 015504 020427 015552 CMP R4, #L.END2 ;;END OF TABLE?
1922 015510 003772 BLE 5$ ;;NO:BR
1923 015512 104033 ERROR+33 ;;OSCILLATOR NOT ACCURATE
1924 015514 000402 BR 7$ ;;EXIT
1925 015516 104401 027353 6$: TYPE ,PASMMSG ;;THE CLOCK PASSED THE RELATIVE ACCURACY TEST
1926 015522 012604 7$: MOV (SP)+, R4 ;;RESTORE
1927 015524 012602 MOV (SP)+, R2 ;;
1928 015526 012600 MOV (SP)+, R0 ;;
1929 015530 000207 RTS FC ;;
1930
1931 015532 000000 L_CNT: 00
1932 015534 007735 L_TAB1: 7735
1933 015536 007736 7736
1934 015540 007734 7734
1935 015542 177777 L_END1: 177777
1936 015544 007727 L_TAB2: 7727
1937 015546 007730 7730
1938 015550 007726 7726
1939 015552 177777 L_END2: 177777
1940
1941
1942 ; THIS ROUTINE IS USED TO GET SOME BASE
1943 ; FOR TIME CHARACTERS TYPED MUST BE
1944 ; VERY ACCURATE AS IT IS USED TO DETERMINE
1945 ; THE ACCURACY OF CLOCK
1946
1947 015554 005037 015722 NO.CLK: CLR N_CNT
1948 015560 105777 163360 1$: TSTB @STKS ;;TO STORE 2MIN INTVL CONSTANT
1949 015564 100375 BPL 1$ ;;IS THERE A CHAR?
1950 015566 117737 163354 024250 MOVB @STKB, ECHOB1 ;;NO:WAIT FOR A CHAR
1951 015574 005077 006262 CLR @CSR ;;
1952 015600 012777 047640 006262 MOV #47640, @RG4 ;;GET TIME CONST FOR 2 SEC
1953 015606 104401 024250 TYPE ,ECHOB1 ;;ECHO THE CHAR
1954 015612 032777 040000 006242 2$: BIT #BIT14, @CSR ;;IS BIT SET IN CSR?
1955 015620 001404 BEQ 3$ ;;NO:BR
1956 015622 005077 006234 CLR @CSR ;;CLEAR BIT SET
1957 015626 005237 015722 INC N_CNT ;;SO MANY TIMES 2 SEC INTERVAL
1958 015632 105777 163306 3$: TSTB @STKS ;;ANOTHER CHAR?
1959 015636 100365 BPL 2$ ;;NO:BR

```

```

1960 015640 017737 006224 015724      MOV    @RG4, N.CON      ;;READ TIME INT REG
1961 015646 017737 163274 024250      MOV    @STKB, ECHOB1   ;;GET CHAR
1962 015654 104401 024250              TYPE   ,ECHOB1
1963 015660 012704 015712              MOV    #TAB, R4        ;;ADDRESS OF TABLE
1964 015664 023724 015722      4$:   CMP    N.CNT, (R4)+  ;;COMP WITH NUM IN TAB
1965 015670 001405                          BEQ    5$
1966 015672 020427 015720              CMP    R4, #TAB.EN
1967 015676 003772                          BLE    4$
1968 015700 104033              ERROR+33
1969 015702 000402              BR     6$
1970 015704 104401 027353      5$:   TYPE   ,PASMSG    ;;THE CLOCK PASSED THE RELATIVE ACCURACY TEST
1971 015710 000207      6$:   RTS    PC
1972
1973 015712 000036      TAB:   36
1974 015714 000037              37
1975 015716 000035              35
1976 015720 177777      TAB.EN: 177777
1977 015722 000000      N.CNT: 00
1978 015724 000000      N.CON: 00
1979
1980
1981
1982      ;THIS IS TABLE DRIVEN ROUTINE,IT TESTS INVALID VALUE OF MONTH+DAY
1983      ;
1984      ;
1985      ;
1986      ;
1987
1988
1989
1990      ;
1991
1992      ;THIS ROUTINE WAITS FOR AN INTERRUPT
1993      ;IT ALSO PRINTS AN ERROR MESSAGE IF NOT INTERRUPTED
1994
1995 015726 010246      INTRTN: MOV    R2, -(SP)
1996 015730 005037 016024      CLR    CONT          ;;CLEAR LOCATION
1997 015734 005037 016026      CLR    CONT1         ;;CLEAR LOCATION
1998
1999 015740 013702 024054      1$:   MOV    CONST,R2   ;;TIME CONSTANT
2000 015744 005737 024212      TST   FLAG.2         ;;IS INTERRUPT OCCURED?
2001 015750 001023      BNE   4$              ;;YES,EXIT
2002
2003 015752 005302      2$:   DEC    R2          ;;DECREMENT COUNT
2004 015754 001376      BNE   2$              ;;BRANCH IF NOT ZERO
2005 015756 005237 016024      INC    CONT
2006 015762 023727 016024 003720      CMP    CONT,#3720
2007 015770 001363      BNE   1$              ;;2000 DECIMAL
2008 015772 005037 016024      CLR    CONT          ;;BRANCH IF NOT EQUAL
2009 015776 005237 016026      INC    CONT1         ;;ANOTHER COUNT

```

2011 016002 023737 016026 016030
 2012 016010 001353
 2013
 2014
 2015 016012 017737 006044 024120
 2016 016020 012602
 2017 016022 000207
 2018
 2019 016024 000000
 2020 016026 000000
 2021 016030 000000
 2022
 2023

CMP CONT1, TIMES
 BNE 1\$
 4\$: MOV @CSR, ACSR
 MOV (SP)+, R2
 RTS PC
 CONT: 0
 CONT1: 0
 TIMES: 00

:: DESIRE TIME EXPIRED?
 :: INTERRUPT FAIL TO OCCUR
 :: IF FLAG.2=0
 :: STATUS OF CSR
 :: RECOVER R2


```
2025 ;
2026 ; THIS IS WAIT ROUTINE
2027 ;
2028 016032 010246 WAIT: MOV R2, -(SP)
2029 016034 005037 016102 CLR CONT2
2030 016040 013702 024054 1$: MOV CONST, R2 ;:GET CONSTANT FOR 0.5MSECOND
2031 ;
2032 016044 005302 2$: DEC R2 ;:DECREMENT CONST
2033 016046 001376 BNE 2$ ;:LOOP BACK TILL ZERO
2034 016050 005237 016102 INC CONT2
2035 016054 023727 016102 003720 CMP CONT2, #3720 ;:EQUIVALENT TO 2000
2036 016062 001366 BNE 1$ ;:GO BACK IF NOT 1 SEC
2037 016064 005037 016102 CLR CONT2
2038 016070 005337 016104 DEC CONT3
2039 016074 001361 BNE 1$ ;:LOOP BACK IF TIME QUANTOM NOT EXPIRED
2040 016076 012602 MOV (SP)+, R2 ;:RECOVER R2
2041 016100 000207 RTS PC ;:EXIT FROM SUB ROUTINE
2042 016102 000000 CONT2: 0
2043 016104 000000 CONT3: 0
2044 ;
2045 ;
2046 ;
2047 016106 105777 163032 CNTLC: TSTB @STKS ;:IS THERE A CHARACTER?
2048 016112 100016 BPL 2$ ;:NO: DON'T WAIT
2049 016114 117746 163026 MOV B @STKB, -(SP) ;:STORE CHAR ON STACK
2050 016120 042716 177600 BIC #^C177, (SP) ;:STRIP OFF THE ASCII
2051 016124 022716 000003 CMP #3, (SP) ;:IS IT CONTROL C?
2052 016130 001003 BNE 1$ ;:NO: EXIT
2053 016132 005726 TST (SP)+ ;:
2054 016134 000137 003530 JMP RESTRT ;:
2055 016140 022726 000007 1$: CMP #7, (SP)+ ;:IS IT CONTROL G?
2056 016144 001001 BNE 2$ ;:NO:BR
2057 016146 104406 GTSWR
2058 016150 000207 2$: RTS PC
2059 ;
2060 ;
2061 ;
2062 016152 010246 DFLAY: MOV R2, -(SP) ;:SAVE R2
2063 016154 010346 MOV R3, -(SP) ;:SAVE R3
2064 016156 012703 000004 MOV #4, R3 ;:
2065 016162 013702 024054 1$: MOV CONST, R2 ;:GET CONSTANT
2066 016166 005302 2$: DEC R2 ;:COUNT DOWN
2067 016170 001376 BNE 2$ ;:BRANCH IF NOT ZERO YET
2068 016172 005303 DEC R3 ;:
2069 016174 001372 BNE 1$ ;:
2070 016176 012603 MOV (SP)+, R3 ;:RESTORE REGISTER
2071 016200 012602 MOV (SP)+, R2 ;:RESTORE R2
2072 016202 000207 RTS PC ;:RETURN
2073 ; THIS ROUTINE WAITS FOR A SECOND
2074 ;
2075 016204 013777 016224 005656 WAIT2: MOV TCNT, @RG4 ;:SET COUNTER TO WAIT
2076 016212 032777 040000 005642 1$: BIT #BIT14, @CSR ;:IS BIT14 SET?
2077 016220 001774 BEQ 1$ ;:NO:BR
2078 016222 000207 RTS PC ;:DESIRED TIME EXPIRED RETURN
2079 016224 000000 TCNT: 0000
2080 ;
```

```

2081 ;THIS ROUTINE WAITS FOR HOLD BIT TO RESET IN TIME(3 SECOND)
2082 ;THIS ROUTINE GETS CONTROL FROM MAIN ROUTINE AND EXITS
2083 ;IN MAIN ROUTINE IN ANY CASE
2084 ;
2085 016226 010246 WAIT1: MOV R2, -(SP)
2086 016230 005037 016320 CLR CONT4
2087 016234 005037 016322 CLR CONT5
2088 016240 032777 000001 005614 1$: BIT #BIT0, @CSR ;:IS (HOLD BIT) RESET?
2089 016246 001422 BEQ 3$ ;:RETURN IF SET
2090 016250 013702 024054 MOV CONST, R2 ;:CONSTANT FOR 0.5SEC
2091 016254 005302 2$: DEC R2 ;:DECREMENT CONST UNTIL 0.5 SECOND
2092 016256 001376 BNE 2$
2093 016260 005237 016320 INC CONT4 ;:GET COUNTER FOR NUMBR OF 0.5 SEC
2094 016264 023727 016320 003720 CMP CONT4, #3720 ;:EQUIVALENT TO 1 SECOND
2095 016272 001362 BNE 1$ ;:BRANCH IF SET
2096 016274 005037 016320 CLR CONT4 ;:INITIALIZE COUNTER
2097 016300 005237 016322 INC CONT5 ;:COUNT NO OF SEC
2098 016304 023727 016322 000005 CMP CONT5, #5 ;:IS IT 5 SECOND YET?
2099 016312 001352 BNE 1$ ;:BRANCH IF NOT 3 SEC YET?
2100 016314 012602 3$: MOV (SP)+, R2 ;:RECOVER R2
2101 016316 000207 RTS PC ;:
2102
2103 016320 000000 CONT4: 0
2104 016322 000000 CONT5: 0
2105
2106 ;
2107 ;THIS ROUTINE GETS CONTROL WHEN READY BIT FAILS TO SET.
2108 ;IN THIS ROUTINE VALID VALUES OF MONTH+DAY AND
2109 ;HOUR+MINUTE IS SET WHICH SUPPOSE TO SET READY BIT
2110 ; IN CSR.
2111 ;IF READY BIT FAILS TO SET PROGRAM HALTS.
2112 ;IF READY BIT SETS PROGRAM GOES TO CALLER.
2113 ;
2114 016324 032777 020000 005530 GETRDY: BIT #BIT13, @CSR ;:IS 'M' BIT SET?
2115 016332 001411 BEQ GET1 ;:BRANCH IF CLEAR
2116 016334 012777 000000 005522 MOV #0, @RGO ;:SET READY BIT
2117 016342 004737 017000 JSR PC, RDYBIT ;:CHECK FOR READY BIT SET
2118 (1) 016346 005737 024210 TST FLAG.1 ;:IS FLAG SET?
2119 016352 000461 BNE GET1 ;:TRY OTHER WAY TO SET RDYBIT
2120 016356 012777 140000 005476 GET1: MOV #BIT15+BIT14, @CSR ;:SET BIT15 AND BIT14 TO GET RDYBIT SET
2121 016364 012777 017073 005472 MOV #17073, @RGO ;:SET YEAR
2122 016372 004737 017000 JSR PC, RDYBIT ;:CHECK FOR READY BIT SET
2123 (1) 016376 005737 024210 TST FLAG.1 ;:IS FLAG SET?
2124 016402 001446 BEQ GET4 ;:EXIT IF SET
2125 016404 012777 040000 005450 GET2: MOV #BIT14, @CSR ;:SET VALID YEAR
2126 016412 012777 017073 005444 MOV #17073, @RGO ;:SET YEAR
2127 (1) 016420 004737 017000 JSR PC, RDYBIT ;:CHECK FOR READY BIT SET
2128 (1) 016424 005737 024210 TST FLAG.1 ;:IS FLAG SET?
2129 016430 001026 BNE GET3 ;:BRANCH IF READY BIT NOT SET
2130 016432 012777 100000 005422 MOV #BIT15, @CSR ;:SET (ST) TIME
2131 016440 012777 000401 005416 MOV #401, @RGO ;:SET 1-JANUARY IN RGO
2132 (1) 016446 004737 017000 JSR PC, RDYBIT ;:CHECK FOR READY BIT SET
2133 (1) 016452 005737 024210 TST FLAG.1 ;:IS FLAG SET?
2134 016456 001013 BNE GET3 ;:BRANCH TO HALT IF RDY DIDN'T SET
2135 016460 012777 100000 005374 MOV #BIT15, @CSR ;:SET TIME (BIT15) IN CSR TO SET

```

