

DL11-W

DIAGNOSTIC
CZDLDC0

AH-8529C-MC

JUL 1978

COPYRIGHT © 76-78

digital

FICHE 1 OF 1

MADE IN USA

This microfiche card contains a grid of frames, each displaying diagnostic data. The data is organized into columns and rows, with some frames containing headers such as 'TEST', 'PASS', 'FAIL', and 'TIME'. The frames are arranged in a regular grid pattern, with the first column containing the most legible text. The data appears to be a series of test results or system status reports.

.REM @

IDENTIFICATION

PRODUCT CODE: AC-8528C-MC
PRODUCT NAME: CZDLDCO DL11-W DIAG
DATE CREATED: MARCH 1978
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DAN CASALETTO

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.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANEIS.

COPYRIGHT (C) 1976 1978 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWNG ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

1.0 GENERAL INFORMATION

1.1 ABSTRACT

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE DL11-W SERIAL LINE/ REAL TIME CLOCK INTERFACE. THE PROGRAM WILL RUN WITHOUT ANY SPECIAL TEST FIXTURES BY DEFAULT. HOWEVER, A WRAP CABLE CAN BE USED AND TESTED BY OPTION (BIT7 OF SWR).

THIS TEST OPERATES ON THE CONSOLE DL11-W SERIAL LINE AND CLOCK INTERFACES AS WELL AS UP TO FIFTEEN(15) ADDITIONAL IDENTICALLY CONFIGURED DL11-W SERIAL LINE INTERFACES. THE DEFAULT ADDRESSES ARE:

A. CONSOLE - 177560 SERIAL LINE
177546 CLOCK

B. OTHER SERIAL LINE - 776500 FIRST SERIAL LINE ADDRESS
OF 15 CONSECUTIVE SERIAL
LINE ADDRESSES

THE PROGRAM IS DESIGNED TO RUN ON ANY PDP-11 WITH 8K OF MEMORY AND A DL11-W MODULE. IT CAN BE RUN UNDER XXDP, APT, AND ACT MONITORS, AND ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER, SCFTWARE SWITCH REGISTER = LOCATION 176

POWER FAILURE IS SUPPORTED FOR SYSTEMS WITH CORE MEMORY.

NOTE: THIS DIAGNOSTIC WITH THE SWR = 000020 (CLOCK TESTS ONLY) SHOULD BE USED ON SWITCHLESS CPU'S TO TEST KW-11L LINE CLOCK MODULES.

1.2 SYSTEM REQUIREMENTS

1.2.1 EQUIPMENT

STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE AND 8K OF MEMORY.

1.2.2 STORAGE

THE PROGRAM USES MEMORY FROM 00000 TO 22372

1.3 ASSUMPTIONS

- A. IF THE UNIT UNDER TEST (UUT) IS THE CONSOLE , THE PROGRAM WILL ASSUME THE REAL TIME CLOCK (RTC) IS ENABLED AND WILL TEST IT UNLESS THE TESTS ARE DISABLED BY BIT6 OF THE SWR.
- B. IF THE UUT IS NOT THE CONSOLE, THE RTC IS NOT TESTED FOR THAT DEVICE.
- C. THE PROGRAM WILL ASSUME THE ERROR FLAG BITS AND THE BREAK FUNCTION OF THE DL11-W ARE DISABLED AND WILL NOT TEST THESE FUNCTIONS UNLESS ENABLED BY BIT10 (FOR ERROR FLAGS) AND BIT8 (FOR BREAK) OF THE SWR. THE DEFAULT CHARACTER SIZE IS 8 BITS (SEE PARA 2.3.2).

2.0 OPERATING INSTRUCTIONS

2.1 LOADING PROCEDURE

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED TAPES.

2.2 STARTING PROCEDURE

LOAD THE SWITCH REGISTER WITH SETTING
(SOFTWARE SWITCH REGISTER LOCATION = 176)

- A. START AT 200.
AFTER CHECKING THE TRANSMITTER, THE PROGRAM WILL PRINT ITS IDENTIFICATION AND REPORT THE NUMBER OF DEVICES UNDER TEST (NUMBER IS OCTAL).
"END PASS" IS PRINTED AFTER A FULL PASS HAS BEEN MADE ON ALL DEVICES UNDER TEST.
- B. START AT 204. ****NOTE****
THE "ECHO" TEST WILL BE EXECUTED. AN "*" IS PRINTED AT THE BEGINNING OF THE TEST. THE ECHO TEST READS A CHARACTER FROM THE TERMINAL, WRITES THAT CHARACTER TO THE TERMINAL AND REPORTS ANY ERROR FLAGS SET IN THE RECEIVER BUFFER.
A CONTROL-C HALTS THE TEST AND PRINTS "STOP" AT THE TERMINAL CONTINUING RESTARTS THE ECHO TEST.
- C. START AT 210. ****NOTE****
THE TERMINAL OUTPUT TEST WILL BE EXECUTED. DEPRESSING ANY CHARACTER AT THE TERMINAL, HALTS THE TEST. CONTINUING RESTARTS THE TEST.
THE TEST OUTPUTS 32 CHARACTERS ON A LINE AND REPEATS THE PATTERN EVERY THREE LINES. THE PATTERN IS AS FOLLOWS
(OCTAL CODE 040 --> 377):

!''#\$%&'()*+,-./0123456789:;<=>?	(OCTAL CODE) (040 --> 077)
@ABCDEFGHIJKLMN O PQRSTU VWXYZ	(100 --> 137)
'ABCDEFGHIJKLMN O PQRSTU VWXYZ	(140 --> 177) LOWER CASE ALPHA

THIS BOTTOM LINE COULD BE THE FOLLOWING IF THE TERMINAL
DOES NOT HAVE LOWER CASE:

@ABCDEFGHIJKLMNOPQRSTUVWXYZ

UPPER CASE ALPHA

****NOTE****

IF THE TESTING ON TERMINALS OTHER THAN THE CONSOLE IS DESIRED
FOR TESTS B OR C,SEE SECTION 2.3.4. AND 2.3.5. OF THIS DOCUMENT.

::++C
::++C

2.3 OPERATING PROCEDURE

2.3.1 OPERATIONAL SWITCH SETTINGS

NOTE: IF NO HARDWARE SWITCH REGISTER IS AVAILABLE THE PROGRAM WILL AUTOMATICALLY USE THE CONTENTS OF LOCATION 176 AS THE SOFTWARE SWITCH REGISTER. THE USER SHOULD SET THIS LOCATION BEFORE STARTING THE PROGRAM. IF A HARDWARE SWITCH IS AVAILABLE AND A SOFTWARE SWR IS DESIRED, PUT ALL SWITCHES UP.

BIT15	- HALT ON ERROR
BIT14	- SCOPE LOOP
BIT13	- INHIBIT ERROR TYPEOUT
BIT12	- UNUSED
BIT11	- UNUSED
BIT10	- ENABLE ERROR FLAGS TESTS
BIT09	- LOOP ON ERROR
BIT08	- ENABLE BREAK FUNCTION TESTS
BIT07	- ENABLE DATA TEST WITH WRAP CABLE
BIT06	- INHIBIT RTC TESTS (ALLOW ONLY SLU TESTS)
BIT05	- ALLOW MANUAL SETTING OF '\$DEVN' (DEVICE MAP)
BIT04	- INHIBIT SLU TESTS (ALLOW ONLY LINE CLOCK TESTS)

BUILT INTO THE PROGRAM IS THE ABILITY TO DYNAMICALLY CHANGE THE CONTENTS OF SWREG DURING PROGRAM EXECUTION. BY STRIKING G (CNTRL G) ON CONSOLE TTY THE OPERATOR SETS A REQUEST FLAG TO CHANGE THE CONTENTS OF SWREG, WHICH IS PROCESSED IN KEY AREAS OF THE PROGRAM CODE (IE) ERROR ROUTINES, AFTER HALTS END OF PASS, AND OTHER APPLICABLE AREAS. BECAUSE THIS DIAGNOSTIC USES THE MAINTENANCE BIT OF THE SERIAL LINE, THE CONTROL-G SHOULD BE ISSUED DURING PROGRAM TYPEOUTS AT THAT TIME THE MAINTENANCE BIT IS SURE TO BE CLEAR.

IF A CONTROL-G IS DETECTED, UPON EXECUTION THE CONTENTS OF THE SWREG ARE DUMPED IN OCTAL ON THE CONSOLE TTY AND ANY CHANGES ARE REQUESTED

(IE) SWR=XXXXXX NEW=

POSSIBLE RESPONSES ARE:

1. <CR> IF NO CHANGES ARE TO BE MADE
2. 6 DIGITS 0-7 TO REPRESENT IN OCTAL THE NEW SWITCH REGISTER VALUE ;LAST DIGIT FOLLOWED BY <CR>.
3. U TO ALLOW REENTERING VALUE IF ERROR IS COMMITTED KEYING IN SWREG VALUE.