```

2133                                     ;; HOUR:MINUTE
2134 016466 012777 000000 005372      MOV  #0, @ARG?      ;; 0 HOUR+0 MINUTE
2135 016474 004737 017000              JSR  PC, RDYBIT    ;; CHECK FOR READY BIT SET
(1) 016500 005737 024210              TST  FLAG.1       ;; IS FLAG SET?
2136 016504 001405                    BEQ  GET4          ;; BRANCH TO RETURN
2137 016506 017737 005350 001122  GET3: MOV  @CSR, BADA  ;; GET STATUS OF CSR
2138 016514 104002                    ERROR+2          ;; HARD ERROR: READY BIT FAIL TO SET
2139 016516 000000                    HALT
2140 016520 000207                    GET4: RTS  PC      ;; RETURN TO CALLER
2141
2142                                     ;
2143                                     ;
2144 016522 004737 016152              GETTIM: JSR  PC, DELAY  ;; WAIT FOR A WHILE
2145 016526 010437 016554              MOV  R4, PRESEC   ;; PREVIOUS SECOND
2146 016532 017737 005332 01655C      MOV  @ARG4, SECOND ;; READ CURRENT VALUE OF SECOND
2147 016540 017737 005316 001122      MOV  @CSR, BADA   ;; READ STATUS OF CSR
2148 016546 000207                    RTS  PC
2149
2150 016550 000000                    SECOND: 00
2151 016552 000000                    DELTAT: 00
2152 016554 000000                    PRESEC: 00
2153 016556 000000                    PREDLT: 00
2154
2155                                     ;
2156                                     ; THIS ROUTINE WILL CAUSE 'M' BIT TO SET
2157                                     ; IT WILL PRINT ERROR IF 'M' BIT OR/AND (DE)
2158                                     ; BIT DIDN'T SET IN CSR WHILE SETTING PRESET
2159                                     ; VALUE
2160
2161 016560 005077 005276              INIPRE: CLR  @CSR      ;; CLEAR CSR
2162 016564 012777 000000 005272      MOV  #0, @ARG0    ;; PRESET MONTH+DAY
2163 016572 012737 077777 024052      MOV  #77777, COUNTR ;; DELAY LOOP SETUP
(1) 016600 004737 017206              JSR  PC, DMT      ;; DEAD MAN TIMER ROUTINE
(1) 016604 032777 100000 005250      BIT  #BIT15, @CSR ;; IS DE BIT SET
(1) 016612 001772                    BEQ  -12          ;; TRY AGAIN
2164 016614 005737 024210              TST  FLAG.1       ;; IS (DE) BIT SET IN CSR?
2165 016620 001056                    BNE  1$          ;; BRANCH IF NOT SET
2166 016622 032777 020000 005232      BIT  #BIT13, @CSR ;; IS 'M' BIT SET IN CSR?
2167 016630 001057                    BNE  2$          ;; BRANCH IF SET
2168 016632 005077 005224              CLR  @CSR         ;; CLEAR CSR
2169 016636 012777 105225 005222      MOV  #105225, @ARG2 ;; PRESET HOUR+MIN
2170 016644 012737 077777 024052      MOV  #77777, COUNTR ;; DELAY LOOP SETUP
(1) 016652 004737 017206              JSR  PC, DMT      ;; DEAD MAN TIMER ROUTINE
(1) 016656 032777 100000 005176      BIT  #BIT15, @CSR ;; IS DE BIT SET
(1) 016664 001772                    BEQ  -12          ;; TRY AGAIN
2171 016666 005737 024210              TST  FLAG.1       ;; IS (DE) BIT SET?
2172 016672 001031                    BNE  1$          ;; BRANCH IF (DE) BIT NOT SET
2173 016674 032777 020000 005160      BIT  #BIT13, @CSR ;; IS 'M' BIT SET?
2174 016702 001032                    BNE  2$          ;; EXIT IF SET
2175 016704 005077 005152              CLR  @CSR         ;; CLEAR CSR
2176 016710 012777 000000 005152      MOV  #0, @ARG4    ;; PRESET SECOND
2177 016716 012737 077777 024052      MOV  #77777, COUNTR ;; DELAY LOOP SETUP
(1) 016724 004737 017206              JSR  PC, DMT      ;; DEAD MAN TIMER ROUTINE
(1) 016730 032777 100000 005124      BIT  #BIT15, @CSR ;; IS DE BIT SET
(1) 016736 001772                    BEQ  -12          ;; TRY AGAIN
2178 016740 005737 024210              TST  FLAG.1       ;; IS (DE) SET?

```

READ TIME OF THE CLOCK

```
2179 016744 001004      BNE      1$          ::BRANCH IF NOT
2180 016746 032777 020000 005106      BIT      #BIT13, @CSR  ::IS 'M' BIT SET?
2181 016754 001005      BNE      2$          ::BRANCH IF SET
2182 016756 017737 005100 001122 1$:      MOV      @CSR, BADA    ::READ STATUS OF CSR
2183 016764 104030      ERROR+30          ::'M'BIT OR (DE) BIT FAIL TO SET
2184 016766 000000      HALT
2185 016770 012777 000000 005064 2$:      MOV      #0, @CSR     ::CLEAR CSR
2186 016776 000207      RTS      PC         ::EXIT
2187
2188
2189
2190
2191 017000 005037 024210      RDYBIT: CLR      FLAG.1
2192 017004 012737 077777 024052      MOV      #77777, COUNTR  ::SET UP LOOP COUNTER
2193 017012 032777 000200 005047 1$:      BIT      #BIT7, @CSR    ::IS READY BIT SET IN CSR?
2194 017020 001006      BNE      2$          ::BRANCH IF SET
2195 017022 005337 024052      DEC      COUNTR       ::DECREMENT LOOP COUNTER
2196 017026 001371      BNE      1$          ::TRY AGAIN
2197 017030 012737 000001 024210      MOV      #1, FLAG.1    ::SET FLAG :READY BIT NOT SET
2198 017036 000207      RTS      PC         ::EXIT
2199
2200
2201
2202 017040 010446      RDYCK1: MOV      R4, -(SP)  ::STORE R4
2203 017042 012704 000005      MOV      #5, R4
2204 017046 005037 024210      CLR      FLAG.1
2205 017052 012737 077777 024052 3$:      MOV      #77777, COUNTR
2206 017060 017746 004776 1$:      MOV      @CSR, -(SP)
2207 017064 042716 057577      BIC      #57577, (SP)
2208 017070 022726 120200      CMP      #BIT15+BIT13+BIT7, (SP)+
2209 017074 001407      BEQ      2$
2210 017076 005337 024052      DEC      COUNTR
2211 017102 001366      BNE      1$
2212 017104 005304      DEC      R4
2213 017106 001361      BNE      3$          ::WAIT MORE
2214 017110 005237 024210      INC      FLAG.1
2215 017114 012604 2$:      MOV      (SP)+, R4
2216 017116 000207      RTS      PC
2217
2218 ;THIS ROUTINE WAITS FOR READY BIT TO SET
2219 ;IF READY BIT DIDN'T SET, FLAG.1 IS SET TO
2220 ;INDICATE READY BIT FAIL TO SET.
2221 017120 010446      RDYCK: MOV      R4, -(SP)  ::SAVE R4
2222 017122 010346      MOV      R3, -(SP)    ::SAVE R3
2223 017124 012703 000005      MOV      #5, R3
2224 017130 005037 024210      CLR      FLAG.1
2225 017134 012737 077777 024052 1$:      MOV      #77777, COUNTR
2226 017142 017704 004714 2$:      MOV      @CSR, R4     ::READ CSR
2227 017146 042704 077577      BIC      #77577, R4
2228 017152 022704 100200      CMP      #BIT15+BIT7, R4
2229 017156 001410      BEQ      3$
2230 017160 005337 024052      DEC      COUNTR
2231 017164 001366      BNE      2$
2232 017166 005303      DEC      R3
2233 017170 001361      BNE      1$
2234 017172 012737 000001 024210      MOV      #1, FLAG.1
```

READ TIME OF THE CLOCK

SEQ 0061

```

2235 017200 012603          3$:  MOV    (SP)+, R3      ::RESTORE R3
2236 017202 012604          MOV    (SP)+, R4      ::RESTORE R4
2237 017204 000207          RTS    PC
2238
2239
2240 017206 005037 024210   DMT:  CLR    FLAG.1
2241 017212 005337 024052   DEC    COUNTR
2242 017216 003005          BGT    1$
2243 017220 012737 000001 024210   MOV    #1,    FLAG.1
2244 017226 062716 000010   ADD    #10,   (SP)
2245 017232 000207          1$:  RTS    PC
2246
2247
2248
2249
2250
2251 017234 010046   DEVCOD: MOV    R0,    -(SP)      ::SAVE R0
2252 017236 010246   MOV    R2,    -(SP)      ::SAVE R2
2253 017240 010446   MOV    R4,    -(SP)      ::SAVE R4
2254 017242 010546   MOV    R5,    -(SP)      ::SAVE R5
2255 017244 104401   TYPE
2256 017246 027306   DCMSG7      ::'ENTER VALUES AFTER(=) IS TYPED''
2257 017250 104401   TYPE
2258 017252 027041   DCMSG2
2259 017254 013746 024064   MOV    RGO,   -(SP)      ::''    DEVICE ADDRESS(    )=''
2260 017260 104402   TYPOC      ::STARTING ADDRESS OF DEVICE
2261 017262 104401   TYPE
2262 017264 027350   DCMSGX
2263 017266 104412   RDOCT      ::TYPE')=''
2264 017270 012604   MOV    (SP)+, R4      ::GET OCTAL NUMBER
2265 017272 022704 000000   CMP    #0,    R4      ::GET NUMBER FROM TOP OF THE STACK
2266 017276 001412   BEQ    DC5          ::IS IT 'CR'?
2267 017300 042704 000007   BIC    #7,    R4      ::YES:BR
2268 017304 012705 024062   MOV    #CSR,  R5      ::CLEAR LAST THREE BITS
2269 017310 042715 177770   DC4:  BIC    #177770,(R5)  ::GET ADDRESS
2270 017314 050425   BIS    R4,    (R5)+   ::CLEAR UNWANTED BITS
2271 017316 020527 024072   CMP    R5,    #PCV0
2272 017322 001372   BNE    DC4
2273 017324 104401   DC5:  TYPE
2274 017326 027067   DCMSG3
2275 017330 013746 024072   MOV    PCV0,  -(SP)    ::TYPE PC INTRPT VECTOR( )=
2276 017334 104402   TYPOC
2277 017336 104401   TYPE
2278 017340 027350   DCMSGX
2279 017342 104412   RDOCT
2280 017344 012604   MOV    (SP)+, R4      ::READ VECTOR VALUE
2281 017346 022704 000000   CMP    #0,    R4      ::IS IT 'CR'?
2282 017352 001454   BEQ    DC6          ::YES:BR
2283 017354 013702 024072   MOV    PCV0,  R2      ::GET PREVIOUS ADDRESS
2284 017360 062702 000002   ADD    #2,    R2      ::NEXT ADDRESS
2285 017364 010277 004502   MOV    R2,    @PCV0
2286 017370 010237 024074   MOV    R2,    PSV0
2287 017374 012777 000004 004472   MOV    #4,    @PSV0
2288 017402 062702 000004   ADD    #4,    R2
2289 017406 010277 004464   MOV    R2,    @PCV1
2290 017412 010237 024100   MOV    R2,    PSV1

```

```
2291 017416 012777 000004 004454      MOV      #4,      @PSV1      ::  
2292 017424 010437 024072      MOV      R4,      PCV0      ::NEW ADDRESS  
2293 017430 012777 017554 004434      MOV      #INTRPT,@PCV0      ::ADDRESS OF INTERRUPT ROUTINE  
2294 017436 062704 000002      ADD      #2,      R4      ::INCREMENT ADDRESS BY 2  
2295 017442 012777 000340 004424      MOV      #PR7,   @PSV0      ::  
2296 017450 062704 000002      ADD      #2,      R4      ::NEXT ADDRESS  
2297 017454 010437 024076      MOV      R4,      PCV1      ::  
2298 017460 012777 017662 004410      MOV      #TIMRTN,@PCV1      ::  
2299 017466 062704 000002      ADD      #2,      R4      ::  
2300 017472 010437 024100      MOV      R4,      PSV1      ::  
2301 017476 012777 000340 004374      MOV      #PR7,   @PSV1      ::  
2302  
2303 017504 104401      DC6:    TYPE  
2304 017506 027171      DCMMSG5  
2305 017510 013704 024102      MOV      BRLV,   R4      ::'BUS REQUEST LEVEL( )='  
2306 017514 010446      MOV      R4,      -(SP)      ::  
2307 017516 104402      TYPOC  
2308 017520 104401      TYPE  
2309 017522 027350      DCMMSGX  
2310 017524 104412      RDOCT  
2311 017526 012604      MOV      (SP)+,  R4      ::  
2312 017530 022704 000000      CMP      #0,      R4      ::IS IT 'CR'?  
2313 017534 002002      BGE      DC7  
2314 017536 010437 024102      MOV      R4,      BRLV      ::  
2315 017542 012605      DC7:    MOV      (SP)+,  R5      ::RESTORE REGISTERS  
2316 017544 012604      MOV      (SP)+,  R4      ::  
2317 017546 012602      MOV      (SP)+,  R2      ::  
2318 017550 012600      MOV      (SP)+,  R0      ::  
2319 017552 000207      RTS      PC  
2320      ;DATE + TIME INTERRUPT ROUTINE  
2321      ;  
2322 017554 011637 024216      INTRPT: MOV      (SP),   TEMP1      ::RETURN ADDRESS  
2323 017560 017737 004276 024120      MOV      @CSR,   ACSR      ::STORE CONTENT OF CSR  
2324 017566 012777 000000 004266      MOV      #0,     @CSR      ::CLEAR CSR (DISABLE INTERRUPT)  
2325 017574 013737 024120 001122      MOV      ACSR,   BADA      ::  
2326 017602 005037 024212      5$:    CLR      FLAG.2      ::CLR FLAG  
2327 017606 032737 000100 024120      1$:    BIT      #100,   ACSR      ::WAS INTRPT ENABLED?  
2328 017614 001001      BNE      2$      ::BRANCH IF YES  
2329 017616 104020      ERROR+20      ::UNEXPECTED INTRPT AS (DIE) NOT SET  
2330 017620 032737 100000 024120      2$:    BIT      #100000,ACSR      ::WAS (DE) BIT SET?  
2331 017626 001001      BNE      3$      ::BRANCH IF YES  
2332 017630 104021      ERROR+21      ::UNEXP INTRPT AS (DE) BIT NOT SET  
2333 017632 022737 000001 024226      3$:    CMP      #1,     INTREG      ::IS IT UPDATE INT?  
2334 017640 001405      BEQ      4$      ::FRANCH IF YES  
2335 017642 022737 000002 024226      CMP      #2,     INTREG      ::  
2336 017650 001401      BEQ      4$      ::BRANCH IF YES  
2337 017652 104022      ERROR+22      ::UNEXP UNEXPLAINABLE INTRPT  
2338 017654 005237 024212      4$:    INC      FLAG.2  
2339 017660 000002      RTI      ::RETURN  
2340  
2341      ;TIME INTERVAL INTERRUPT ROUTINE  
2342 017662 011637 024220      TIMRTN: MOV      (SP),   TEMP2      ::  
2343 017666 005037 024212      CLR      FLAG.2  
2344 017672 017737 004172 001126      MOV      @RG4,   BAD      ::READ TIME INTERVAL  
2345 017700 017737 004156 024120      MOV      @CSR,   ACSR  
2346 017706 005077 004150      CLR      @CSR
```

```

2347 017712 042777 040000 004150      BIC  #40000, @RG4      ;;CLEAR INTRPT ENABLE FLAG
2348 017720 005037 024212              CLR  FLAG.2
2349 017724 032737 000040 024120 1$:  BIT  #40,  ACSR      ;;TIME INTRPT WAS ENABLED?
2350 017732 001001              BNE  2$
2351 017734 104020              ERROR+20              ;;UNEXPECTED INTERRUPT
2352 017736 032737 040000 024120 2$:  BIT  #40000, ACSR      ;;(TI) BIT SET IN CSR
2353 017744 001001              BNE  3$              ;;BRANCH IF YES
2354 017746 104023              ERROR+23
2355 017750 022737 000005 024226 3$:  CMP  #5,  INTREG      ;;FLAG FOR TIMEINTERVAL INTRP
2356 017756 001401              BEO  4$              ;;BRANCH IF YES
2357 017760 104022              ERROR+22              ;;UNEXPECTED INTERRUPT
2358 017762 005237 024212          4$:  INC  FLAG.2          ;;FLAG TO INDICATE INTERRUPTED
2359 017766 000002              RTI
2360
2361      ;
2362      ; TRAP ROUTINE
2363      ;
2364 017770 012600      TRAP4:  MOV  (SP)+,  R0      ;;POP STACK INTO R0
2365 017772 104401 020000      TYPE  1$
2366 017776 000423              BR   2$
2367 020000 052600 042516 050130 1$:  .ASCIZ <200>/UNEXPECTED TRAP TO VECTOR 4 FROM ../
      020006 041505 042524 020104
      020014 051124 050101 052040
      020022 020117 042526 052103
      020030 051117 032040 043040
      020036 047522 020115 027056
      020044 000
2368      .EVEN
2369 020046 010046      2$:  MOV  R0,  -(SP)      ;;SAVE R0 FOR TYP0UT
2370 020050 104402      TYPOC      ;;GO TYPE OCTAL ASCII
2371 020052 000000      HALT
2372 020054 000137 001712      JMP  START
2373
2374
2375
2376      .SBTTL  END OF PASS ROUTINE
      (1)
      (2)
      (1)      ;*****
      (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 START
      (1)
      (1)
      (1)      $EOP:
      (1)      SCOPE
      (1)      CLR  $STNM      ;;ZERO THE TEST NUMBER
      (1)      CLR  $TIMES      ;;ZERO THE NUMBER OF ITERATIONS
      (1)      CLR  $PASS      ;;INCREMENT THE PASS NUMBER
      (1)      INC  $PASS      ;;DON'T ALLOW A NEG. NUMBER
      (1)      BIC  #100000, $PASS
      (1)      DEC  (PC)+      ;;LOOP?
      (1)      $EOPCT: .WORD 1
      (1)      BGT  $DOAGN      ;;YES
      (1)      MOV  (PC)+, @ (PC)+      ;;RESTORE COUNTER
      (1)      $ENDCT: .WORD 1
      (1)      $EOPCT
      (1)      TYPE  , $ENDMG      ;;TYPE 'END PASS #'
  
```

```

(2) 020124 013746 001214      MOV      $PASS,-(SP)      ;;SAVE $PASS FOR TYPEOUT
(2) 020130 104405              TYPDS                    ;;GO TYPE--DECIMAL ASCII WITH SIGN
(1) 020132 104401 020162      TYPE      , $NULL        ;;TYPE A NULL CHARACTER
(1) 020136 013700 000042      $GET42: MOV      @#42,RO   ;;GET MONITOR ADDRESS
(1) 020142 001405              BEQ      $DOAGN          ;;BRANCH IF NO MONITOR
(1) 020144 000005              RESET                    ;;CLEAR THE WORLD
(1) 020146 004710      $ENDAD: JSR      PC,(RO)  ;;GO TO MONITOR
(1) 020150 000240              NOP                      ;;SAVE ROOM
(1) 020152 000240              NOP                      ;;FOR
(1) 020154 000240              NOP                      ;;ACT11
(1) 020156              $DOAGN:              ;;
(1) 020156 000137              JMP      @ (PC)+          ;;RETURN
(1) 020160 001712      $RTNAD: .WORD      START
(1) 020162      377      377      000      $NULL: .BYTE      -1,-1,0  ;;NULL CHARACTER STRING
(1) 020165      015      042412 042116 $ENDMG: .ASCIZ    <15><12>/END PASS #/
(1) 020172 050040 051501 020123
(1) 020200 000043
    
```

2377
2378
2379
2380

.NLIST MC,MD,CND
.SBTTL TYPE ROUTINE

```

(1)
(2)
(1) *****
(1) *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1) *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1) *NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1) *NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER 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) 020202 105737 001157      $TYPE: TSTB      $TPFLG      ;;IS THERE A TERMINAL?
(1) 020206 100002              BPL      1$              ;;BR IF YES
(1) 020210 000000              HALT                    ;;HALT HERE IF NO TERMINAL
(1) 020212 000430              BR      3$              ;;LEAVE
(1) 020214 010046      1$:  MOV      RO,-(SP)      ;;SAVE RO
(1) 020216 017600 000002      MOV      @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
(1) 020222 122737 000001 001226      CMPB     #APTENV,$ENV    ;;RUNNING IN APT MODE
(1) 020230 001011      BNE     62$              ;;NO,GO CHECK FOR APT CONSOLE
(1) 020232 132737 000100 001227      BITB     #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1) 020240 001405      BEQ     62$              ;;NO,GO CHECK FOR CONSOLE
(1) 020242 010037 020252      MOV      RO,61$         ;;SETUP MESSAGE ADDRESS FOR APT
(1) 020246 004737 020472      JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
(1) 020252 000000      61$: .WORD      0          ;;MESSAGE ADDRESS
(1) 020254 132737 000040 001227      62$: BITB     #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 020262 001003      BNE     60$              ;;YES,SKIP TYPE OUT
(1) 020264 112046      2$:  MOVB     (RO)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 020266 001005      BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
(1) 020270 005726      TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
(1) 020272 012600      60$: MOV      (SP)+,RO      ;;RESTORE RO
    
```



```
(1) 020274 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
(1) 020300 000002 RTI ::RETURN
(1) 020302 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
(1) 020306 001430 BEQ 8$
(1) 020310 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT <CRLF>
(1) 020314 001006 BNE 5$
(1) 020316 005726 TST (SP)+ ::POP <CR><LF> EQUIV
(1) 020320 104401 TYPE ::TYPE A CR AND LF
(1) 020322 001203 $CRLF
(1) 020324 105037 020460 CLRB $CHARCNT ::CLEAR CHARACTER COUNT
(1) 020330 000755 BR 2$ ::GET NEXT CHARACTER
(1) 020332 004737 020414 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
(1) 020336 123726 001156 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
(1) 020342 001350 BNE 2$ ::IF NO GO GET NEXT CHAR.
(1) 020344 013746 001154 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
(1) ::AND THE NULL CHAR.
(1) 020350 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
(1) 020354 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
(1) 020356 004737 020414 JSR PC,$TYPEC ::GO TYPE A NULL
(1) 020362 105337 020460 DECB $CHARCNT ::DO NOT COUNT AS A COUNT
(1) 020366 000770 BR 7$ ::LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 020370 112716 000040 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
(1) 020374 004737 020414 9$: JSR PC,$TYPEC ::TYPE A SPACE
(1) 020400 132737 000007 020460 BITB #7,$CHARCNT ::BRANCH IF NOT AT
(1) 020406 001372 BNE 9$ ::TAB STOP
(1) 020410 005726 TST (SP)+ ::POP SPACE OFF STACK
(1) 020412 000724 BR 2$ ::GET NEXT CHARACTER
(1) 020414 105777 160530 $TYPEC: TSTB @STPS ::WAIT UNTIL PRINTER IS READY
(1) 020420 100375 BPL $TYPEC
(1) 020422 116677 000002 160522 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 020430 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
(1) 020436 001003 BNE 1$ ::BRANCH IF NO
(1) 020440 105037 020460 CLRB $CHARCNT ::YES--CLEAR CHARACTER COUNT
(1) 020444 000406 BR $TYPEX ::EXIT
(1) 020446 122766 000012 000002 1$: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
(1) 020454 001402 BEQ $TYPEX ::BRANCH IF YES
(1) 020456 105227 INCB (PC)+ ::COUNT THE CHARACTER
(1) 020460 000000 $CHARCNT: WORD 0 ::CHARACTER COUNT STORAGE
(1) 020462 000207 $TYPEX: RTS PC
(1)
(1) 2381 .SBTTL APT COMMUNICATIONS ROUTINE
(1)
(1) 020464 112737 000001 020730 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
(1) 020472 112737 000001 020726 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
(1) 020500 000403 BR $ATYC
(1) 020502 112737 000001 020730 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
(1) 020510 $ATYC:
(3) 020510 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
(3) 020512 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
(1) 020514 105737 020726 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
(1) 020520 001450 BEQ 5$ ::IF NOT: BR
(1) 020522 122737 000001 001226 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
```

```

(1) 020530 001031          BNE      3$          ;;IF NOT: BR
(1) 020532 132737 000100 001227 BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 020540 001425          BEQ     3$          ;;IF NOT: BR
(1) 020542 017600 000004          MOV     @4(SP),RO    ;;GET MESSAGE ADDR.
(1) 020546 062766 000002 000004 ADD     #2,4(SP)    ;;BUMP RETURN ADDR.
(1) 020554 005737 001206          1$: TST     $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
(1) 020560 001375          BNE     1$          ;;IF NOT: WAIT
(1) 020562 010037 001222          MOV     RO,$MSGAD   ;;PUT ADDR IN MAILBOX
(1) 020566 105720          2$: TSTB   (RO)+     ;;FIND END OF MESSAGE
(1) 020570 001376          BNE     2$          ;;
(1) 020572 163700 001222          SUB     $MSGAD,RO   ;;SUB START OF MESSAGE
(1) 020576 006200          ASR     RO          ;;GET MESSAGE LNTH IN WORDS
(1) 020600 010037 1224          MOV     RO,$SMSGLGT ;;PUT LENGTH IN MAILBOX
(1) 020604 012737 0004 001206 MOV     #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 020612 000413          BR      5$          ;;
(1) 020614 017637 000004 020640 3$: MOV     @4(SP),4$    ;;PUT MSG ADDR IN JSR LINKAGE
(1) 020622 062766 000002 000004 ADD     #2,4(SP)    ;;BUMP RETURN ADDRESS
(3) 020630 013746 177776          MOV     177776,-(SP) ;;PUSH 177776 ON STACK
(1) 020634 004737 020202          JSR    PC,$TYPE    ;;CALL TYPE MACRO
(1) 020640 000000          4$: .WORD 0
(1) 020642          5$:
(1) 020642 105737 020730          10$: TSTB   $FFLG     ;;SHOULD REPORT FATAL ERROR?
(1) 020646 001416          BEQ     12$        ;;IF NOT: BR
(1) 020650 005737 001226          TST    $ENV        ;;RUNNING UNDER APT?
(1) 020654 001413          BEQ     12$        ;;IF NOT: BR
(1) 020656 005737 001206          11$: TST     $MSGTYPE   ;;FINISHED LAST MESSAGE?
(1) 020662 001375          BNE     11$        ;;IF NOT: WAIT
(1) 020664 017637 000004 001210 MOV     @4(SP),$FATAL ;;GET ERROR #
(1) 020672 062766 000002 000004 ADD     #2,4(SP)    ;;BUMP RETURN ADDR.
(1) 020700 005237 001206          INC    $MSGTYPE    ;;TELL APT TO TAKE ERROR
(1) 020704 105037 020730          12$: CLRB   $FFLG     ;;CLEAR FATAL FLAG
(1) 020710 105037 020727          CLRB   $LFLG     ;;CLEAR LOG FLAG
(1) 020714 105037 020726          CLRB   $MFLG     ;;CLEAR MESSAGE FLAG
(3) 020720 012601          MOV    (SP)+,R1   ;;POP STACK INTO R1
(3) 020722 012600          MOV    (SP)+,RO   ;;POP STACK INTO RO
(1) 020724 000207          RTS     PC        ;;RETURN
(1) 020726 000          $MFLG: .BYTE 0    ;;MESSG. FLAG
(1) 020727 000          $LFLG: .BYTE 0    ;;LOG FLAG
(1) 020730 000          $FFLG: .BYTE 0    ;;FATAL FLAG
(1) 020732          .EVEN
(1) 000200          APTSIZE=200
(1) 000001          APTENV=001
(1) 000100          APTSPOOL=100
(1) 000040          APTCSUP=040
2382          .SBTTL TTY INPUT ROUTINE
(1)
(2)          ;*****
(1)          .ENABL LSB
(1)
(2)          ;*****
(1)          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1)          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1)          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1)          ;*WHEN OPERATING IN TTY FLAG MODE.
(1) 020732 022737 000176 001140 $CKSWR: CMP     #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
(1) 020740 001114          BNE     15$        ;;BRANCH IF NO

```