2.3.2 SETTING BITS PER CHARACTER

THIS PROGRAM DEFAULTS TO TESTING 8 BITS PER CHARACTER. IF THE SERIAL LINE IS SET FOR 5-->7 BITS PER CHARACTER, SET THE MEMORY LOCATION '\$USWR' AS FOLLOWS:

CHAR. SIZE (# OF BITS)	'\$USWR' CONTENTS (BINARY)	(OCTAL)
8	100000000	400
7	010000000	200
6	001000000	100
5	000100000	40

'\$USWR' IS USED IN THE DATA PATH TESTS TO LIMIT THE BINARY COUNT TEST PATTERN TO THE NUMBER OF BITS SELECTED ON THE SERIAL LINE.

2.3.3 RUNNING UNDER APT

THE APT MAILBOX IS LOCATED AT LOCATION 500, TO ALLOW ADDITIONAL SERIAL LINE VECTOR ASSIGNMENTS TO THE 400 AREA OF MEMORY.

THE EXECUTION TIMES PROVIDED ARE FOR EXECUTION WITH AN 11/34 PROCESSOR, CORE MEMORY, AND 110 BAUD.

2.3.4 RUN WITH ALTERNATE CONSOLE ADDRESS

TO USE A CONSOLE ADDRESS OTHER THAN 177560, OR VECTOR OTHER THAN 60, THE OPERATOR MUST SUPPLY THE PROGRAM WITH THE CORRECT ADDRESSES BY INSERTING THEM AT THE TAG LABELED "CRCSR":

CRCSR: ADDRESS OF RECEIVER RCSR
CRBUF: ADDRESS OF RECEIVER BUFFER
CTCSR: ADDRESS OF TRANSMITTER CSR
CTBUF: ADDRESS OF TRANSMITTER BUFFER
CRVECT: ADDRESS OF RECEIVER VECTOR
CRPSW: ADDRESS OF ASSOCIATED PSW
CTVECT: ADDRESS OF TRANSMITTER VECTOR
CTPSW: ADDRESS OF ASSOCIATED PSW

2.3.5 TESTING ADDITIONAL SERIAL LINES

THIS PROGRAM WILL SUPPORT TESTING OF MULTIPLE SLU'S. IT REQUIRES THE ADDRESS OF THE FIRST ADDITIONAL RCSR (STORED AT "\$BASE") AND ITS INTERRUPT VECTOR (STORED AT "\$VECT1"); AND WILL BE ABLE TO ADDRESS ANY SLU STARTING AT THE SPECIFIED BASE ADDRESS UP TO 15 CONSECUTIVE DEVICES.

EXAMPLE: \$BASE: 776500
 \$VECT1: 300
THE PROGRAM WILL BE ABLE TO TEST THE CONSOLE PLUS ANY ADDITIONAL DL11-W SLU'S WITHIN THE RANGE 776500 --> 776660

\$BASE AND \$VECT1 DEFAULT TO 776500 AND 300 RESPECTIVELY.

THE PROGRAM ASSOCIATES UNIT NUMBERS TO DEVICES AS FOLLOWS:
(NUMBERS IN PARENTHESIS ARE OCTAL)

UNIT# 0 --> CONSOLE ADDRESS STORED AT "CRCSR"
UNIT# 1 --> BASE ADDRESS STORED AT "\$BASE"
 ASSOCIATED BASE VECTOR STORED AT "\$VECT1"
UNIT# 2 --> BASE ADDRESS + (10)
 BASE VECTOR + (10)
UNIT# 3 --> BASE ADDRESS + (20)
 BASE VECTOR + (20)
UNIT# 4 --> BASE ADDRESS + (30)
 BASE VECTOR + (30)
 .
 .
 .
 V
UNIT# 5 --> BASE ADDRESS + (160)
 BASE VECTOR + (160)

STARTING AT LOCATION 200 AND HAVING BITS OF SWR CLEAR, THE PROGRAM WILL SELF
SIZE THE NUMBER OF DEVICES (STARTING AT THE BAS ADDRESS) AND
STORE A BIT MAP AT "\$DEV" (DEVICE MAP) TO INDICATE WHICH UNIT NUMBERS
ARE PRESENT AND WILL BE TESTED:

```
-----  
: UNIT ! UNIT ! .....! UNIT ! UNIT ! CONSOLE!  
: 15 ! 14 ! .....! 2 ! 1 !  
:-----
```

A BIT MAP CAN BE ENTERED AT "\$DEV" PRIOR TO STARTING THE PROGRAM
SETTING BITS OF THE SWR INHIBITS THE SELF-SIZING AND DEVICE MAP
GENERATION, AND USES THE VALUE STORED BY THE OPERATOR.

EXAMPLE:

SWR = 000040 BINARY 0 000 000 000 100 000
\$BASE: 776500
\$VECT1: 300

\$DEV: 13 BINARY - 0 000 000 000 001 011

THE PROGRAM WILL TEST -
UNIT# 0 = CONSOLE
UNIT# 1 = 776500 ; 300
UNIT# 3 = 776520 ; 320

2.4 EXECUTION TIMES - (110 BAUD)

LONGEST SUBTEST TIME = 50 SECONDS
PASS TIME = 60 SECONDS
ADDITIONAL DEVICES = 55 SECONDS/DEVICE

3.0 ERROR REPORTING

IF A ROUTINE FAILS AND THE INHIBIT ERROR TYPEOUT (BIT13) OF THE SWR
IS NOT SET, A PRINTOUT RESULTS IN THE FORM:

“(SOME ASCII MESSAGE)”
TEST# ERR PC RCSR ANY APPLICABLE DAT HEADINGS
XXXXXX XXXXXX XXXXXX ANY APPLICABLE DATA

NOTE: "RCSR" IS DEPENDENT ON THE FAILURE
& THEREFORE COULD BE TCSR,RBUF,TBUF,OR LKS

WHERE "XXXXXX" IS AN OCTAL NUMBER.
THIS ERROR PRINTOUT OCCURS PROVIDED THE ERROR THAT EXISTS
WOULD NOT HINDER THE TYPEOUT. IN CASES WHERE IT IS NOT POSSIBLE
TO PRINT AN ERROR MESSAGE (I.E. FATAL CONSOLE TRANSMITTER
FAILURES), A HALT OCCURS AND THE PC CAN BE EXAMINED BY THE OPERATOR
TO FIND THE ERROR INFORMATION IN THE PROGRAM LISTING.

NOTE: FOR SOFTWARE SWITCH OPERATION, THE SWITCH REGISTER CAN
BE CHANGED BY TYPING A CONTROL-G AT THE CONSOLE DURING ERROR PRINTOUTS.
AFTER CONTINUING FROM THE ERROR HALT THE OLD SWR CONTENTS
IS DISPLAYED AND THE NEW CONTENTS CAN BE ENTERED.
IF ERROR HALTS ARE DISABLED, THE CONTROL-G RESPONSE OCCURS
IMMEDIATELY FOLLOWING THE TYPEOUT.

4.0 SUBROUTINE ABSTRACTS

4.1 TRAPCATCHER

THIS IS A SERIES OF INSTRUCTIONS STARTING AT LOCATION 0 DESIGNED TO DETECT AND ISOLATE UNEXPECTED TRAPS TO THE TRAP AND INTERRUPT VECTOR AREA OF MEMORY.

EACH VECTOR ENTRANCE ADDRESS IS LOADED WITH THE ADDRESS OF THE NEXT LOCATION. THE NEXT LOCATION IS LOADED WITH A BREAK POINT TRAP (000003). THUS AN ILLEGAL TRAP OR INTERRUPT CAUSES A TRAP THROUGH THE BPT VECTOR(14) WHICH POINTS TO THE "CATCH" ROUTINE.

THE "CATCH" ROUTINE REPORTS THE PC THAT CAUSED THE ORIGINAL TRAP AND THE LOCATION OF THE TRAP VECTOR (IF UNDER APT, AN ERROR IS INDICATED TO APT). AFTER REPORTING THE ERROR THE PROGRAM HALTS. THE PROGRAM MUST BE RESTARTED AT THIS POINT.

4.2 WRPSW

THIS ROUTINE IS USED TO WRITE THE PSW BY POPPING VALUES FROM THE STACK. THIS METHOD IS USED TO BE COMPATIBLE WITH ALL 11 FAMILY PROCESSORS.

4.3 SCOPE

THIS ROUTINE CALL IS PLACED BETWEEN EACH SUBTEST. IT RECORDS THE STARTING ADDRESS OF EACH SUBTEST AS IT IS BEING ENTERED AND UPDATES THE TEST NUMBER. IF A SCOPE LOOP IS REQUESTED, IT WILL JUMP TO THE START OF THE SUBTEST AT WHICH THE SCOPE LOOP IS REQUESTED.

4.4 ERROR

THIS ROUTINE CALL IS PLACED WHEREEVER AN ERROR REPORT IS DESIRED. THE LOWER BYTE OF THIS CALL IS USED AS THE ERROR NUMBER AND AS A POINTER INTO THE ERROR TABLE. THIS ROUTINE REPORTS ERRORS TO APT USING '\$APTYPE' AND TYPES ERROR REPORTS TO THE CONSOLE USING '\$ERRTYPE'.

4.5 \$POWER

THIS ROUTINE SAVES ALL GENERAL REGISTERS DURING POWER-DOWN AND RESTORES THEM AT POWER-UP. IF A POWER FAILURE OCCURS 'POWER' IS PRINTED AT THE CONSOLE AFTER POWER IS RESTORED.