```

(1) 020742 105777 160176          TSTB  @STKS          ;;CHAR THERE?
(1) 020746 100111                   BPL   15$           ;;IF NO, DON'T WAIT AROUND
(1) 020750 117746 160172          MOVB  @STKB,-(SP)   ;;SAVE THE CHAR
(1) 020754 042716 177600          BIC   #'C177,(SP) ;;STRIP-OFF THE ASCII
(1) 020760 022726 000007          CMP   #7,(SP)+     ;;IS IT A CONTROL G?
(1) 020764 001102                   BNE   15$           ;;NO, RETURN TO USER
(1) 020766 123727 001134 000001   CMPB  $AUTOB,#1    ;;ARE WE RUNNING IN AUTO-MODE?
(1) 020774 001476                   BEQ   15$           ;;BRANCH IF YES
(1) 020776 104401 021544          TYPE  ,SCNTLG      ;;ECHO THE CONTROL-G (^G)
(1) 021002 104401 021551          SGT$WR: TYPE ,SMSWR ;;TYPE CURRENT CONTENTS
(2) 021006 013746 000176          MOV   SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
(2) 021012 104402                   TYPEOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 021014 104401 021562          TYPE  ,SMNEW       ;;PROMPT FOR NEW SWR
(1) 021020 005046          19$: CLR -(SP)     ;;CLEAR COUNTER
(1) 021022 005046          CLR -(SP)         ;;THE NEW SWR
(1) 021024 105777 160114          7$: TSTB @STKS    ;;CHAR THERE?
(1) 021030 100375                   BPL   7$           ;;IF NOT TRY AGAIN
(1) 021032 117746 160110          MOVB  @STKB,-(SP) ;;PICK UP CHAR
(1) 021036 042716 177600          BIC   #'C177,(SP) ;;MAKE IT 7-BIT ASCII
(1) 021042 021627 000003          CMP   (SP),#3     ;;IS IT A CONTROL-C?
(1) 021046 001015                   BNE   9$           ;;BRANCH IF NOT
(1) 021050 104401 021532          TYPE  ,SCNTLC     ;;YES, ECHO CONTROL-C (^C)
(1) 021054 062706 000006          ADD   #6,SP       ;;CLEAN UP STACK
(1) 021060 123727 001135 000001   CMPB  $INTAG,#1   ;;REENABLE TTY KEYBOARD INTERRUPTS?
(1) 021066 001003                   BNE   8$           ;;BRANCH IF NO
(1) 021070 012777 000100 160046   MOV   #100,@STKS ;;ALLOW TTY KEYBOARD INTERRUPTS
(1) 021076 000137 003530          8$: JMP  RESTRT   ;;CONTROL-C RESTART
(1) 021102 021627 000025          9$: CMP (SP),#25  ;;IS IT A CONTROL-U?
(1) 021106 001005                   BNE   10$          ;;BRANCH IF NOT
(1) 021110 104401 021537          TYPE  ,SCNTLU     ;;YES, ECHO CONTROL-U (^U)
(1) 021114 062706 000006          20$: ADD #6,SP    ;;IGNORE PREVIOUS INPUT
(1) 021120 000737                   BR   19$          ;;LET'S TRY IT AGAIN
(1) 021122 021627 000015          10$: CMP (SP),#15 ;;IS IT A <CR>?
(1) 021126 001022                   BNE   16$          ;;BRANCH IF NO
(1) 021130 005766 000004          TST   4(SP)       ;;YES, IS IT THE FIRST CHAR?
(1) 021134 001403                   BEQ   11$          ;;BRANCH IF YES
(1) 021136 016677 000002 157774   MOV   2(SP),@SWR  ;;SAVE NEW SWR
(1) 021144 062706 000006          11$: ADD #6,SP    ;;CLEAN UP STACK
(1) 021150 104401 001203          14$: TYPE ,SCRLF  ;;ECHO <CR> AND <LF>
(1) 021154 123727 001135 000001   CMPB  $INTAG,#1  ;;RE-ENABLE TTY KBD INTERRUPTS?
(1) 021162 001003                   BNE   15$          ;;BRANCH IF NOT
(1) 021164 012777 000100 157752   MOV   #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
(1) 021172 000002                   RTI   ;;RETURN
(1) 021174 004737 020414          16$: JSR PC,$TYPEC ;;ECHO CHAR
(1) 021200 021627 000060          CMP   (SP),#60   ;;CHAR < 0?
(1) 021204 002420                   BLT   18$          ;;BRANCH IF YES
(1) 021206 021627 000067          CMP   (SP),#67   ;;CHAR > 7?
(1) 021212 003015                   BGT   18$          ;;BRANCH IF YES
(1) 021214 042726 000060          BIC   #60,(SP)+  ;;STRIP-OFF ASCII

```

```
(1) 021220 005766 000002          TST      2(SP)          ;; IS THIS THE FIRST CHAR
(1) 021224 001403                   BEQ      17$           ;; BRANCH IF YES
(1) 021226 006316                   ASL      (SP)          ;; NO, SHIFT PRESENT
(1) 021230 006316                   ASL      (SP)          ;; CHAR OVER TO MAKE
(1) 021232 006316                   ASL      (SP)          ;; ROOM FOR NEW ONE.
(1) 021234 005266 000002          17$: INC      2(SP)          ;; KEEP COUNT OF CHAR
(1) 021240 056616 177776          BIS      -2(SP),(SP)   ;; SET IN NEW CHAR
(1) 021244 000667                   BR       7$           ;; GET THE NEXT ONE
(1) 021246 104401 001202          18$: TYPE  ,SQUES      ;; TYPE ?<CR><LF>
(1) 021252 000720                   BR       20$          ;; SIMULATE CONTROL-U
(1)                                .DSABL  LSB
(1)
(1)
(2)                                ;;*****
(1)                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                ;*CALL:
(1)                                ;* RDCHR                      ;; INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                ;* RETURN HERE                ;; CHARACTER IS ON THE STACK
(1)                                ;*                               ;; WITH PARITY BIT STRIPPED OFF
(1)                                ;*
(1)                                ;*
(1) 021254 011646                   $RDCHR: MOV     (SP),-(SP) ;; PUSH DOWN THE PC
(1) 021256 016666 000004 000002   MOV     4(SP),2(SP)    ;; SAVE THE PS
(1) 021264 105777 157654          1$: TSTB   @STKS        ;; WAIT FOR
(1) 021270 100375                   BPL     1$            ;; A CHARACTER
(1) 021272 117766 157650 000004   MOVB   @STKB,4(SP)    ;; READ THE TTY
(1) 021300 042766 177600 000004   BIC    #'C<177>,4(SP) ;; GET RID OF JUNK IF ANY
(1) 021306 026627 000004 000023   CMP    4(SP),#23     ;; IS IT A CONTROL-S?
(1) 021314 001013                   BNE     3$            ;; BRANCH IF NO
(1) 021316 105777 157622          2$: TSTB   @STKS        ;; WAIT FOR A CHARACTER
(1) 021322 100375                   BPL     2$            ;; LOOP UNTIL ITS THERE
(1) 021324 117746 157616          MOVB   @STKB,-(SP)    ;; GET CHARACTER
(1) 021330 042716 177600          BIC    #'C177,(SP)   ;; MAKE IT 7-BIT ASCII
(1) 021334 022627 000021          CMP    (SP)+,#21     ;; IS IT A CONTROL-Q?
(1) 021340 001366                   BNE     2$            ;; IF NOT DISCARD IT
(1) 021342 000750                   BR      1$            ;; YES, RESUME
(1) 021344 026627 000004 000140   3$: CMP    4(SP),#140  ;; IS IT UPPER CASE?
(1) 021352 002407                   BLT     4$            ;; BRANCH IF YES
(1) 021354 026627 000004 000175   CMP    4(SP),#175    ;; IS IT A SPECIAL CHAR?
(1) 021362 003003                   BGT     4$            ;; BRANCH IF YES
(1) 021364 042766 000040 000004   BIC    #40,4(SP)    ;; MAKE IT UPPER CASE
(1) 021372 000002                   4$: RTI                ;; GO BACK TO USER
(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) 021374 010346                   $RDLIN: MOV     R3,-(SP) ;; SAVE R3
(1) 021376 012703 021522          1$: MOV     #$TTYIN,R3 ;; GET ADDRESS
(1) 021402 022703 021532          2$: CMP     #$TTYIN+8.,R3 ;; BUFFER FULL?
(1) 021406 101415                   BLOS   4$            ;; BR IF YES
(1) 021410 104410                   RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
(1) 021412 112613                   MOVB   (SP)+(R3)     ;; GET CHARACTER
(1) 021414 122713 000003          CMPB   #3,(R3)       ;; IS IT A CONTROL-C?
```

```

(1) 021420 001005 BNE 10$ ::BRANCH IF NO
(1) 021422 104401 021532 TYPE ,SCNTLC ::TYPE A CONTROL-C (^C)
(1) 021426 012603 MOV (SP)+,R3 ::RESTORE R3
(1) 021430 000137 003530 JMP RESTR ::GOTO CONTROL-C RESTART
(1) 021434 122713 000177 10$: CMPB #177,(R3) ::IS IT A RUBOUT
(1) 021440 001003 BNE 3$ ::SKIP IF NOT
(1) 021442 104401 001202 4$: TYPE ,SQUES ::TYPE A '?'
(1) 021446 000753 BR 1$ ::CLEAR THE BUFFER AND LOOP
(1) 021450 111337 021520 3$: MOVB (R3),9$ ::ECHO THE CHARACTER
(1) 021454 104401 021520 TYPE ,9$
(1) 021460 122723 000015 CMPB #15,(R3)+ ::CHECK FOR RETURN
(1) 021464 001346 BNE 2$ ::LOOP IF NOT RETURN
(1) 021466 105063 177777 CLRB -1(R3) ::CLEAR RETURN (THE 15)
(1) 021472 104401 001204 TYPE ,SLF ::TYPE A LINE FEED
(1) 021476 012603 MOV (SP)+,R3 ::RESTORE R3
(1) 021500 011646 MOV (SP)-,(SP) ::ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 021502 016666 000004 000002 MOV 4(SP),2(SP) :: FIRST ASCII CHARACTER ON IT
(1) 021510 012766 021522 000004 MOV #STTYIN,4(SP)
(1) 021516 000002 RTI ::RETURN
(1) 021520 000 9$: .BYTE 0 ::STORAGE FOR ASCII CHAR. TO TYPE
(1) 021521 000 .BYTE 0 ::TERMINATOR
(1) 021522 000010 $TTYIN: .BLKB 8. ::RESERVE 8 BYTES FOR TTY INPUT
(1) 021532 041536 005015 000 $CNTLC: .ASCIZ /^C/<15><12> ::CONTROL 'C'
(1) 021537 136 006525 000012 $CNTLU: .ASCIZ /^U/<15><12> ::CONTROL 'U'
(1) 021544 043536 005015 000 $CNTLG: .ASCIZ /^G/<15><12> ::CONTROL 'G'
(1) 021551 015 051412 051127 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 021556 036440 000040 $MNEW: .ASCIZ / NEW = /
(1) 021562 020040 042516 020127
(1) 021570 020075 000
(1) 021574 000002
2383 .EVEN
.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) 021574 011646 $RDOCT: MOV (SP)-,(SP) ::PROVIDE SPACE FOR THE
(1) 021576 016666 000004 000002 MOV 4(SP),2(SP) ::INPUT NUMBER
(3) 021604 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
(3) 021606 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
(3) 021610 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
(1) 021612 104411 1$: RDLIN ::READ AN ASCII LINE
(1) 021614 012600 MOV (SP)+,R0 ::GET ADDRESS OF 1ST CHARACTER
(1) 021616 005001 CLR R1 ::CLEAR DATA WORD
(1) 021620 005002 CLR R2
(1) 021622 112046 2$: MOVB (R0)+,-(SP) ::PICKUP THIS CHARACTER
(1) 021624 001412 BEQ 3$ ::IF ZERO GET OUT
(1) 021626 006301 ASL R1 ::*2
(1) 021630 006102 ROL R2
(1) 021632 006301 ASL R1 ::*4
(1) 021634 006102 ROL R2
(1) 021636 006301 ASL R1 ::*8

```