4.6 CKSWR

THIS ROUTINE CALL IS USED TO DETECT THE RECEPTION OF A CONTROL-G FROM THE CONSOLE. THE CALL USES '\$READ' TO PERFORM THE CONTROL-G SEQUENCE OF DISPLAYING THE CONTENTS OF THE SOFTWARE SWITCH REGISTER AND THE ENTERING THE NEW CONTENTS FROM THE TERMINAL.

%

```
422
423
424
425
426
427      .SBTTL  BASIC DEFINITIONS
428
429      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
430      001100  STACK= 1100
431      .EQUIV  EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
432      .EQUIV  IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL
433
434      ;*MISCELLANEOUS DEFINITIONS
435      000011  HT= 11          ;;CODE FOR HORIZONTAL TAB
436      000012  LF= 12          ;;CODE FOR LINE FEED
437      000015  CR= 15          ;;CODE FOR CARRIAGE RETURN
438      000200  CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
439      177776  PS= 177776     ;;PROCESSOR STATUS WORD
440      .EQUIV  PS,PSW
441      177774  STKLMT= 177774  ;;STACK LIMIT REGISTER
442      177772  PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
443      177570  DSWR= 177570   ;;HARDWARE SWITCH REGISTER
444      177570  DDISP= 177570  ;;HARDWARE DISPLAY REGISTER
445
446      ;*GENERAL PURPOSE REGISTER DEFINITIONS
447      000000  R0= X0          ;;GENERAL REGISTER
448      000001  R1= X1          ;;GENERAL REGISTER
449      000002  R2= X2          ;;GENERAL REGISTER
450      000003  R3= X3          ;;GENERAL REGISTER
451      000004  R4= X4          ;;GENERAL REGISTER
452      000005  R5= X5          ;;GENERAL REGISTER
453      000006  R6= X6          ;;GENERAL REGISTER
454      000007  R7= X7          ;;GENERAL REGISTER
455      000006  SP= X6          ;;STACK POINTER
456      000007  PC= X7          ;;PROGRAM COUNTER
457
458      ;*PRIORITY LEVEL DEFINITIONS
459      000000  PR0= 0          ;;PRIORITY LEVEL 0
460      000040  PR1= 40         ;;PRIORITY LEVEL 1
461      000100  PR2= 100        ;;PRIORITY LEVEL 2
462      000140  PR3= 140        ;;PRIORITY LEVEL 3
463      000200  PR4= 200        ;;PRIORITY LEVEL 4
464      000240  PR5= 240        ;;PRIORITY LEVEL 5
465      000300  PR6= 300        ;;PRIORITY LEVEL 6
466      000340  PR7= 340        ;;PRIORITY LEVEL 7
467
468      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
469      100000  SW15= 100000
470      040000  SW14= 40000
471      020000  SW13= 20000
472      010000  SW12= 10000
473      004000  SW11= 4000
474      002000  SW10= 2000
475      001000  SW09= 1000
476      000400  SW08= 400
477      000200  SW07= 200
```

BASIC DEFINITIONS

```
478      000100      SW06= 100
479      000040      SW05=  40
480      000020      SW04=  20
481      000010      SW03=  10
482      000004      SW02=   4
483      000002      SW01=   2
484      000001      SW00=   1
485      .EQUIV SW09,SW9
486      .EQUIV SW08,SW8
487      .EQUIV SW07,SW7
488      .EQUIV SW06,SW6
489      .EQUIV SW05,SW5
490      .EQUIV SW04,SW4
491      .EQUIV SW03,SW3
492      .EQUIV SW02,SW2
493      .EQUIV SW01,SW1
494      .EQUIV SW00,SW0
```

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

```
496      .EQUIV SW09,SW9
497      100000      BIT15= 100000
498      040000      BIT14=  40000
499      020000      BIT13=  20000
500      010000      BIT12=  10000
501      004000      BIT11=   4000
502      002000      BIT10=   2000
503      001000      BIT09=   1000
504      000400      BIT08=   400
505      000200      BIT07=   200
506      000100      BIT06=   100
507      000040      BIT05=   40
508      000020      BIT04=   20
509      000010      BIT03=   10
510      000004      BIT02=   4
511      000002      BIT01=   2
512      000001      BIT00=   1
513      .EQUIV BIT09,BIT9
514      .EQUIV BIT08,BIT8
515      .EQUIV BIT07,BIT7
516      .EQUIV BIT06,BIT6
517      .EQUIV BIT05,BIT5
518      .EQUIV BIT04,BIT4
519      .EQUIV BIT03,BIT3
520      .EQUIV BIT02,BIT2
521      .EQUIV BIT01,BIT1
522      .EQUIV BIT00,BIT0
```

;*BASIC "CPU" TRAP VECTOR ADDRESSES

```
524      .EQUIV BIT09,BIT9
525      000004      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
526      000010      RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
527      000014      TBITVEC=14        ;; "T" BIT
528      000014      TRTVEC= 14         ;; TRACE TRAP
529      000014      BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
530      000020      IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
531      000024      PWRVEC= 24         ;; POWER FAIL
532      000030      EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
533      000034      TRAPVEC=34        ;; "TRAP" TRAP
```

BASIC DEFINITIONS

```
534      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
535      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
536      000240      PIRQVEC=240    ;;PROGRAM INTERRUPT REQUEST VECTOR
537      176500      ABASE= 176500
538      000300      AVECT1= 300
539      000400      AUSWR= 400
540      000001      $TN= 1
541      161000      $SWR= 161000
542      000003      BPT= 000003      ;THIS IS THE COMMAND FOR A TRAP
543                                     ; THROUGH 14 (BPT TRAP)
544
545      000000      .=0
546      ;;*****
547      ;*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ".+2,BP;"
548      ;*SEQUENCE TO CATCH ILLEGAL TRAPS & INTERRUPTS
549      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
550
551
552      000014      .=14      ;THE BPT TRAP VECTOR POINTS TO THE
553 000014 012254  .WORD  CATCH  ; ILLEGAL TRAP HANDLER "CATCH"
554 000016 000340  .WORD  340
555
556      000042      .= 42
557 000042 000000  .WORD  0
558
559
560
561
562      000174      .= 174
563 000174 000000  DISPREG: .WORD 0
564 000176 000000  SWREG:  .WORD 0
565
566      000200      .=200
567  J0200 000137 002546  JMP  START      ;DO INTERFACE TEST
568 000204 000137 014762  JMP  ECHO       ;DO ECHO TEST
569 000210 000137 015202  JMP  OUTTST    ;DO OUTPUT TEST TO TERMINAL
```

BASIC DEFINITIONS

```
570
571      000500
572      .SBTTL      .=      500
573      .SBTTL      ACT11 HOOKS
574      ;:*****
575      ;HOOKS REQUIRED BY ACT11
576      000500      $SVPC=.      ;SAVE PC
577      000046      .=46
578      000046      012152      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
579      000052      000052      .=52
580      000052      000000      .WORD      0      ;;2)SET LOC.52 TO ZERO
581      000500      .= $SVPC      ;; RESTORE PC
582      .SBTTL      APT PARAMETER BLOCK
583
584      ;:*****
585      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
586      ;:*****
587      000500      .$X=.      ;;SAVE CURRENT LOCATION
588      000024      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
589      000024      000200      200      ;;FOR APT START UP
590      000044      000044      .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
591      000044      000500      $APTHDR      ;;POINT TO APT HEADER BLOCK
592      000500      .=.$X      ;;RESET LOCATION COUNTER
593      ;:*****
594      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
595      ;INTERFACE SPEC.
596
597      000500      $APTHD:
598      000500      000000      $HIBTS: .WORD      0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
599      000502      001066      $MBADR: .WORD      $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
600      000504      000050      $TSTM: .WORD      50      ;;RUN TIM OF LONGEST TEST
601      000506      000060      $PASTM: .WORD      60      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
602      000510      000055      $UNITM: .WORD      55      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
603      000512      000030      .WORD      $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
604
```


COMMON TAGS

```
605          .SBTTL COMMON TAGS
606
607          ;:*****
608          ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
609          ;*USED IN THE PROGRAM.
610
611          001000          .=1000
612 001000          $CMTAG:          ;;START OF COMMON TAGS
613 001000 000000          .WORD 0          ;;CONTAINS THE TEST NUMBER
614 001002 000          $TSTNM: .BYTE 0          ;;CONTAINS ERROR FLAG
615 001003 000          $ERFLG: .BYTE 0          ;;CONTAINS SUBTEST ITERATION COUNT
616 001004 000000          $ICNT: .WORD 0          ;;CONTAINS SCOPE LOOP ADDRESS
617 001006 000000          $LPADR: .WORD 0          ;;CONTAINS SCOPE RETURN FOR ERRORS
618 001010 000000          $LPERR: .WORD 0          ;;CONTAINS TOTAL ERRORS DETECTED
619 001012 000000          $ERTTL: .WORD 0          ;;CONTAINS ITEM CONTROL BYTE
620 001014 000          $ITEMB: .BYTE 0          ;;CONTAINS MAX. ERRORS PER TEST
621 001015 001          $ERMAX: .BYTE 1          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
622 001016 000000          $ERRPC: .WORD 0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
623 001020 000000          $GDADR: .WORD 0          ;;CONTAINS ADDRESS OF 'BAD' DATA
624 001022 000000          $BDADR: .WORD 0          ;;CONTAINS 'GOOD' DATA
625 001024 000000          $GDDAT: .WORD 0          ;;CONTAINS 'BAD' DATA
626 001026 000000          $BDDAT: .WORD 0          ;;RESERVED--NOT TO BE USED
627 001030 000000          .WORD 0
628 001032 000000          .WORD 0
629 001034 000          $AUTOB: .BYTE 0          ;;AUTOMATIC MODE INDICATOR
630 001035 000          $INTAG: .BYTE 0          ;;INTERRUPT MODE INDICATOR
631 001036 000000          .WORD 0
632 001040 177570          SWR: .WORD DSWR          ;;ADDRESS OF SWITCH REGISTER
633 001042 177570          DISPLAY: .WORD DDISP          ;;ADDRESS OF DISPLAY REGISTER
634 001044 177560          $TKS: 177560          ;;TTY KBD STATUS
635 001046 177562          $TKB: 177562          ;;TTY KBD BUFFER
636 001050 177564          $TPS: 177564          ;;TTY PRINTER STATUS REG. ADDRESS
637 001052 177566          $TPB: 177566          ;;TTY PRINTER BUFFER REG. ADDRESS
638 001054 000          $NULL: .BYTE 0          ;;CONTAINS NULL CHARACTER FOR FILLS
639 001055 002          $FILLS: .BYTE 2          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
640 001056 012          $FILLC: .BYTE 12          ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
641 001057 000          $TPFLG: .BYTE 0          ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
642 001060 000000          $ESCAPE: 0          ;;ESCAPE ON ERROR ADDRESS
643 001062 077          $QUES: .ASCII /?/          ;;QUESTION MARK
644 001063 015          $CRLF: .ASCII <15>          ;;CARRIAGE RETURN
645 001064 000012          $LF: .ASCIIZ <12>          ;;LINE FEED
646          ;:*****
647          .SBTTL APT MAILBOX-ETABLE
648
649          ;:*****
650          .EVEN
651 001066          $MAIL:          ;;APT MAILBOX
652 001066 000000          $MSGTY: .WORD AMSGTY          ;;MESSAGE TYPE CODE
653 001070 000000          $FATAL: .WORD AFATAL          ;;FATAL ERROR NUMBER
654 001072 000000          $TESTN: .WORD ATESTN          ;;TEST NUMBER
655 001074 000000          $PASS: .WORD APASS          ;;PASS COUNT
656 001076 000000          $DEVCT: .WORD ADEVCT          ;;DEVICE COUNT
657 001100 000000          $UNIT: .WORD AUNIT          ;;I/O UNIT NUMBER
658 001102 000000          $MSGAD: .WORD AMSGAD          ;;MESSAGE ADDRESS
659 001104 000000          $MSGLG: .WORD AMSGLG          ;;MESSAGE LENGTH
660 001106          $ETABLE:          ;;APT ENVIRONMENT TABLE
```

```
661 001106 000 $ENV: .BYTE AENV ;;ENVIRONMENT BYTE
662 001107 000 $ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
663 001110 000000 $SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
664 001112 000400 $USWR: .WORD AUSWR ;;USER SWITCHES
665 001114 000000 $CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
666 : * BITS 15-11=CPU TYPE
667 : * 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
668 : * 1 /70=06,PDQ=07,Q=10
669 : * BIT 10=REAL TIME CLOCK
670 : * BIT 9=FLOATING POINT PROCESSOR
671 : * BIT 8=MEMORY MANAGEMENT
672 001116 000 $MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
673 001117 000 $MTYP1: .BYTE AMTYP1 ;;MEM. TYPE,BLK#1
674 : * MEM.TYPE BYTE -- (HIGH BYTE)
675 : * 900 NSEC CORE=001
676 : * 300 NSEC BIPOLAR=002
677 : * 500 NSEC MOS=003
678 001120 000000 $MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
679 : * MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
680 001122 000 $MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
681 001123 000 $MTYP2: .BYTE AMTYP2 ;;MEM.TYPE,BLK#2
682 001124 000000 $MADR2: .WORD AMADR2 ;;MEM.LAST ADDRESS,BLK#2
683 001126 000 $MAMS3: .BYTE AMAMS3 ;;HIGH ADDRESS,M.S.BYTE
684 001127 000 $MTYP3: .BYTE AMTYP3 ;;MEM.TYPE,BLK#3
685 001130 000000 $MADR3: .WORD AMADR3 ;;MEM.LAST ADDRESS,BLK#3
686 001132 000 $MAMS4: .BYTE AMAMS4 ;;HIGH ADDRESS,M.S.BYTE
687 001133 000 $MTYP4: .BYTE AMTYP4 ;;MEM.TYPE,BLK#4
688 001134 000000 $MADR4: .WORD AMADR4 ;;MEM.LAST ADDRESS,BLK#4
689 001136 000300 $VECT1: .WORD AVECT1 ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
690 001140 000000 $VECT2: .WORD AVECT2 ;;INTERRUPT VECTOR#2BUS PRIORITY#2
691 001142 176500 $BASE: .WORD ABASE ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
692 001144 000000 $DEVN: .WORD ADEVN ;;DEVICE MAP
693 001146
694 .MEXIT
```

ERROR POINTER TABLE

```
695      .SBTTL  ERROR POINTER TABLE
696
697      ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
698      ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
699      ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
700      ;*NOTE1:      IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
701      ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
702
703      ;*      EM      ;;POINTS TO THE ERROR MESSAGE
704      ;*      DH      ;;POINTS TO THE DATA HEADER
705      ;*      DT      ;;POINTS TO THE DATA
706      ;*      DF      ;;POINTS TO THE DATA FORMAT
707
708
709      001146      $ERRTB:
710
711      001146      . $ERRTB:
712      001146      015266      EM1      ;"CAN NOT ACCESS TCSR"
713      001146      021451      DH1      ;"TEST# ERR PC  TCSR"
714      001152      022204      DT1      ;$TESTN,$ERRPC,TCSR
715      001154      000000      0
716
717      001156      015312      EM2      ;"CAN NOT ACCESS TBUF"
718      001160      021476      DH2      ;"TEST# ERR PC  TBUF"
719      001162      022214      DT2      ;$TESTN,$ERRPC,TBUF
720      001164      000000      0
721
722      001166      015336      EM3      ;"TCSR DONE NOT CLEARED WITH TBUF FULL"
723      001170      021451      DH1      ;"TEST# ERR PC  TCSR"
724      001172      022204      DT1      ;$TESTN,$ERRPC,TCSR
725      001174      000000      0
726
727      001176      015403      EM4      ;"TCSR DONE NOT SET"
728      001200      021451      DH1      ;"TEST# ERR PC  TCSR"
729      001202      022204      DT1      ;$TESTN,$ERRPC,TCSR
730      001204      000000      0
731
732      001206      015425      EM5      ;TCSR DONE NOT SET WITH RESET
733      001210      021451      DH1      ;"TEST# ERR PC  TCSR"
734      001212      022204      DT1      ;$TESTN,$ERRPC,TCSR
735      001214      000000      0
736
737      001216      015462      EM6      ;"CAN NOT ACCESS RCSR"
738      001220      021523      DH6      ;"TEST# ERR PC  RCSR"
739      001222      022224      DT6      ;$TESTN,$ERRPC,RCSR
740      001224      000000      0
741
742      001226      015506      EM7      ;"CAN NOT ACCESS RBUF"
743      001230      021550      DH7      ;"TEST# ERR PC  RBUF"
744      001232      022234      DT7      ;$TESTN,$ERRPC,RBUF
745      001234      000000      0
746
747      001236      015532      EM10     ;"CAN NOT ACCESS LKS"
748      001240      021575      DH10     ;"TEST# ERR PC  LKS"
749      001242      022244      DT10     ;$TESTN,$ERRPC,LKS
750      001244      000000      0
```