```

(1) 021640 006102          RCL      R2
(1) 021642 042716 177770   BIC      #'(7,(SP)      ::STRIP THE ASCII JUNK
(1) 021646 062601          ADD      (SP)+,R1      ::ADD IN THIS DIGIT
(1) 021650 000764          BR       2$            ::LOOP
(1) 021652 005726          3$: TST      (SP)+      ::CLEAN TERMINATOR FROM STACK
(1) 021654 010166 000012   MOV      R1,12(SP)    ::SAVE THE RESULT
(1) 021660 010237 021674   MOV      R2,$SHIOCT
(3) 021664 012602          MOV      (SP)+,R2     ::POP STACK INTO R2
(3) 021666 012601          MOV      (SP)+,R1     ::POP STACK INTO R1
(3) 021670 012600          MOV      (SP)+,R0     ::POP STACK INTO R0
(1) 021672 000002          RTI
(1) 021674 000000          $SHIOCT: .WORD 0      ::HIGH ORDER BITS GO HERE
2384 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)
(1) *****
(1) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1) *OCTAL (ASCII) NUMBER AND TYPE IT.
(1) *$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) 021676 017646 000000   $TYPOS: MOV      @ (SP),-(SP)    ::PICKUP THE MODE
(1) 021702 116637 000001 022121 MOVB     1(SP), $OFILL      ::LOAD ZERO FILL SWITCH
(1) 021710 112637 022123   MOVB     (SP)+, $OMODE+1   ::NUMBER OF DIGITS TO TYPE
(1) 021714 062716 000002   ADD      #2,(SP)         ::ADJUST RETURN ADDRESS
(1) 021720 000406          BR       $TYPON
(1) 021722 112737 000001 022121 $TYPOC: MOVB     #1, $OFILL      ::SET THE ZERO FILL SWITCH
(1) 021730 112737 000006 022123 MOVB     #6, $OMODE+1     ::SET FOR SIX(6) DIGITS
(1) 021736 112737 000005 022120 $TYPON: MOVB     #5, $OCNT      ::SET THE ITERATION COUNT
(1) 021744 010346          MOV      R3,-(SP)      ::SAVE R3
(1) 021746 010446          MOV      R4,-(SP)      ::SAVE R4
(1) 021750 010546          MOV      R5,-(SP)      ::SAVE R5
(1) 021752 113704 022123   MOVB     $OMODE+1,R4    ::GET THE NUMBER OF DIGITS TO TYPE
(1) 021756 005404          NEG      R4
(1) 021760 062704 000006   ADD      #6,R4          ::SUBTRACT IT FOR MAX. ALLOWED
(1) 021764 110437 022122   MOVB     R4, $OMODE      ::SAVE IT FOR USE
(1) 021770 113704 022121   MOVB     $OFILL,R4      ::GET THE ZERO FILL SWITCH
(1) 021774 016605 000012   MOV      12(SP),R5     ::PICKUP THE INPUT NUMBER
(1) 022000 005003          CLR      R3            ::CLEAR THE OUTPUT WORD
(1) 022002 006105          1$: ROL      R5          ::ROTATE MSB INTO 'C'

```



```

(1) 022160 005000      1$: CLR R0           ;;ZERO THE CONSTANTS INDEX
(1) 022162 012703 022340 MOV #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
(1) 022166 112723 000040 MOVB #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
(1) 022172 005002      2$: CLR R2           ;;CLEAR THE BCD NUMBER
(1) 022174 016001 022330 MOV $DTBL(R0),R1    ;;GET THE CONSTANT
(1) 022200 160105      3$: SUB R1,R5          ;;FORM THIS BCD DIGIT
(1) 022202 002402      BLT 4$           ;;BR IF DONE
(1) 022204 005202      INC R2           ;;INCREASE THE BCD DIGIT BY 1
(1) 022206 000774      BR 3$
(1) 022210 060105      4$: ADD R1,R5          ;;ADD BACK THE CONSTANT
(1) 022212 005702      TST R2           ;;CHECK IF BCD DIGIT=0
(1) 022214 001002      BNE 5$           ;;FALL THROUGH IF 0
(1) 022216 105716      TSTB (SP)        ;;STILL DOING LEADING 0'S?
(1) 022220 100407      BMI 7$           ;;BR IF YES
(1) 022222 106316      5$: ASLB (SP)        ;;MSD?
(1) 022224 103003      BCC 6$           ;;BR IF NO
(1) 022226 116663 000001 177777 MOVB 1(SP),-1(R3)   ;;YES--SET THE SIGN
(1) 022234 052702 000060 6$: BIS #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1) 022240 052702 000040 7$: BIS #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 022244 110223      MOVB R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 022246 005720      TST (R0)+         ;;JUST INCREMENTING
(1) 022250 020027 000010      CMP R0,#10       ;;CHECK THE TABLE INDEX
(1) 022254 002746      BLT 2$           ;;GO DO THE NEXT DIGIT
(1) 022256 003002      BGT 8$           ;;GO TO EXIT
(1) 022260 010502      MOV R5,R2         ;;GET THE LSD
(1) 022262 000764      BR 6$           ;;GO CHANGE TO ASCII
(1) 022264 105726      8$: TSTB (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 022266 100003      BPL 9$           ;;BR IF NO
(1) 022270 116663 177777 177776 9$: MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 022276 105013      CLRB (R3)        ;;SET THE TERMINATOR
(3) 022300 012605      MOV (SP)+,R5     ;;POP STACK INTO R5
(3) 022302 012603      MOV (SP)+,R3     ;;POP STACK INTO R3
(3) 022304 012602      MOV (SP)+,R2     ;;POP STACK INTO R2
(3) 022306 012601      MOV (SP)+,R1     ;;POP STACK INTO R1
(3) 022310 012600      MOV (SP)+,R0     ;;POP STACK INTO R0
(1) 022312 104401 022340      TYPE $SDBLK      ;;NOW TYPE THE NUMBER
(1) 022316 016666 000002 000004 MOV 2(SP),4(SP)   ;;ADJUST THE STACK
(1) 022324 012616      MOV (SP)+,(SP)
(1) 022326 000002      RTI           ;;RETURN TO USER
(1) 022330 023420      $DTBL: 1000.
(1) 022332 001750      1000.
(1) 022334 000144      100.
(1) 022336 000012      10.
(1) 022340 000004      $SDBLK: .BLKW 4

```

2386

.SBTTL ROUTINE TO SIZE MEMORY

```

(1)
(1) *****
(1) *CALL:
(1) * JSR PC,$SIZE
(1) * RETURN
(1) *$LSTAD WILL CONTAIN:
(1) * WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK
(1) * WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
(1) *$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
(1) *$KT11 IS THE MEMORY MANAGEMENT KEY
(1) *BIT07 = 0 DON'T USE MEMORY MANAGEMENT

```



```

(1) ;* MUST BE SETUP BEFORE THE CALL
(1) ;*+BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
(1) ;* DETERMINED BY ROUTINE
(1) $SIZE: MOV R0,-(SP) ;:SAVE R0 ON THE STACK
(1) MOV R1,-(SP) ;:SAVE R1 ON THE STACK
(1) MOV R2,-(SP) ;:SAVE R2 ON THE STACK
(1) MOV R3,-(SP) ;:SAVE R3 ON THE STACK
(1) MOV @#ERRVEC,-(SP) ;:SAVE PRESENT ERROR VECTOR PS & PC
(1) MOV @#ERRVEC+2,-(SP)
(1) MOV SP,R0 ;:SAVE THE STACK POINTER
(1) ;:SET THE ERRVEC PS TO THE PRESENT PS
(2) TRAP ;:PUSH OLD PSW AND PC ON STACK
(1) MOV (SP)+,@#ERRVEC+2 ;:SAVE THE PSW IN @#ERRVEC+2
(1) MOV #3776,R1 ;:SETUP ADDRESS
(1) TSTB (PC)+ ;:USE MEMORY MANAGEMENT?
(1) $KT11: .WORD 200 ;:SET TO USE MEMORY MANAGEMENT
(1) BPL $SCORE ;:BR IF NO
(1) MOV #SKTNEX,@#ERRVEC ;:SET FOR TIMEOUT
(1) TST @#SR0 ;:KT11 ARE YOU THERE?
(1) BIS #100000,$KT11 ;:YES--SET KT11 KEY
(1) CLR -(SP) ;:INITIALIZE FOR 'PAR' LOADING
(1) MOV #KIPAR0,R2 ;:ADDRESS OF FIRST 'PAR'
(1) MOV #*DB,R3 ;:LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
(1) MOV #77406,-40(R2) ;:PDR = 4K, UP, READ/WRITE
(1) MOV (SP),(R2)+ ;:LOAD 'PAR'
(1) ADD #200,(SP) ;:UPDATE FOR NEXT 'PAR'
(1) SOB R3,1$ ;:LOOP UNTIL ALL EIGHT ARE LOADED
(1) MOV #177600,-(R2) ;:SETUP KIPAR7 FOR I/O
(1) CLR -(R2) ;:SETUP KIPAR6 FOR TESTING
(1) MOV #2$,@#ERRVEC ;:CATCH TIMEOUT IF NO SR3
(1) MOV #20,@#SR3 ;:ENABLE 22 BIT MODE
(1) BR 3$ ;:THIS PDP-11 HAS A SR3 REGISTER
(1) CMP (SP)+,(SP)+ ;:CLEAN OFF THE STACK--NO SR3
(1) INC @#SR0 ;:TURN ON MEMORY MANAGEMENT
(1) MOV #SKTOUT,@#ERRVEC ;:SET FOR TIME OUT
(1) TST @#143776 ;:TRAP ON NON-EX-MEM
(1) ADD #40,(R2) ;:MAKE A 1K STEP
(1) CMP @#KIPAR7,(R2) ;:LAST ONE?
(1) BHI 4$ ;:NO--TRY IT
(1) $KTOUT: MOV (R2),R2 ;:GET LAST BANK+1
(1) CLR @#SR0 ;:TURN OFF MEMORY MANAGEMENT
(1) BR $SIZE
(1) $KTNEX: BIC #100000,$KT11 ;:KT11 NON-EXISTENT
(1) $SCORE: MOV #SKROUT,@#ERRVEC ;:SET FOR TIMEOUT
(1) CLR R2 ;:SET UP BANK
(1) ADD #4000,R1 ;:INCREMENT BY 1K
(1) ADD #40,R2 ;:1K STEP
(1) TST (R1) ;:TRAP ON TIME OUT
(1) CMP #177776,R1 ;:LAST ONE
(1) BNE 1$ ;:NO--TRY AGAIN
(1) $SKROUT: SUB #4000,R1
(1) $SIZE: SUB #40,R2 ;:DROP BACK
(1) MOV R0,SP ;:RESTORE THE STACK
(1) MOV (SP)+,@#ERRVEC+2 ;:RESTORE ERROR VECTOR
(1) MOV (SP)+,@#ERRVEC

```

```

(1) 022630 010137 022652          MOV    R1,$LSTAD      ;;LAST ADDRESS
(1) 022634 010237 022654          MOV    R2,$LSTBK     ;;LAST BANK
(1) 022640 012603                 MOV    (SP)+,R3      ;;RESTORE R3
(1) 022642 012602                 MOV    (SP)+,R2      ;;RESTORE R2
(1) 022644 012601                 MOV    (SP)+,R1      ;;RESTORE R1
(1) 022646 012600                 MOV    (SP)+,R0      ;;RESTORE R0
(1) 022650 000207                 RTS     PC
(1) 022652 000000          $LSTAD: .WORD 0      ;;CONTAINS THE LAST ADDRESS
(1) 022654 000000          $LSTBK: .WORD 0      ;;CONTAINS THE LAST BANK
2387          .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)          $ERROR:
(1) 022656 104407          CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 022660 113737 001102 024042      MOVB    $TSTNM,TSTNUM ;;SET UP TEST # ON ERROR
(1) 022666 105237 001103          7$: INCB    $ERFLG      ;;SET THE ERROR FLAG
(1) 022672 001775          BEQ     7$           ;;DON'T LET THE FLAG GO TO ZERO
(1) 022674 013777 001102 156240      MOV    $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 022702 032777 002000 156230      BIT    #BIT10,@SWR    ;;BELL ON ERROR?
(1) 022710 001402          BEQ     1$           ;;NO - SKIP
(1) 022712 104401 001176          TYPE    ,SBELL      ;;RING BELL
(1) 022716 005237 001112          1$: INC     $ERTTL     ;;COUNT THE NUMBER OF ERRORS
(1) 022722 011637 001116          MOV    (SP),$ERRPC   ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 022726 162737 000002 001116      SUB    #2,$ERRPC
(1) 022734 117737 156156 001114      MOVB   @ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 022742 032777 020000 156170      BIT    #BIT13,@SWR   ;;SKIP TYPEOUT IF SET
(1) 022750 001055          BNE    20$           ;;SKIP TYPEOUTS
(1) 022752 021627 001002          CMP    (SP),#1002   ;;IF RETURN PC LESS THAN 1002
(1) 022756 101046          BHI    12$           ;;ERROR IS ILLEGAL TRAP
(1)          ;;:PROCESS UNEXPECTED TRAP OR INTERRUPT
(1) 022760 016637 000004 001116      MOV    4(SP),$ERRPC  ;;GET PC AT TIME OF FALSE TRAP
(1) 022766 162737 000002 001116      SUB    #2,$ERRPC    ;;ADJUST PC
(1) 022774 104401 023040          TYPE    ,10$        ;;TYPE HEADER
(2) 023000 013746 001116          MOV    $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
(2) 023004 104402          TYPE    ,11$        ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 023006 104401 023046          TYPE    ,11$
(1) 023012 162716 000004          SUB    #4,(SP)      ;;GET FALSE TRAP VECTOR ADDR
(1) 023016 011637 001116          MOV    (SP),$ERRPC
(2) 023022 013746 001116          MOV    $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
(2) 023026 104402          TYPE    ,11$        ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 023030 104401 001203          TYPE    ,11$
(1) 023034 022626          CMP    (SP)+,(SP)+  ;;POP FALSE TRAP VECTOR PC&ADDR
(1) 023036 000422          BR     20$
(1) 023040 050200 036503 000040 10$: .ASCIZ <200>'PC= '
(1) 023046 020040 047125 054105 11$: .ASCIZ ' UNEXPECTED TRAP TO '

```

```

(1) 023054 042520 052103 042105
(1) 023062 052040 040522 020120
(1) 023070 047524 000040
(1)
(1) 023074 12$: .EVEN
(1) 023074 004737 023206 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
(1) 023100 104401 001203 TYPE ,SCLF
(1) 023104 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 023104 122737 000001 001226 BNE 2$ ;;NO,SKIP APT ERROR REPORT
(1) 023112 001007 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 023114 113737 001114 023126 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
(1) 023122 004737 020502
(1) 023126 000 21$: .BYTE 0
(1) 023127 000 .BYTE 0
(1) 023130 000777 22$: BR 22$ ;;APT ERROR LOOP
(1) 023132 005777 156002 2$: TST @SWR ;;HALT ON ERROR
(1) 023136 100002 BPL 3$ ;;SKIP IF CONTINUE
(1) 023140 000000 HALT ;;HALT ON ERROR!
(1) 023142 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 023144 032777 001000 155766 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(1) 023152 001402 BEQ 4$ ;;BR IF NO
(1) 023154 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 023160 005737 001174 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(1) 023164 001402 BEQ 5$ ;;BR IF NONE
(1) 023166 013716 001174 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 023172
(1) 023172 022737 020146 000042 5$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 023200 001001 BNE 6$ ;;BRANCH IF NO
(1) 023202 000000 HALT ;;YES
(1) 023204
(1) 023204 000002 6$: RTI ;;RETURN
2388 .SBTTL ERROR MESSAGE TIMEOUT ROUTINE
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 023206 $ERRTYP:
(1) 023206 104401 001203 TYPE ,SCLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 023212 010046 MOV RO,-(SP) ;;SAVE RO
(1) 023214 005000 CLR RO ;;PICKUP THE ITEM INDEX
(1) 023216 153700 001114 BISB @#$ITEMB,RO
(1) 023222 001004 BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
(2) 023224 013746 001116 MOV $ERRPC,-(SP) ;;TYPE THE PC OF THE ERROR
(2) 023230 104402 TYPOC ;;SAVE $ERRPC FOR TYPEOUT
(1) 023232 000426 BR 6$ ;;ERROR ADDRESS
(1) 023234 005300 1$: DEC RO ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 023236 006300 ASL RO ;;GET OUT
(1) 023240 006300 ASL RO ;;ADJUST THE INDEX SO THAT IT WILL
(1) 023242 006300 ASL RO ;; WORK FOR THE ERROR TABLE
(1) 023244 062700 001332 ADD #$ERRTB,RO ;;FORM TABLE POINTER
(1) 023250 012037 023260 MOV (RO)+,2$ ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 023254 001404 BEQ 3$ ;;SKIP TYPEOUT IF NO POINTER

```

```
(1) 023256 104401                TYPE                                ;;TYPE THE 'ERROR MESSAGE'  
(1) 023260 000000                2$: .WORD 0                        ;;'ERROR MESSAGE' POINTER GOES HERE  
(1) 023262 104401 001203          TYPE ,SCRLF                          ;;'CARRIAGE RETURN' & 'LINE FEED'  
(1) 023266 012037 023276          3$: MOV (R0)+,4$                     ;;PICKUP 'DATA HEADER' POINTER  
(1) 023272 001404                BEQ 5$                                ;;SKIP TYPEOUT IF 0  
(1) 023274 104401                TYPE                                ;;TYPE THE 'DATA HEADER'  
(1) 023276 000000                4$: .WORD 0                        ;;'DATA HEADER' POINTER GOES HERE  
(1) 023300 104401 001203          TYPE ,SCRLF                          ;;'CARRIAGE RETURN' & 'LINE FEED'  
(1) 023304 011000                5$: MOV (R0),R0                       ;;PICKUP 'DATA TABLE' POINTER  
(1) 023306 001004                BNE 7$                                ;;GO TYPE THE DATA  
(1) 023310 012600                MOV (SP)+,R0                          ;;RESTORE R0  
(1) 023312 104401 001203          TYPE ,SCRLF                          ;;'CARRIAGE RETURN' & 'LINE FEED'  
(1) 023316 000207                RTS PC                                ;;RETURN  
(1) 023320                7$:  
(2) 023320 013046                MOV @ (R0)+,-(SP)                    ;;SAVE @ (R0)+ FOR TYPEOUT  
(2) 023322 104402                TPOC                                  ;;GO TYPE--OCTAL ASCII(ALL DIGITS)  
(1) 023324 005710                TST (R0)                              ;;IS THERE ANOTHER NUMBER?  
(1) 023326 001770                BEQ 6$                                ;;BR IF NO  
(1) 023330 104401 023336          TYPE 8$                               ;;TYPE TWO(2) SPACES  
(1) 023334 000771                BR 7$                                 ;;LOOP  
(1) 023336 020040 000            8$: .ASCII / /                          ;;TWO(2) SPACES  
(1) 023342 023342                .EVEN  
2389 .SBTTL SCOPE HANDLER ROUTINE  
(1)  
(2) ;;*****  
(1) ;;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT  
(1) ;;*AND LOAD THE TEST NUMBER($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) 023342 $SCOPE:  
(1) 023342 104407                CKSWR ;;TEST FOR CHANGE IN SOFT-SWR  
(1) ;;GO TO ERROR ROUTINE IF RETURN PC LESS THAN 1002  
(1) ;;OTHERWISE CONTINUE  
(1) 023344 021627 001002          CMP (SP),#1002 ;;UNEXPECTED TRAP OR INTERRUPT  
(1) 023350 101002                BHI 1$                                ;;ARE TRAPPED HERE VIA IOT  
(1) 023352 000137 022656          JMP $ERROR ;;GO PROCESS UNEXPECTED TRAP  
(1) 023356 032777 040000 155554 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?  
(1) 023364 001114                BNE $OVER ;;YES IF SW14=1  
(1) ;;#####START OF CODE FOR THE XOR TESTER#####  
(1) 023366 000416          $XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE  
(1) ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)  
(1) 023370 013746 000004          MOV @ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR  
(1) 023374 012737 023414 000004  MOV #5$,@ERRVEC ;;SET FOR TIMEOUT  
(1) 023402 005737 177060          TST @#177060 ;;TIME OUT ON XOR?  
(1) 023406 012637 000004          MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR  
(1) 023412 000463                BR ;;GO TO THE NEXT TEST  
(1) 023414 022626                5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT  
(1) 023416 012637 000004          MOV (SP)+,@ERRVEC ;;RESTORE THE ERROR VECTOR  
(1) 023422 000423                BR 7$ ;;LOOP ON THE PRESENT TEST
```

```

(1) 023424          6S:;##### END OF CODE FOR THE XOR TESTER#####
(1) 023424 032777 000400 155506 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
(1) 023432 001404 BEQ 2$ ;;BR IF NO
(1) 023434 127737 155500 001102 CMPB @SWR,$STNM ;;ON THE RIGHT TEST? SWR<7:0>
(1) 023442 001465 BEQ $OVER ;;BR IF YES
(1) 023444 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
(1) 023450 001421 BEQ 3$ ;;BR IF NO
(1) 023452 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 023460 101015 BHI 3$ ;;BR IF NO
(1) 023462 032777 001000 155450 BIT #BIT09,@SWR ;;LOOP ON ERROR?
(1) 023470 001404 BEQ 4$ ;;BR IF NO
(1) 023472 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 023500 000446 BR $OVER
(1) 023502 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
(1) 023506 005037 001172 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 023512 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
(1) 023514 032777 004000 155416 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1) 023522 001011 BNE 1$ ;;BR IF YES
(1) 023524 005737 001214 TST $PASS ;;IF FIRST PASS OF PROGRAM
(1) 023530 001406 BEQ 1$ ;; INHIBIT ITERATIONS
(1) 023532 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 023536 023737 001172 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 023544 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 023546 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1) 023554 013737 023632 001172 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 023562 105237 001102 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS
(1) 023566 113737 001102 001212 MOVB $STNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(1) 023574 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(1) 023600 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(1) 023604 005037 001174 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 023610 112737 000001 001115 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 023616 013777 001102 155316 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 023624 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(1) 023630 000002 RTI ;;FIXES PS
(1) 023632 003720 $MXCNT: 2000. ;;MAX. NUMBER OF ITERATIONS

```

```

2390 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)
(1)
(1) 023634 012737 024000 000024 $PWRDN: MOV # $ILLUP,@PWRVEC ;;SET FOR FAST UP
(1) 023642 012737 000340 000026 MOV #340,@PWRVEC+2 ;;PRIO:7
(3) 023650 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 023652 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 023654 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 023656 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 023660 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
(3) 023662 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(3) 023664 017746 155250 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
(1) 023670 010637 024004 MOV SP,$SAVR6 ;;SAVE SP
(1) 023674 012737 023706 000024 MOV # $PWRUP,@PWRVEC ;;SET UP VECTOR
(1) 023702 000000 HALT
(1) 023704 000776 BR -2 ;;HANG UP
(1)
(2)
(1)
(1) 023706 012737 024000 000024 $PWRUP: MOV # $ILLUP,@PWRVEC ;;SET FOR FAST DOWN

```

```

(1) 023714 013706 024004      MOV    $$SAVR6,SP      ;;GET SP
(1) 023720 005037 024004      CLR    $$SAVR6        ;;WAIT LOOP FOR THE TTY
(1) 023724 005237 024004      1$:   INC    $$SAVR6    ;;WAIT FOR THE INC
(1) 023730 001375              BNE    1$             ;;OF WORD
(3) 023732 012677 155202      MOV    (SP)+,@SWR     ;;POP STACK INTO @SWR
(3) 023736 012605              MOV    (SP)+,R5       ;;POP STACK INTO R5
(3) 023740 012604              MOV    (SP)+,R4       ;;POP STACK INTO R4
(3) 023742 012603              MOV    (SP)+,R3       ;;POP STACK INTO R3
(3) 023744 012602              MOV    (SP)+,R2       ;;POP STACK INTO R2
(3) 023746 012601              MOV    (SP)+,R1       ;;POP STACK INTO R1
(3) 023750 012600              MOV    (SP)+,R0       ;;POP STACK INTO R0
(1) 023752 012737 023634 000024  MOV    #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 023760 012737 000340 000026  MOV    #340,@#PWRVEC+2 ;;PRIO:7
(1) 023766 104401              TYPE                    ;;REPORT THE POWER FAILURE
(1) 023770 024006      $PWRMG: .WORD    PWRMSG      ;;POWER FAIL MESSAGE POINTER
(1) 023772 012716      MOV    (PC)+,(SP)      ;;RESTART AT START
(1) 023774 001712      $PWRAD: .WORD    START     ;;RESTART ADDRESS
(1) 023776 000002      RTI
(1) 024000 000000      $ILLUP: HALT           ;;THE POWER UP SEQUENCE WAS STARTED
(1) 024002 000776      BR    .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 024004 000000      $$SAVR6: 0            ;;PUT THE SP HERE
2391 024006 005015 042522 052123  PWRMSG: .ASCIZ <15><12>/RESTARTED FROM POWER FAIL/
      024014 051101 042524 020104
      024022 051106 046517 050040
      024030 053517 051105 043040
      024036 044501 000114

2392
2393      : KW11C BUS REGISTER ADDRESS POINTERS
2394
2395
2396      : KW11B VECTOR ADDRESS POINTERS
2397
2398
2399      : KW11B DEVICE LEVEL
2400
2401
2402      : COMMAND STATUS REGISTER BIT ASSIGNMENTS
2403
2404      : COMMON PROGRAM LOCATION(S)
2405
2406 024042 000000      TSTNUM: 0            ;CONTAINS TEST NUMBER ON ERROR
2407      : *****
2408      : PROGRAM PARAMETER BLOCK
2409 024044 000000      KONST: 0
2410 024046 000000      NUMBR: 0
2411 024050 000000      KOUNT: 0
2412 024052 000000      COUNTR: 0
2413 024054 000170      CONST: 170
2414
2415 024056 000000      READA: 0
2416 024060 000000      READB: 0
2417
2418      : *****DEVICE ADDRESSES*****
2419
2420 024062 160206      CSR: 160206
2421 024064 160000      RGO: 160000
    
```

2422	024066	160002	RG2:	160002
2423	024070	160004	RG4:	160004
2424			:	
2425			:	
2426			:	
2427	024072	000300	PCV0:	300
2428	024074	000340	PSV0:	340
2429	024076	000304	PCV1:	304
2430	024100	000340	PSV1:	340
2431			:	
2432			:	
2433			:	
2434			:	
2435	024102	000006	BRLV:	6
2436			:	
2437			:	
2438	024104	000000	PRI:	0000
2439	024106	000240	DPRI:	240
2440	024110	000007	LEVEL:	7
2441	024112	000000	SFTSW:	0
2442	024114	000000	SAVR1:	0
2443	024116	000000	HOLD:	0
2444	024120	000000	ACSR:	0
2445	024122	000000	OCTNUM:	0
2446	024124	000000	COMMA:	0
2447	024126	000000	DIGCNT:	0
2448	024130	000000	INSWCH:	0
2449	024132	000000	YEAR:	0
2450	024134	000000	MONTH:	0
2451	024136	000000	DAY:	0
2452	024140	000000	HOUR:	0
2453	024142	000000	MIN:	0
2454	024144	000000	SEC:	0
2455	024146	000000	TIMINV:	0
2456	024150	000000	DMONTH:	0
2457	024152	000000	DHOUR:	0
2458	024154	000000	DTIM:	0
2459	024156	000000	STR:	0
2460	024160	000000	NUM:	0
2461	024162	000000	CNT1:	0
2462	024164	000000	COUNT:	0
2463	024166	000000	COUNTER:	0
2464	024170	000000	SFSWR:	0
2465	024172	000000	SWITCH:	0
2466	024174	000000	CNT:	0
2467	024176	000000	TEMP:	0
2468	024200	000000	ADR:	0
2469	024202	000000	EXPT:	0
2470	024204	000000	SAFE:	0
2471	024206	000000	XX:	00
2472	024210	000000	FLAG.1:	0
2473	024212	000000	FLAG.2:	0
2474	024214	000000	GOOD1:	0
2475	024216	000000	TEMP1:	0
2476	024220	000000	TEMP2:	0
2477	024222	000000	FLAG:	0

***** INTERRUPT VECTOR ADDRESSES *****

***** BUS REQUEST LEVEL *****

***** OTHER ADDRESSES *****

.MAIN. MACY11 30A(1052) 13-MAR-80 07:40 PAGE 5-23
CZKWL.A.P11 13-MAR-80 07:39 POWER DOWN AND UP ROUTINES

SEQ 0080