751				
752	001246	015555	EM11	;'BIT0 OF TCSR NOT CLEAR AFTER RESET''
753	001250	021451	DH1	;'TEST# ERR PC TCSR''
754	001252	022204	DT1	;\$TESTN,\$ERRPC,TCSR
755	001254	000000	0	
756				
757	001256	015620	EM12	;'CAN NOT SET BIT0 OF TCSR''
758	001260	021451	DH1	;'TEST# ERR PC TCSR''
759	001262	022204	DT1	;\$TESTN,\$ERRPC,TCSR
760	001264	000000	0	
761				
762	001266	015651	EM13	;'CAN NOT CLEAR BIT0 OF TCSR''
763	001270	021451	DH1	;'TEST# ERR PC TCSR''
764	001272	022204	DT1	;\$TESTN,\$ERRPC,TCSR
765	001274	000000	0	
766				
767	001276	015704	EM14	;'RESET DID NOT CLEAR BIT0 OF TCSR''
768	001300	021451	DH1	;'TEST# ERR PC TCSR''
769	001302	022204	DT1	;\$TESTN,\$ERRPC,TCSR
770	001304	000000	0	
771				
772	001306	015745	EM15	;'BIT2 OF TCSR NOT CLEAR AFTER RESET''
773	001310	021451	DH1	;'TEST# ERR PC TCSR''
774	001312	022204	DT1	;\$TESTN,\$ERRPC,TCSR
775	001314	000000	0	
776				
777	001316	016010	EM16	;'CAN NOT SET BIT2 OF TCSR''
778	001320	021451	DH1	;'TEST# ERR PC TCSR''
779	001322	022204	DT1	;\$TESTN,\$ERRPC,TCSR
780	001324	000000	0	
781				
782	001326	016041	EM17	;'CAN NOT CLEAR BIT2 OF TCSR''
783	001330	021451	DH1	;'TEST# ERR PC TCSR''
784	001332	022204	DT1	;\$TESTN,\$ERRPC,TCSR
785	001334	000000	0	
786				
787	001336	016074	EM20	;'RESET DID NOT CLEAR BIT2 OF TCSR''
788	001340	021451	DH1	;'TEST# ERR PC TCSR''
789	001342	022204	DT1	;\$TESTN,\$ERRPC,TCSR
790	001344	000000	0	
791				
792	001346	016135	EM21	;'BIT6 OF TCSR NOT CLEAR AFTER RESET2
793	001350	021451	DH1	;'TEST# ERR PC TCSR''
794	001352	022204	DT1	;\$TESTN,\$ERRPC,TCSR
795	001354	000000	0	
796				
797	001356	016201	EM22	;'XMIT INTERRUPT WITH PRIORITY 7''
798	001360	021451	DH1	;'TEST# ERR PC TCSR''
799	001362	022204	DT1	;\$TESTN,\$ERRPC,TCSR
800	001364	000000	0	
801				
802	001366	016236	EM23	;'CAN NOT SET BIT6 OF TCSR''
803	001370	021451	DH1	;'TEST# ERR PC TCSR''
804	001372	022204	DT1	;\$TESTN,\$ERRPC,TCSR
805	001374	000000	0	
806				

807	001376	016267	EM24	;'CAN NOT CLEAR BIT6 OF TCSR'
808	001400	021451	DH1	;'TEST# ERR PC TCSR'
809	001402	022204	DT1	;\$TESTN,\$ERRPC,TCSR
810	001404	000000	0	
811				
812	001406	016322	EM25	;'RESET DID NOT CLEAR BIT6 OF TCSR'
813	001410	021451	DH1	;'TEST# ERR PC TCSR'
814	001412	022204	DT1	;\$TESTN,\$ERRPC,TCSR
815	001414	000000	0	
816				
817	001416	016363	EM26	;'BIT6 OF RCSR NOT CLEAR AFTER RESET'
818	001420	021523	DH6	;'TEST# ERR PC RCSR'
819	001422	022224	DT6	;\$TESTN,\$ERRPC,RCSR
820	001424	000000	0	
821				
822	001426	016426	EM27	;'RCVR INTERRUPT WITH PRIORITY 7'
823	001430	021523	DH6	;'TEST# ERR PC RCSR'
824	001432	022224	DT6	;\$TESTN,\$ERRPC,RCSR
825	001434	000000	0	
826				
827	001436	016465	EM30	;'CAN NOT SET BIT6 OF RCSR'
828	001440	021523	DH6	;'TEST# ERR PC RCSR'
829	001442	022224	DT6	;\$TESTN,\$ERRPC,RCSR
830	001444	000000	0	
831				
832	001446	016516	EM31	;'CAN NOT CLEAR BIT6 OF RCSR'
833	001450	021523	DH6	;'TEST# ERR PC RCSR'
834	001452	022224	DT6	;\$TESTN,\$ERRPC,RCSR
835	001454	000000	0	
836				
837	001456	016551	EM32	;'CAN NOT CLEAR BIT6 OF RCSR WITH RESET2
838	001460	021523	DH6	;'TEST# ERR PC RCSR'
839	001462	022224	DT6	;\$TESTN,\$ERRPC,RCSR
840	001464	000000	0	
841				
842	001466	016617	EM33	;'BIT6 OF LKS NOT CLEAR AFTER RESET'
843	001470	021575	DH10	;'TEST# ERR PC LKS'
844	001472	022244	DT10	;\$TESTN,\$ERRPC,LKS
845	001474	000000	0	
846				
847	001476	016661	EM34	;'LKS INTERRUPT WITH PRIORITY 7'
848	001500	021575	DH10	;'TEST# ERR PC LKS'
849	001502	022244	DT10	;\$TESTN,\$ERRPC,LKS
850	001504	000000	0	
851				
852	001506	016717	EM35	;'CAN NOT SET BIT6 OF LKS'
853	001510	021575	DH10	;'TEST# ERR PC LKS'
854	001512	022244	DT10	;\$TESTN,\$ERRPC,LKS
855	001514	000000	0	
856				
857	001516	016747	EM36	;'CAN NOT CLEAR BIT6 OF LKS'
858	001520	021575	DH10	;'TEST# ERR PC LKS'
859	001522	022244	DT10	;\$TESTN,\$ERRPC,LKS
860	001524	000000	0	
861				
862	001526	017001	EM37	;'RESET DID NOT CLEAR BIT6 OF LKS'

863	001530	021575	DH10	:"TEST# ERR PC LKS"
864	001532	022244	DT10	:\$TESTN,\$ERRPC,LKS
865	001534	000000	0	
866				
867	001536	017041	EM40	:"DUAL ADDRESSING ERROR"
868	001540	021621	DH40	:"TEST# ERR PC GOOD BAD"
869	001542	022254	DT40	:\$TESTN,\$ERRPC,\$GDADR,\$BDCSR
870	001544	000000	0	
871				
872	001546	017067	EM41	:"BIT7 OF LKS NOT SET AFTER RESET2"
873	001550	021575	DH10	:"TEST# ERR PC LKS"
874	001552	022244	DT10	:\$TESTN,\$ERRPC,LKS
875	001554	000000	0	
876				
877	001556	017127	EM42	:"CAN NOT CLEAR BIT7 OF LKS"
878	001560	021575	DH10	:"TEST# ERR PC LKS"
879	001562	022244	DT10	:\$TESTN,\$ERRPC,LKS
880	001564	000000	0	
881				
882	001566	017161	EM43	:"BIT7 OF LKS DOES NOT SET"
883	001570	021575	DH10	:"TEST# ERR PC LKS"
884	001572	022244	DT10	:\$TESTN,\$ERRPC,LKS
885	001574	000000	0	
886				
887	001576	017212	EM41	:"RTC INTERRUPT AT PRIORITY 7"
888	001600	021575	DH10	:"TEST# ERR PC LKS"
889	001602	022244	DT10	:\$TESTN,\$ERRPC,LKS
890	001604	000000	0	
891				
892	001606	017246	EM45	:"RTC INTERRUPTS WHEN DISABLED"
893	001610	021575	DH10	:"TEST# ERR PC LKS"
894	001612	022244	DT10	:\$TESTN,\$ERRPC,LKS
895	001614	000000	0	
896				
897	001616	017303	EM46	:"RTC INTERRUPT DID NOT OCCUR"
898	001620	021575	DH10	:"TEST# ERR PC LKS"
899	001622	022244	DT10	:\$TESTN,\$ERRPC,LKS
900	001624	000000	0	
901				
902	001626	017303	EM47	:"RTC INTERRUPT DID NOT OCCUR"
903	001630	021575	DH10	:"TEST# ERR PC LKS"
904	001632	022244	DT10	:\$TESTN,\$ERRPC,LKS
905	001634	000000	0	
906				
907	001636	017337	EM50	:"RTC DOUBLE INTERRUPT"
908	001640	021575	DH10	:"TEST# ERR PC LKS"
909	001642	022244	DT10	:\$TESTN,\$ERRPC,LKS
910	001644	000000	0	
911				
912	001646	017364	EM51	:"RESET DID NOT CLEAR RTC INTERRUPT"
913	001650	021575	DH10	:"TEST# ERR PC LKS"
914	001652	022244	DT10	:\$TESTN,\$ERRPC,LKS
915	001654	000000	0	
916				
917	001656	017414	EM52	:"RTC INTERRUPT DID NOT CLEAR WITH BIT7 OF LKS"
918	001660	021575	DH10	:"TEST# ERR PC LKS"