```

2478 024224 000000      INTSW:  0
2479 024226 000000      INTREG: 0
2480 024230 000000      SOFSWT: 0
2481 024232 000000      LASTPC: 0
2482 024234 000000      TIMEIN: 0
2483
2484 024236 000000      LOC:    0
2485 024240 000000      PT:     0
2486 024242 000074      HZ:     74
2487 024244      000    200    000  ECHOB:  .BYTE  0,200,0
2488      024250      .EVEN
2489 024250 000000      ECHOB1: 0
2490      .EVEN
2491 024252 000000      ERRPC: 00
2492 024254 000000      DIGIT:  0
2493 024256 000000      0
2494 024260 000000      0
2495 024262 000000      0
2496 024264 000000      0
2497 024266 000000      0
2498 024270 000000      0
2499 024272 000000      0
2500 024274 000000      0
2501 024276 000000      0
2502 024300 000000      0
2503 024302 000000      0
2504 024304 000000      0
2505 024306 000000      0
2506 024310 000000      0
2507 024312 000000      0
2508 024314 000000      0
2509 024316 000000      DIGEND: 0
2510 024320 000000      0
2511 024322 000000      0
2512      .EVEN
2513      .SBTTL TRAP DECODER
(1)
(2)
(1)      ::*****
(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) 024324 010046      $TRAP: MOV    RO,-(SP)      ;;SAVE RO
(1) 024326 016600 000002      MOV    2(SP),RO        ;;GET TRAP ADDRESS
(1) 024332 005740      TST    -(RO)           ;;BACKUP BY 2
(1) 024334 111000      MOVSB (RO),RO          ;;GET RIGHT BYTE OF TRAP
(1) 024336 006300      ASL    RO               ;;POSITION FOR INDEXING
(1) 024340 016000 024360      MOV    $TRPAD(RO),RO   ;;INDEX TO TABLE
(1) 024344 000200      RTS    RO              ;;GO TO ROUTINE
(1)
(1)      ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
(1) 024346 011646      $TRAP2: MOV   (SP),-(SP)  ;;MOVE THE PC DOWN
(1) 024350 016666 000004 000002      MOV    4(SP),2(SP)     ;;MOVE THE PSW DOWN

```



```

(1) 024356 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)          ;
(3)          ; ROUTINE
(3)          ;-----
(3) 024360 024346          $TRPAD: .WORD $TRAP2
(3) 024362 020202          $TYPE  ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
(3) 024364 021722          $TYPOC ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 024366 021676          $TYPOS ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 024370 021736          $TYPON ;;CALL=TYPON       TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 024372 022124          $TYPDS ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(1)
(3) 024374 021002          $GTSWR ;;CALL=GTSWR     TRAP+6(104406) GET SOFT-SWR SETTING
(1)
(3) 024376 020732          $CKSWR ;;CALL=CKSWR     TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
(3) 024400 021254          $RDCHR ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3) 024402 021374          $RDLIN ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3) 024404 021574          $RDOCT ;;CALL=RDOCT     TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
2514
2515
2516
2517 024406 042522 042101 053040 EM1: .SBTTL ASCII MESSAGES
      024414 046101 042525 042040 .ASCIZ /READ VALUE DIDN'T MATCH WITH EXPECTED/
      024422 042111 023516 020124
      024430 040515 041524 020110
      024436 044527 044124 042440
      024444 050130 041505 042524
      024452 000104
2518 024454 042522 042101 020131 EM2: .ASCIZ /READY BIT FAIL TO SET/
      024462 044502 020124 040506
      024470 046111 052040 020117
      024476 042523 000124
2519 024502 044524 042515 044440 EM3: .ASCIZ /TIME INTERVAL COUNTER DOESN'T DECREMENT IN ORDER/
      024510 052116 051105 040526
      024516 020114 047503 047125
      024524 042524 020122 047504
      024532 051505 023516 020124
      024540 042504 051103 046505
      024546 047105 020124 047111
      024554 047440 042122 051105
      024562 000
2520 024563 102 052111 021440 EM4: .ASCIZ /BIT #14 IN CSR DIDN'T SET/
      024570 032061 044440 020116
      024576 051503 020122 044504
      024604 047104 052047 051440
      024612 052105 000
2521 024615 102 052111 021440 EM5: .ASCIZ /BIT #14 IN CSR DIDN'T RESET/
      024622 032061 044440 020116
      024630 051503 020122 044504
      024636 047104 052047 051040
      024644 051505 052105 000
2522 024651 122 040505 054504 EM6: .ASCIZ /READY BIT DIDN'T SET;YEAR VALUE MAY BE INVALID/

```

	024656	041040	052111	042040		
	024664	042111	023516	020124		
	024672	042523	035524	042531		
	024700	051101	053040	046101		
	024706	042525	046440	054501		
	024714	041040	020105	047111		
	024722	040526	044514	000104		
2523	024730	042522	042101	020131	EM7:	.ASCII /READY BIT DIDN'T SET;ONE OR BOTH VALUES/<200>
	024736	044502	020124	044504		
	024744	047104	052047	051440		
	024752	052105	047473	042516		
	024760	047440	020122	047502		
	024766	044124	053040	046101		
	024774	042525	100123			
2524	025000	043117	046440	047117		.ASCIZ /OF MONTH AND DAY MAY BE INVALID/
	025006	044124	040440	042116		
	025014	042040	054501	046440		
	025022	054501	041040	020105		
	025030	047111	040526	044514		
	025036	000104				
2525	025040	042522	042101	020131	EM10:	.ASCII /READY BIT DIDN'T SET;ONE OR BOTH VALUES/<200>
	025046	044502	020124	044504		
	025054	047104	052047	051440		
	025062	052105	047473	042516		
	025070	047440	020122	047502		
	025076	044124	053040	046101		
	025104	042525	100123			
2526	025110	043117	044040	052517		.ASCIZ /OF HOUR AND MINUTE MAY BE INVALID/
	025116	020122	047101	020104		
	025124	044515	052516	042524		
	025132	046440	054501	041040		
	025140	020105	047111	040526		
	025146	044514	000104			
2527	025152	051105	047522	035122	EM11:	.ASCIZ /ERROR: BIT #15 (DE) FAIL TO SET/
	025160	041040	052111	021440		
	025166	032461	024040	042504		
	025174	020051	040506	046111		
	025202	052040	020117	042523		
	025210	000124				
2528	025212	051105	047522	035122	EM12:	.ASCIZ /ERROR: CLOCK IS STILL RUNNING/
	025220	041440	047514	045503		
	025226	044440	020123	052123		
	025234	046111	020114	052522		
	025242	047116	047111	000107		
2529	025250	051105	047522	035122	EM13:	.ASCIZ /ERROR: CLOCK IS NOT RUNNING/
	025256	041440	047514	045503		
	025264	044440	020123	047516		
	025272	020124	052522	047116		
	025300	047111	000107			
2530	025304	047111	042524	051122	EM14:	.ASCII /INTERRUPT FAIL TO OCCUR WHILE UPDATING THE CLOCK /<200>
	025312	050125	020124	040506		
	025320	046111	052040	020117		
	025326	041517	052503	020122		
	025334	044127	046111	020105		
	025342	050125	040504	044524		
	025350	043516	052040	042510		

	025356	041440	047514	045503	
	025364	100040			
2531	025366	051120	041117	041101	.ASCIZ /PROBABLE CAUSE INTERRUPT WAS NOT ENABLED/
	025374	042514	041440	052501	
	025402	042523	044440	052116	
	025410	051105	052522	052120	
	025416	053440	051501	047040	
	025424	052117	042440	040516	
	025432	046102	042105	000	
2532	025437	120	042522	042523	EM15: .ASCII /PRESET INTERRUPT DIDN'T OCCUR;PROBABLE CAUSE/<200>
	025444	020124	047111	042524	
	025452	051122	050125	020124	
	025460	044504	047104	052047	
	025466	047440	041503	051125	
	025474	050073	047522	040502	
	025502	046102	020105	040503	
	025510	051525	100105		
2533	025514	047111	051124	052120	.ASCIZ /INTRPT NOT ENABLE OR AND INVALID PRESET VALUE/
	025522	047040	052117	042440	
	025530	040516	046102	020105	
	025536	051117	040440	042116	
	025544	044440	053116	046101	
	025552	042111	050040	042522	
	025560	042523	020124	040526	
	025566	052514	000105		
2534	025572	047523	052106	042440	EM16: .ASCIZ /SOFT ERROR/
	025600	051122	051117	000	
2535	025605	125	042516	050130	EM17: .ASCIZ /UNEXPECTED INTERRUPT AS SOFTSWT NOT SET/
	025612	041505	042524	020104	
	025620	047111	042524	051122	
	025626	050125	020124	051501	
	025634	051440	043117	051524	
	025642	052127	047040	052117	
	025650	051440	052105	000	
2536	025655	125	042516	050130	EM20: .ASCIZ /UNEXPECTED INTERRUPT AS INTRP WAS NOT ENABLED/
	025662	041505	042524	020104	
	025670	047111	042524	051122	
	025676	050125	020124	051501	
	025704	044440	052116	050122	
	025712	053440	051501	047040	
	025720	052117	042440	040516	
	025726	046102	042105	000	
2537	025733	125	042516	050130	EM21: .ASCIZ /UNEXPECTED INTERRUPT AS (DE) WAS NOT SET/
	025740	041505	042524	020104	
	025746	047111	042524	051122	
	025754	050125	020124	051501	
	025762	024040	042504	020051	
	025770	040527	020123	047516	
	025776	020124	042523	000124	
2538	026004	047125	054105	042520	EM22: .ASCIZ /UNEXPECTED AND UNEXPLAINABLE INTERRUPT/
	026012	052103	042105	040440	
	026020	042116	052440	042516	
	026026	050130	040514	047111	
	026034	041101	042514	044440	
	026042	052116	051105	052522	
	026050	052120	000		

.MAIN. MACY11 30A(1052) 13-MAR-80 07:40 PAGE 5-27
 7KWL.A.P:1 13-MAR-80 07:39 ASCII MESSAGES

SEQ 0084

2539	026053	125	042516	050130	EM23:	.ASCII	/UNEXPECTED INTRPT AS (TI) WAS NOT SET/
	026060	041505	042524	020104			
	026066	047111	051124	052120			
	026074	040440	020123	052050			
	026102	024511	053440	051501			
	026110	047040	052117	051440			
	026116	052105	000				
2540	026121	124	046511	020105	EM24:	.ASCII	/TIME VALUE FAIL TO HOLD WHILE HOLD BIT/<200>
	026126	040526	052514	020105			
	026134	040506	046111	052040			
	026142	020117	047510	042114			
	026150	053440	044510	042514			
	026156	044040	046117	020104			
	026164	044502	120124				
2541	026170	040527	020123	042523		.ASCII	/WAS SET IN CSR/
	026176	020124	047111	041440			
	026204	051123	000				
2542	026207	105	051122	051117	EM25:	.ASCII	/ERROR:RDY BIT WAS NOT SUPPOSE TO SET/<200>
	026214	051072	054504	041040			
	026222	052111	053440	051501			
	026230	020040	047516	020124			
	026236	052523	050120	051517			
	026244	020105	047524	051440			
	026252	052105	000200				
2543	026256	051105	047522	035122	EM26:	.ASCII	/ERROR:PROBABLY 'M' BIT DIDN'T SET/<200>
	026264	051120	041117	041101			
	026272	054514	023440	023515			
	026300	041040	052111	042040			
	026306	042111	023516	020124			
	026314	042523	100124	000			
2544	026321	105	051122	051117	EM27:	.ASCII	/ERROR: TIME INTERVAL INTERRUPT DIDN'T OCCUR/<200>
	026326	020072	044524	042515			
	026334	044440	052116	051105			
	026342	040526	020114	047111			
	026350	042524	051122	050125			
	026356	020124	044504	047104			
	026364	052047	047440	041503			
	026372	051125	000200				
2545	026376	051105	047522	035122	EM30:	.ASCII	/ERROR: 'M' OR (DE) BIT NOT SET IN CSR/<200>
	026404	023440	023515	047440			
	026412	020122	042050	024505			
	026420	041040	052111	047040			
	026426	052117	051440	052105			
	026434	044440	020116	051503			
	026442	100122	000				
2546	026445	105	051122	051117	EM31:	.ASCII	/ERROR:'HOLD BIT FAIL TO RESET'/<200>
	026452	021072	047510	042114			
	026460	041040	052111	043040			
	026466	044501	020114	047524			
	026474	051040	051505	052105			
	026502	100042	000				
2547	026505	105	051122	051117	EM32:	.ASCII	/ERROR:'HOLD BIT FAIL TO RESET'/<200>
	026512	021072	047510	042114			
	026520	041040	052111	043040			
	026526	044501	020114	047524			
	026534	051040	051505	052105			

```

2548 026542 100042 000
      026545 110 051101 020104 EM33: .ASCIZ /HARD ERROR:'OSCILLATOR SEEMS TO BE INACCURATE'/<200>
      026552 042440 051122 051117
      026560 021072 051517 044503
      026566 046114 052101 051117
      026574 051440 042505 051515
      026602 052040 020117 042502
      026610 044440 040516 041503
      026616 051125 052101 021105
      026624 000200
2549 026626 047507 020124 047516 EM34: .ASCIZ /GOT NO INTERRUPT WITH BPU AT LEVEL 0/<200>
      026634 044440 052116 051105
      026642 052522 052120 053440
      026650 052111 020110 050102
      026656 020125 052101 046040
      026664 053105 046105 030040
      026672 000200
2550 026674 047111 042524 051122 EM35: .ASCIZ /INTERRUPT OCCUR AT WRONG LEVEL/<200>
      026702 050125 020124 041517
      026710 052503 020122 052101
      026716 053440 047522 043516
      026724 046040 053105 046105
      026732 000200
2551 026734 042047 023505 041040 EM36: .ASCIZ /'DE' BIT IN CSR FAIL TO CLEAR/<200>
      026742 052111 044440 020116
      026750 051503 020122 040506
      026756 046111 052040 020117
      026764 046103 040505 100122
      026772 000
2552 026773 200 044124 020105 DCMSG1: .ASCIZ <200>/THE FOLLOWING VALUES ARE BEING USED/<200>
      027000 047506 046114 053517
      027006 047111 020107 040526
      027014 052514 051505 040440
      027022 042522 041040 044505
      027030 043516 052440 042523
      027036 100104 000
2553 027041 200 042011 053105 DCMSG2: .ASCIZ <200>/ DEVICE ADDRESSES( /
      027046 041511 020105 042101
      027054 051104 051505 042523
      027062 024123 020040 000
2554 027067 200 050011 020103 DCMSG3: .ASCIZ <200>/ PC INTERRUPT VECTOR ADDRESS(/
      027074 047111 042524 051122
      027102 050125 020124 042526
      027110 052103 051117 040440
      027116 042104 042522 051523
      027124 000050
2555 027126 004600 050042 020103 DCMSG4: .ASCIZ <200>/ *PC INTERRUPT VECTOR ADDRESS(2)=/
      027134 047111 042524 051122
      027142 050125 020124 042526
      027150 052103 051117 040440
      027156 042104 042522 051523
      027164 031050 036451 000
2556 027171 200 041011 051525 DCMSG5: .ASCIZ <200>/ BUS REQUEST LEVEL(/
      027176 051040 050505 042525
      027204 052123 046040 053105
      027212 046105 000050

```

```

2557 027216 004600 042042 020117 DCM5G6: .ASCIZ <200>/ 'DO YOU WANT TO CHANGE ANY OF THESE VALUES (Y OR N)?'/
      027224 047531 020125 040527
      027232 052116 052040 020117
      027240 044103 047101 042507
      027246 040440 054516 047440
      027254 020106 044124 051505
      027262 020105 040526 052514
      027270 051505 024040 020131
      027276 051117 047040 037451
      027304 000042
2558 027306 021200 047105 042524 DCM5G7: .ASCIZ <200>/'ENTER VALUE AFTER (=) IS TYPED'/
      027314 020122 040526 052514
      027322 020105 043101 042524
      027330 020122 036450 020051
      027336 051511 052040 050131
      027344 042105 000042
2559 027350 036451 000 DCM5G8: .ASCIZ /)=/
2560 027353 200 044124 020105 PASM5G: .ASCIZ <200>/THE CLOCK PASSED THE RELATIVE ACCURACY TEST/<200>
      027360 046103 041517 020113
      027366 040520 051523 042105
      027374 052040 042510 051040
      027402 046105 052101 053111
      027410 020105 041501 052503
      027416 040522 054503 052040
      027424 051505 100124 000
2561 027431 105 051122 051117 DH1: .ASCII /ERROR TEST# SHOULD WAS CSR/<200>
      027436 020040 052040 051505
      027444 021524 020040 051440
      027452 047510 046125 020104
      027460 020040 020040 040527
      027466 020123 041440 051123
      027474 200
2562 027475 120 020103 020040 .ASCIZ /PC BE/
      027502 020040 020040 020040
      027510 020040 020040 042502
      027516 000
2563 027517 105 051122 051117 DH2: .ASCII /ERROR TEST# CSR /<200>
      027524 020040 052040 051505
      027532 021524 020011 041440
      027540 051123 100011
2564 027544 041520 000 .ASCIZ /PC/
2565 027547
2566 027547 200 051105 047522 DH3:
DH3: .ASCIZ <200>/ERROR PC/<200>
      027554 020122 041520 000200
2567 027562 051105 047522 020122 DH4: .ASCIZ /ERROR TEST# PRESEC CURSEC CSR/<200>
      027570 020040 042524 052123
      027576 020043 020040 051120
      027604 051505 041505 020040
      027612 041440 051125 042523
      027620 020103 041440 051123
      027626 000200
2568 027630 050040 000103 .ASCIZ /PC/
2569 027634 051105 047522 004522 DH5: .ASCIZ /ERROR TEST#/<200>
      027642 042524 052123 100043
      027650 000
2570 027651 120 000103 .ASCIZ /PC/

```

.MAIN. MACY11 30A(1052) 13-MAR-80 07:40 PAGE 5-30
CZKWA.P11 13-MAR-80 07:39 ASCII MESSAGES

SEQ 0087

2571						.EVEN
2572	027654	001116	024042	001124	DT1:	\$ERRPC,TSTNUM,GOOD,BAD,BADA,0
	027662	001126	001122	000000		
2573	027670	001116	024042	001122	DT2:	\$ERRPC,TSTNUM,BADA,0
	027676	000000				
2574	027700	001116	000000		DT3:	\$ERRPC,0
2575	027704	001116	024042	016554	DT4:	\$ERRPC,TSTNUM,PRESEC,SECOND,BADA,0
	027712	016550	001122	000000		
2576	027720	001116	024042	000000	DT5:	\$ERRPC,TSTNUM,0
2577						
2578						
2579		000001				.END

BAD = 001126	23#	459*	460*	461	468*	472*	473	479*	483*	484	526*	535*	536
	621*	624	674*	675	707	792*	793	803*	811	812*	813	822	823*
	824	834	837*	838	846	847*	849	924*	926	939*	940	993*	995
	1001*	1003	1081*	1083	1089*	1090	1205*	1206	1328*	1329	1335*	1336	1362*
	1363	1369*	1370	1494*	1496	1503*	1505	1510*	1512	2344*	2572		
BADA = 001122	25#	463*	475*	486*	528*	538*	545*	552*	627*	710	713*	741*	748*
	804*	815*	826*	840*	851*	865*	873*	920*	928*	935*	942*	970*	997*
	1005*	1056*	1067*	1085*	1096*	1145*	1150*	1172*	1318*	1331*	1338*	1352*	1365*
	1372*	1379*	1421*	1431*	1446*	1457*	1466*	1475*	1489*	1498*	1507*	1514*	1643*
	1663*	1688*	2137*	2147*	2182*	2325*	2572	2573	2575				
BASEND 011422	1182	1184	1289#										
BASIC 011050	1179#												
BASTAB 011220	1181	1213#											
BITAB 004376	451	491#											
BITEN 004436	452	489	507#										
BITO = 000001	14#	24	568	576	737	739	743	746	777	779	783	798	810
	829	1243	1272	1278	1284	2088							
BIT00 = 000001	14#												
BIT01 = 000002	14#												
BIT02 = 000004	14#												
BIT03 = 000010	14#												
BIT04 = 000020	14#												
BIT05 = 000040	14#												
BIT06 = 000100	14#												
BIT07 = 000200	14#												
BIT08 = 000400	14#	2389											
BIT09 = 001000	14#	2387	2389										
BIT1 = 000002	14#	1237	1254	1266	1762								
BIT10 = 002000	14#	2387											
BIT11 = 004000	14#	2389											
BIT12 = 010000	14#												
BIT13 = 020000	14#	2114	2166	2173	2180	2208	2387						
BIT14 = 040000	14#	541	550	807	912	1213	1249	1416	1639	1954	2076	2120	2124
	2389												
BIT15 = 100000	14#	861	869	917	932	979	983	1062	1073	1077	1092	1133	1152
	1161	1219	1225	1254	1260	1311	1345	1441	1659	1684	2120	2128	2132
	2163	2170	2177	2208	2228								
BIT2 = 000004	14#	923	938	1231	1752								
BIT3 = 000010	14#	518	568	620	697	798							
BIT4 = 000020	14#												
BIT5 = 000040	14#	656	697										
BIT6 = 000100	14#	1311	1345	1484									
BIT7 = 000200	14#	983	1077	1866	1904	2193	2208	2228					
BIT8 = 000400	14#	436	439										
BIT9 = 001000	14#												
BPTVEC = 000014	14#												
BRLV 024102	673	2305	2314*	2435#									
CHKINT 005374	650#												
CKSWR = 104407	388	444	488	512	556	564	603	613	631	688	720	736	751
	859	905	967	1053	1131	1180	1301	1410	2387	2389	2513#		
CNT 024174	1711*	1712*	1715*	1728*	2466#								
CNTLC 016106	443	487	511	555	563	602	612	630	687	719	735	750	853
	858	887	903	966	1052	1130	1179	1300	1409	1517	2047#		
CNT1 024162	2461#												
CONPA 024124	1564*	1578*	1588	1607	1612	2446#							
CONST 024054	565*	580	589*	594*	597*	1999	2030	2065	2090	2413#			

DISPLA	001142	33#	254*	2387*	2389*													
DISPLY	007022	861#	868	883	886													
DISPRE	000174	27#	254															
DMONTH	024150	1669*	2456#															
DMT	017206	521	917	932	982	1076	2163	2170	2177	2240#								
DPRI	024106	649*	652*	653	686	904	1299	1408	2439#									
DSWR =	177570	14#	33	254														
DTIM	024154	2458#																
DT1	027654	37	47	63	68	73	104	110	115	138	143	185	2572#					
DT2	027670	42	53	58	78	83	89	94	99	121	126	132	148	153				
		158	169	190	2573#													
DT3	027700	175	2574#															
DT4	027704	164	2575#															
DT5	027720	180	2576#															
ECHOB	024244	1590*	1591	2487#														
ECHOB1	024250	1572*	1573	1950*	1953	1961*	1962	2489#										
EMTVEC=	000030	14#	254*															
EM1	024406	35	2517#															
EM10	025040	71	2525#															
EM11	025152	76	2527#															
EM12	025212	81	2528#															
EM13	025250	86	2529#															
EM14	025304	92	2530#															
EM15	025437	97	2532#															
EM16	025572	102	2534#															
EM17	025605	107	2535#															
EM2	024454	40	2518#															
EM20	025655	113	2536#															
EM21	025733	118	2537#															
FM22	026004	124	2538#															
EM23	026053	129	2539#															
FM24	026121	135	2540#															
EM25	026207	141	2542#															
EM26	026256	146	2543#															
EM27	026321	151	2544#															
EM3	024502	45	2519#															
EM30	026376	156	2545#															
EM31	026445	162	2546#															
EM32	026505	167	2547#															
EM33	026545	173	2548#															
EM34	026626	178	2549#															
EM35	026674	183	2550#															
EM36	026734	188	2551#															
EM4	024563	50	2520#															
EM5	024615	56	2521#															
EM6	024651	61	2522#															
EM7	024730	66	2523#															
ERRPC	024252	2491#																
ERRVEC=	000004	14#	254*	259*	348*	2386*	2389*											
EXPC1	024202	569*	574*	578	587	593*	596*	2469#										
FLAG	024222	1710*	2477#															
FLAG.1	024210	524	863	871	915	918	933	968	991	1054	1065	1079	1094	1143				
		1148	1154	1163	1170	1192	1325	1359	1376	1419	1429	1444	1455	1464				
		1473	1565*	1641	1661	1667	1686	2117	2122	2126	2130	2135	2164	2171				
		2178	2191*	2197*	2204*	2214*	2224*	2234*	2240*	2243*	2472#							
FLAG.2	024212	260*	271	273*	296*	654*	661	695*	702	704*	1310*	1316	1324*	1344*				

L.CLK	015360	1830	1896#																	
L.CNT	015532	1909*	1912	1919	1931#															
L.END1	015542	1914	1935#																	
L.END2	015552	1921	1939#																	
L.TAB1	015534	1899	1932#																	
L.TAB2	015544	1918	1936#																	
MIN	024142	1682	1737*	2453#																
MMVEC =	000250	26#																		
MONTH	024134	1646	1734*	2450#																
NEWIM	013136	399	1558#	1581	1623															
NO.CLK	015554	1845	1947#																	
NUM	024160	1709*	1718*	1720*	1721	1724*	2460#													
NUMBER	024046	513*	531*	535	2410#															
N.CNT	015722	1947*	1957*	1964	1977#															
N.CON	015724	1960*	1978#																	
OCTMOD	002522	261	310#																	
OCTNUM	024122	1563*	1713*	1726*	1730	2445#														
PASMSG	027353	1878	1925	1970	2560#															
PCVO	024072	360	370*	2271	2560#															
PCV1	024076	365	372*	2289*	2275	2283	2285*	2292*	2293*	2427#										
PIRQ =	177772	14#				2298*	2429#													
PIRQVE =	000240	14#																		
PKBUF =	172542	20#	1863*																	
PKCNT =	172544	21#																		
KCSR =	172540	19#	1865	1866																
POINTR	014416	1706	1739#																	
PREDLT	016556	2153#																		
PRESEC	016554	2145*	2152#	2575																
PRESET	012224	1408#																		
PR1	024104	352*	371	373	2438#															
PRO =	000000	14#																		
PR1 =	000040	14#																		
PR2 =	000100	14#																		
PR3 =	000140	14#																		
PR4 =	000200	14#																		
PR5 =	000240	14#																		
PR6 =	000300	14#																		
PR7 =	000340	14#	450	611	2295	2301														
PS =	177776	14#																		
PSVO	024074	371*	2286*	2287*	2295*	2428#														
PSV1	024100	373*	2290*	2291*	2300*	2301*	2430#													
PSW =	177776	14#	450*	611*	653*	686*	904*	1299*	1408*											
PT	024240	2485#																		
PWRMSG	024006	2390	2391#																	
PWRVEC =	000024	14#	254*	2390*																
P.CLK	015220	1842	1857#																	
P.CNT	015346	1871*	1872	1884#																
P.END	015356	1874	1888#																	
P.TAB	015350	1860	1885#																	
RANDOM	010530	1130#																		
RANKTA	002566	263	329#																	
RDCMR =	104410	1570	2382	2513#																
RDLIN =	104411	391	1822	1834	2383	2513#														
RDOCT =	104412	2263	2279	2310	2513#															
RDYBIT	017000	863	871	915	968	1054	1065	1094	1136	1148	1154	1163	1192	1325						
		1359	1376	1641	1661	1667	1686	2117	2122	2126	2130	2135	2191#							

.SWRHI	8#	
.\$ACT1	9#	30
.\$APT8	9#	33#
.\$APTH	9#	32
.\$APTY	9#	2381
.\$CMTA	7#	33
.\$EOP	8#	2376
.\$ERRO	8#	2387
.\$ERRT	8#	2388
.\$POWE	7#	2390
.\$RDE	9#	
.\$RDOC	6#	2383
.\$READ	9#	2382
.\$SCOP	8#	2389
.\$SIZE	6#	2386
.\$TRAP	6#	2513
.\$TYPD	8#	2385
.\$TYPE	9#	2380
.\$TYPO	7#	2384
.\$4OCA	7#	27

. ABS. 027726 000

ERRORS DETECTED: 0

CZKWL.A.BIN,CZKWL.A.LST/CRF=CZKWL.A.P11
RUN-TIME: 66 35 4 SECONDS
RUN-TIME RATIO: 158/106=1.4
CORE USED: 28K (55 PAGES)