919	001662	022244	DT10	;\$TESTN,\$ERRPC,LKS
920	001664	000000	0	
921				
922	001666	017471	EM53	
923	001670	021655	DH53	;'TEST# ERR PC LKS CNT1 CNT2''
924	001672	022266	DT53	;\$TESTN,\$ERRPC,LKS,FIRST,SECND
925	001674	000000	0	
926				
927	001676	017515	EM54	;'XMIT INTERRUPTS WHEN DISABLED''
928	001700	021451	DH1	;'TEST# ERR PC TCSR''
929	001702	022204	DT1	;\$TESTN,\$ERRPC,TCSR
930	001704	000000	0	
931				
932	001706	017653	EM55	;'XMIT DID NOT INTERRUPT''
933	001710	021451	DH1	;'TEST# ERR PC TCSR''
934	001712	022204	DT1	;\$TESTN,\$ERRPC,TCSR
935	001714	000000	0	
936				
937	001716	017553	EM56	;'XMIT INTERRUPT AT PRIORITY 7''
938	001720	021451	DH1	;'TEST# ERR PC TCSR''
939	001722	022204	DT1	;\$TESTN,\$ERRPC,TCSR
940	001724	000000	0	
941				
942	001726	017611	EM57	;'XMIT INTERRUPTS WITH ENABLE CLEAR''
943	001730	021451	DH1	;'TEST# ERR PC TCSR''
944	001732	022204	DT1	;\$TESTN,\$ERRPC,TCSR
945	001734	000000	0	
946				
947	001736	017653	EM60	;'XMIT DID NOT INTERRUPT''
948	001740	021451	DH1	;'TEST# ERR PC TCSR''
949	001742	022204	DT1	;\$TESTN,\$ERRPC,TCSR
950	001744	000000	0	
951				
952	001746	017702	EM61	;'XMIT RE-INTERRUPTED''
953	001750	021451	DH1	;'TEST# ERR PC TCSR''
954	001752	022204	DT1	;\$TESTN,\$ERRPC,TCSR
955	001754	000000	0	
956				
957	001756	017726	EM62	;'LOADING TBUF DID NOT CLEAR INTERRUPT''
958	001760	021451	DH1	;'TEST# ERR PC TCSR''
959	001762	022204	DT1	;\$TESTN,\$ERRPC,TCSR
960	001764	000000	0	
961				
962	001766	017773	EM63	;'RCVR ACTIVE NOT SET''
963	001770	021523	DH6	;'TEST# ERR PC RCSR''
964	001772	022224	DT6	;\$TESTN,\$ERRPC,RCSR
965	001774	000000	0	
966				
967	001776	020017	EM64	;'RECEIVER DONE NEVER SET''
968	002000	021523	DH6	;'TEST# ERR PC RCSR''
969	002002	022224	DT6	;\$TESTN,\$ERRPC,RCSR
970	002004	000000	0	
971				
972	002006	020043	EM65	;'RCVR ACTIVE NOT CLEARED WITH DONE SET2
973	002010	021523	DH6	;'TEST# ERR PC RCSR''
974	002012	022224	DT6	;\$TESTN,\$ERRPC,RCSR

975	002014	000000	0	
976				
977	002016	020111	EM66	;'RESET DID NOT CLEAR RCVR DONE''
978	002020	021523	DH6	;'TEST# ERR PC RCSR''
979	002022	022224	DT6	;\$TESTN,\$ERRPC,RCSR
980	002024	000000	0	
981				
982	002026	020147	EM67	;'RDR ENABLE SET DID NOT CLEAR RCVR DONE''
983	002030	021523	DH6	;'TEST# ERR PC RCSR''
984	002032	022224	DT6	;\$TESTN,\$ERRPC,RCSR
985	002034	000000	0	
986				
987	002036	020212	EM70	;'READING RBUF DID NOT CLEAR RCVR DONE''
988	002040	021523	DH6	;'TEST# ERR PC RCSR''
989	002042	022224	DT6	;\$TESTN,\$ERRPC,RCSR
990	002044	000000	0	
991				
992	002046	020257	EM71	;'RCVR INTERRUPTS WITH ENABLE CLEAR''
993	002050	021523	DH6	;'TEST# ERR PC RCSR''
994	002052	022224	DT6	;\$TESTN,\$ERRPC,RCSR
995	002054	000000	0	
996				
997	002056	020426	EM72	;'RCVR DID NOT INTERRUPT''
998	002060	021523	DH6	;'TEST# ERR PC RCSR''
999	002062	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1000	002064	000000	0	
1001				
1002	002066	020321	EM73	;'RCVR INTERRUPTS AT PRIORITY 7''
1003	002070	021523	DH6	;'TEST# ERR PC RCSR''
1004	002072	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1005	002074	000000	0	
1006				
1007	002076	020357	EM74	;'RCVR INTERRUPT REQUEST PASSED WITH ENABLE CLEAR''
1008	002100	021523	DH6	;'TEST# ERR PC RCSR''
1009	002102	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1010	002104	000000	0	
1011				
1012	002106	020426	EM75	;'RCVR DID NOT INTERRUPT''
1013	002110	021523	DH6	;'TEST# ERR PC RCSR''
1014	002112	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1015	002114	000000	0	
1016				
1017	002116	020455	EM76	;'RECEIVER RE-INTERRUPTED''
1018	002120	021523	DH6	;'TEST# ERR PC RCSR''
1019	002122	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1020	002124	000000	0	
1021				
1022	002126	020501	EM77	;'READING RBUF DID NOT CLEAR INTERRUPT''
1023	002130	021523	DH6	;'TEST# ERR PC RCSR''
1024	002132	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1025	002134	000000	0	
1026				
1027	002136	020546	EM100	;'RESET DID NOT CLEAR RCVR INTERRUPT''
1028	002140	021523	DH6	;'TEST# ERR PC RCSR''
1029	002142	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1030	002144	000000	0	

1031				
1032				
1033	002146	020611	EM101	;'OR FLAG DID NOT SET'
1034	002150	021523	DH6	
1035	002152	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1036	002154	000000	0	
1037				
1038	002156	020637	EM102	;'ERROR' NOT SET WITH 'OR' FLAG'
1039	002160	021523	DH6	;'TEST# ERR PC RCSR'
1040	002162	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1041	002164	000000	0	
1042				
1043	002166	020676	EM103	;'BREAK DID NOT TRANSM ALL ZEROES'
1044	002170	021722	DH103	;'TEST# ERR PC RCSR DATA'
1045	002172	022302	DT103	;\$TESTN,\$ERRPC,RCSR,\$BDDAT
1046	002174	000000	0	
1047				
1048	002176	020734	EM104	;'BREAK DID NOT SET FRAMING ERROR'
1049	002200	021523	DH6	;'TEST# ERR PC RCSR'
1050	002202	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1051	002204	000000	0	
1052				
1053	002206	020771	EM105	;'DATA COMPARE ERROR'
1054	002210	021757	DH105	;'TEST# ERR PC RCSR GOOD BAD'
1055	002212	022314	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1056	002214	000000	0	
1057				
1058	002216	021014	EM106	;'DATA COMPARE ERROR WITH CABLE'
1059	002220	021757	DH105	;'TEST# ERR PC RCSR GOOD BAD'
1060	002222	022314	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1061	002224	000000	0	
1062				
1063	002226	021052	EM107	;'TIMEOUT IN EXERCISER TEST'
1064	002230	021523	DH6	;'TEST# ERR PC RCSR'
1065	002232	022224	DT6	;\$TESTN,\$ERRPC,RCSR
1066	002234	000000	0	
1067				
1068	002236	021104	EM110	;'INCORRECT RECEIVE COUNT
1069	002240	022023	DH110	;'TEST# ERR PC RCSR TRANS RCV'
1070	002242	022330	DT110	;\$TESTN,\$ERRPC,RCSR,XMTCNT,RCVCNT
1071	002244	000000	0	
1072				
1073	002246	021134	EM111	;'DATA COMPARE ERROR IN EXERCISER'
1074	002250	021757	DH105	;'TEST# ERR PC RCSR GOOD BAD'
1075	002252	022314	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1076	002254	000000	0	
1077				
1078	002256	021174	EM112	;'TRAP CATCHER'
1079	002260	022067	DH112	;'TEST# ERR PC RCSR OLDPC TRAP ADR'
1080	002262	022344	DT112	;\$TESTN,\$ERRPC,RCSR,OLDPC,BDVECT
1081	002264	000000	0	
1082				
1083	002266	021211	EM113	;'NO CLK INTERRUPT IN EXERCISER'
1084	002270	021575	DH10	;'TEST# ERR PC LKS'
1085	002272	022244	DT10	;\$TESTN,\$ERRPC,LKS
1086	002274	000000	0	

```
1087
1088 002276 021247          EM114          ;''ERROR' NOT SET WITH 'FR' FLAG''
1089 002300 021523          DH6           ;'TEST# ERR PC  RCSR''
1090 002302 022224          DT6           ;$TESTN,$ERRPC,RCSR
1091 002304 000000          0
1092
1093 002306 021306          EM115          ;RCV ACTVIE NOT CLEAR WITH INIT
1094 002310 021523          DH6           ;'TEST# ERR PC  RCSR''
1095 002312 022224          DT6           ;$TESTN,$ERRPC,RCSR
1096 002314 000000          0
1097
1098 002316 021345          EM116          ;RCV ACTIVE WITHOUT ''START'' BIT
1099 002320 021523          DH6           ;'TEST# ERR PC  RCSR''
1100 002322 022224          DT6           ;$TESTN,$ERRPC,RCSR
1101 002324 000000          0
1102
1103 002326 021404          EM117          ;RDR ENABLE NOT CLEAR WITH RCV ACTIVE
1104 002330 021523          DH6           ;'TEST# ERR PC  RCSR''
1105 002332 022224          DT6           ;$TESTN,$ERRPC,RCSR
1106 002334 000000          0
1107
1108 002336 000000          CTSTFL: .WORD 0          ;CONSOLE UNDER TEST FLAG
1109 002340 000000          TMP1:  .WORD 0          ;TEMP LOCATION FOR TABLE OFFSETS
1110 002342 000000          TMP2:  .WORD 0          ;TEMP LOCATION FOR DEVICE COUNT
1111 002344 000000          TMP3:  .WORD 0          ;LOCATION FOR DEVICE MAP BIT TEST MASK
1112          ;REGISTER AND VECTOR ADDRESSES FOR THE DL-11W UNDER TEST
1113
1114 002346 000000          RCSR:  .WORD 0
1115 002350 000000          RBUF:  .WORD 0
1116 002352 000000          TCSR:  .WORD 0
1117 002354 000000          TBUF:  .WORD 0
1118 002356 000000          RVECT: .WORD 0
1119 002360 000000          RPSW:  .WORD 0
1120 002362 000000          TVECT: .WORD 0
1121 002364 000000          TPSW:  .WORD 0
1122
1123          ;CONSOLE REGISTER AND VECTOR ADDRESSES FOR THE DL-11W
1124
1125 002366 177560          CRCSR: 177560          ;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
1126 002370 177562          CRBUF: 177562          ;ADDRESS OF RECEIVER BUFFER
1127 002372 177564          CTCRSR:177564          ;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
1128 002374 177566          CTBUF: 177566          ;ADDRESS OF TRANSMITTER BUFFER
1129 002376 000060          CRVECT: 60            ;RECEIVER INTERRUPT VECTOR
1130 002400 000062          CRPSW: 62
1131 002402 000064          CTVECT: 64            ;TRANSMITTER INTERRUPT VECTOR
1132 002404 000066          CTPSW: 66
1133
1134          ;REAL TIME CLOCK REGISTER AND VECTOR ADDRESSES
1135 002406 177546          LKS:   .WORD 177546
1136 002410 000100          RTCVT: .WORD 100
1137 002412 000102          RTCPSW: .WORD 102
1138
1139 002414 000020          ADRTBL: .BLKW 20
1140 002454 000020          VCTTBL: .BLKW 20
1141
1142
```

ERROR POINTER TABLE

```
1143 ;SUBROUTINE TO GENERATE DEVICE ADDRESS TABLE
1144
1145 002514 012702 002414 DEVADR: MOV #ADRTBL,R2 ;POINT R2 TO THE DEVICE ADDRESS TABLE
1146 002520 013700 001142 MOV $BASE,R0 ;LOAD BASE DEVICE ADDRESS IN R0
1147 002524 010001 MOV R0,R1 ;
1148 002526 062701 000170 ADD #170,R1 ;POINT R1 TO LAST DEVICE ADDRESS
1149 002532 010022 1$: MOV R0,(R2)+ ;MOVE DEVICE ADDRESS TO TABLE
1150 002534 062700 000010 ADD #10,R0 ;POINT R0 TO NEXT DEVICE ADDRESS
1151 002540 020001 CMP R0,R1 ;FINISHED GENERATING TABLE?
1152 002542 003773 BLE 1$ ;BR, IF LAST DEVICE ADDRESS NOT LOADED
1153 002544 000207 RTS PC
1154
1155
1156
1157 002546 005037 001070 START: CLR $FATAL ;CLEAR ERROR NO.
1158 002552 005037 001066 CLR $MSGTYP ;CLEAR MESSAGE TYPE
1159 002556 005037 001072 CLR $TESTN ;CLEAR TEST NO.
1160 002562 005037 002336 CLR CTSTFL ;CLEAR CONSOLE UNDER TEST FLAG
1161 002566 005037 001076 CLR $DEVCT ;CLEAR DEVICE COUNT
1162 002572 005037 001100 CLR $UNIT ;CLEAR UNIT NUMBER
1163 002576 005737 001112 TST $USWR ;IS $USWR LOADED?
1164 002602 001003 BNE 1$ ;BR IF YES
1165 002604 012737 000400 001112 MOV #400,$USWR ;ELSE, DEFAULT TO $USWR=400
1166 002612 012737 000006 000004 1$: MOV #6,@#4 ;INITIALIZE TIMEOUT VECTORS TO TRAP
1167 002620 012737 000003 000006 MOV #3,@#6 ;CATCHER ROUTINE
1168
1169 .SBTTL INITIALIZE THE COMMON TAGS
1170 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1171 002626 012706 001000 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
1172 002632 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
1173 002634 022706 001040 CMP #SWR,R6 ;;DONE?
1174 002640 001374 BNE -6 ;;LOOP BACK IF NO
1175 002642 012706 001000 MOV #1000,SP ;;SETUP THE STACK POINTER
1176
1177 ;;INITIALIZE A FEW VECTORS
1177 002646 012737 013000 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1178 002654 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
1179 002662 012737 012304 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1180 002670 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
1181 002676 012737 014704 000034 MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1182 002704 012737 000340 000036 MOV #340,@TRAPVEC+2;LEVEL 7
1183 002712 012737 012622 000024 MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
1184 002720 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
1185 002726 013737 012132 012124 MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
1186 002734 005037 001060 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1187 002740 112737 000001 001015 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
1188 002746 012737 002746 001006 MOV #,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1189 002754 012737 002754 001010 MOV #,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
1190
1191 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1191 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1192 002762 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
1193 002766 012737 003022 000004 MOV #64,$ERRVEC ;;SET UP ERROR VECTOR
1194 002774 012737 177570 001040 MOV #DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1195 003002 012737 177570 001042 MOV #DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1196 003010 022777 177777 176022 CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
1197 003016 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1198 ;;AND THE HARDWARE SWR IS NOT = -1
```

INITIALIZE THE COMMON TAGS

```
1199 003020 000403          BR      65$          ;;BRANCH IF NO TIMEOUT
1200 003022 012716 003030    64$:    MOV      #65$, (SP)    ;;SET UP FOR TRAP RETURN
1201 003026 000002          RTI
1202 003030 012737 000176 001040 65$:    MOV      #SWREG,SWR    ;;POINT TO SOFTWARE SWR
1203 003036 012737 000174 001042    MOV      #DISPREG,DISPLAY
1204 003044 012637 000004    66$:    MOV      (SP)+,@#ERRVEC    ;;RESTORE ERROR VECTOR
1205
1206 003050 005037 001074          CLR      $PASS          ;;CLEAR PASS COUNT
1207 003054 132737 000200 001107    BITB    #APTSIZE,$ENVM    ;;TEST USER SIZE UNDER APT
1208 003062 001403          PEQ      67$          ;;YES,USE NON-APT SWITCH
1209 003064 012737 001110 001040    MOV      #$$SWREG,SWR    ;;NO,USE APT SWITCH REGISTER
1210 003072
1211 003072 032777 000020 175740    67$:    BIT      #BIT4,@SWR      ;TEST CLOCK ONLY?
1212 003100 001404          BEQ      INIT          ;BR IF NOT
1213 003102 005237 002336          INC      CTSTFL        ;ELSE, SET CONSOLE TEST FLAG TO ENABLE CLOCK TESTS
1214 003106 000137 004056          JMP      ID            ; AND JUMP TO TYPE PROGRAM ID
1215 003112 132737 000001 001106    INIT:   BITB    #BIT0,$ENV    ;CHECK IF ON APT
1216 003120 001404          BEQ      MANL          ;BR IF NOT APT
1217 003122 132737 000200 001107    BITB    #BIT7,$ENVM    ;DID APT SIZE
1218 003130 001056          BNE      APTSZD        ;BR, IF APT SIZED
1219 003132 032777 000040 175700    MANL:   BIT      #BIT5,@SWR    ;WAS '$DEVN' MANUALLY SET?
1220 003140 001052          BNE      APTSZD        ;IF YES, SKIP SELF-SIZING
1221
1222 003142 004737 002514          SIZE:   JSR      PC,DEVADR    ;GENERATE DEVICE ADDRESS TABLE
1223 003146 005037 002342          CLR      TMP2          ;CLR TEMP LOCATION TO KEEP DEVICE COUNT
1224 003152 005037 001144          CLR      $DEVN        ;CLEAR DEVICE MAP
1225 003156 013703 000004          MOV      @#4,R3        ;SAVE TIMEOUT VECTOR
1226 003162 012737 003212 000004    MOV      #4,@#4        ;SET TIMEOUT POINTER
1227 003170 013700 001142          MOV      $BASE,R0      ;LOAD BASE ADDRESS
1228 003174 062700 000160          ADD      #160,R0       ;POINT R0 TO UNIT #15 (UNIT#0=CONSOLE)
1229 003200 005710          3$:    TST      (R0)          ;CHECK FOR DEVICE EXISTANCE
1230 003202 005237 001144          INC      $DEVN        ;INDICATE DEVICE EXISTANCE IN DEVICE MAP
1231 003206 005237 002342          INC      TMP2          ;INCREMENT DEVICE COUNT
1232 003212 012706 001000          4$:    MOV      #1000,SP      ;RESET STACK POINTER
1233 003216 006337 001144          ASL      $DEVN        ;ADJUST DEVICE MAP FOR NEXT UNIT CHECK
1234 003222 162700 000010          SUB      #10,R0        ;POINT R0 TO NEXT DEVICE NUMBER
1235 003226 023700 001142          CMP      $BASE,R0      ;FINISHED SIZING?
1236 003232 003762          BLE      3$           ;BR, IF BASE ADDRESS HAS NOT BEEN CHECKED
1237 003234 013700 002366          MOV      CRCSR,R0      ;LOAD CONSOLE DEVICE ADDRESS
1238 003240 012737 003260 000004    MOV      #5,@#4        ;SET UP TIMEOUT POINTER
1239 003246 005710          TST      (R0)          ;TEST FOR CONSOLE EXISTANCE
1240 003250 005237 001144          INC      $DEVN        ;INDICATE CONSOLE EXISTANCE IN DEVICE MAP
1241 003254 005237 002342          INC      TMP2          ;INCREMENT DEVICE COUNT
1242 003260 010337 000004          5$:    MOV      R3,@#4        ;RESTORE TIMEOUT VECTOR
1243 003264 000415          BR      VCTADR        ;BR TO GENERATE VECTOR ADDRESS TABLE
1244
1245 003266 005037 002342          APTSZD: CLR      TMP2          ;CLEAR TEMP LOCATION TO KEEP DEVICE CNT
1246 003272 013702 001144          MOV      $DEVN,R2      ;MOVE DEVICE MAP TO R2
1247 003276 005702          TSTDVM: TST      R2      ;TEST MSB OF DEVICE MAP
1248 003300 100002          BPL      1$           ;BR, IF MSB IS ZERO
1249 003302 005237 002342          INC      TMP2          ;INCREMENT DEVICE COUNT, IF MSB=1
1250 003306 006302          1$:    ASL      R2           ;SHIFT NEXT BIT INTO MSB POSITION
1251 003310 001401          BEQ      DVADT        ;BR, IF NO OTHER BITS ARE SET IN $DEVN
1252 003312 000771          BR      TSTDVM        ;CONTINUE CHECKING $DEVN, IF MORE BITS SET
1253 003314 004737 002514          DVADT: JSR      PC,DEVADR    ;GENERATE DEVICE ADDRESS TABLE
1254
```

INITIALIZE THE COMMON TAGS

```
1255 ;GENERATE VECTOR ADDRESS TABLE
1256
1257 003320 012702 002454 VCTADR: MOV #VCTTBL,R2 ;GET LOCATION OF VECTOR TABLE
1258 003324 113700 001136 MOVB @#$VECT1,R0 ;COPY BASE VECTOR
1259 003330 042700 177400 BIC #177400,R0 ;CLEAR BYTE SIGN EXTENSION
1260 003334 010001 MOV R0,R1 ;
1261 003336 062701 000170 ADD #170,R1 ;POINT R1 TO LAST DEVICE VECTOR
1262 003342 010022 1$: MOV R0,(R2)+ ;PUT VECTOR ADDRESS IN TABLE
1263 003344 062700 000010 ADD #10,R0 ;POINT R0 TO NEXT VECTOR ADDRESS
1264 003350 020001 CMP R0,R1 ;FINISHED GENERATING VECTOR TABLE?
1265 003352 003773 BLE 1$ ;BR, IF LAST VECTOR IS NOT LOADED
1266
1267 ;MOVE DEVICE COUNT INTO DEVICE COUNT MESSAGE
1268
1269 003354 013700 002342 MOV TMP2,R0 ;COPY DEVICE COUNT INTO R0
1270 003360 005001 CLR R1 ;CLEAR AUXILARY REGISTER
1271 003362 000300 SWAB R0 ;PUT DEVICE COUNT IN UPPER BYTE OF R0
1272 003364 006300 ASL R0 ;MOVE MSB OF COUNT INTO
1273 003366 006300 ASL R0 ;MSB OF R0
1274 003370 006300 SHIFT: ASL R0 ;PUT MSB OF COUNT INTO CARRY
1275 003372 106101 ROLB R1 ;MOVE MSB OF COUNT INTO R1
1276 003374 006300 ASL R0 ;MOVE NEXT BIT TO CARRY
1277 003376 106101 ROLB R1 ;MOVE INTO R1
1278 003400 006300 ASL R0 ;MOVE LAST BIT OF DIGIT
1279 003402 106101 ROLB R1 ;INTO R1
1280 003404 062701 000060 ADD #60,R1 ;CONVERT DIGIT TO ASCII
1281 003410 000301 SWAB R1 ;MOVE DIGIT TO UPPER BYTE
1282 003412 032701 000020 BIT #BIT4,R1 ;HAVE BOTH DIGITS BEEN MOVED TO R1?
1283 003416 001764 BEQ SHIFT ;BR, IF NOT
1284 003420 010137 022154 MOV R1,M2A ;MOVE DEVICE COUNT TO OUTPUT MESSAGE
1285
1286
1287 003424 052737 000002 002344 BEGIN: BIS #BIT1,TMP3 ;SET UP BIT MASK TO TEST $DEVN FOR DEVICES EXCEPT CONSOL
1288 003432 005037 002340 CLR TMP1 ;CLEAR LOCATION TO STORE TABLE OFFSETS
1289 003436 032737 000001 001144 BIT #BIT0,$DEVN ;IS CONSOLE TO BE TESTED?
1290 003444 001001 BNE TCONS ;BR, IF CONSOLE IS TO BE TESTED
1291 003446 000414 BR TSTDEV ;BR, TO TEST OTHER DEVICES
1292 003450 005237 002336 TCONS: INC CTSTFL ;INDICATE CONSOLE UNDER TEST
1293 003454 012700 002366 MOV #CRCSR,R0 ;SET UP CONSOLE DEVICE ADDRESSES
1294 003460 012701 002346 MOV #RCSR,R1 ;POINT R1 TO UUT ADDRESS TABLE
1295 003464 012021 1$: MOV (R0)+,(R1)+ ;TRANSFER CONSOLE ADDRESSES
1296 003466 022701 002364 CMP #TPSW,R1 ;FINISHED TRANSFER?
1297 003472 002374 BGE 1$ ;BR, IF NOT
1298 003474 000137 003622 JMP TST1 ;GO TEST CONSOLE INTERFACE
1299
1300 ;PREPARE ADDRESSES AND VECTORS FOR UUT
1301 003500 033737 002344 001144 TSTDEV: BIT TMP3,$DEVN ;CHECK TO SEE IF DEVICE IS TO BE TESTED
1302 003506 001010 BNE SETADR ;BR, IF YES
1303 003510 006337 002344 ASL TMP3 ;SHIFT MASK TO CHECK NEXT $DEVN BIT
1304 003514 062737 000002 002340 ADD #2,TMP1 ;INCREMENT TABLE INDEX
1305 003522 005237 001100 INC $UNIT ;INCREMENT UNIT NUMBER
1306 003526 000764 BR TSTDEV ;GO TEST NEXT BIT OF DEVICE MAP
1307
1308 003530 005237 001100 SETADR: INC $UNIT ;UPDATE UNIT NUMBER
1309 003534 006337 002344 ASL TMP3 ;UPDATE DEVICE MAP TEST MASK
1310 003540 013702 002340 MOV TMP1,R2 ;MOVE TABLE OFFSET TO R2
```

INITIALIZE THE COMMON TAGS

```
1311 003544 062737 000002 002340      ADD    #2,TMP1      ;UPDATE TABLE OFFSET FOR NEXT DEVICE
1312 003552 016200 002414              MOV    ADRTBL(R2),R0 ;PUT UUT ADDRESS INTO R0
1313 003556 012701 002346              MOV    #RCSR,R1    ;POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
1314 003562 010021              ADR:  MOV    R0,(R1)+ ;TRANSFER UUT ADDRESS
1315 003564 062700 000002              ADD    #2,R0       ;POINT TO NEXT UUT REGISTER
1316 003570 030027 000006              BIT    R0,#6       ;FINISHED TRANSFER?
1317 003574 001372              BNE    ADR         ;BR, IF NOT
1318
1319 003576 016200 002454              VECT: MOV    VCTTBL(R2),R0 ;PUT UUT VECTOR INTO R0
1320 003602 010021              MOV    R0,(R1)+   ;TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
1321 003604 062700 000002              ADD    #2,R0       ;POINT TO NEXT VECTOR
1322 003610 030027 000006              BIT    R0,#6       ;FINISHED TRANSFER?
1323 003614 001372              BNE    VECT        ;BR, IF NOT
1324 003616 000137 003622              JMP    TST1        ;GO TEST DEVICE
1325
1326
```

INITIALIZE THE COMMON TAGS

```
1327
1328
1329
1330      ;:*****
1331      ;*TEST 1      TEST ABILITY TO REFERENCE TCSR
1332      ;:*****
1332 003622 000004      TST1:  SCOPE
1333 003624 013703 000004      MOV      @#4,R3      ;SAVE TIMEOUT VECTOR
1334 003630 012737 003644 000004      MOV      #1$,@#4      ;SET UP TIMEOUT VECTOR
1335 003636 005777 176510      TST      @TCSR      ;REFERENCE THE XMIT COMMAND/STATUS REG.
1336 003642 000412      BR      4$      ; GO TO END OF TEST
1337
1338 003644 022626      1$:  CMP      (SP)+,(SP)+      ;RESTORE SP AFTER TIMEOUT
1339 003646 005737 002336      TST      CTSTFL      ;CHECK IF DEVICE IS CONSOLE
1340 003652 001002      BNE      2$      ;IF YES, SKIP ERROR TYPEOUT
1341 003654 104001      ERROR   1      ;REPORT ERROR TO APT & TTY
1342 003656 000404      BR      4$      ;BR TO END OF TEST
1343
1344 003660 004737 013170      2$:  JSR      PC,$ATY4      ;: ONLY REPORT A FATAL ERROR
1345 003664 000001      1      ;: THE ERROR NUMBER (FROM APT LIST)
1346 003666 000000      3$:  HALT
1347 003670 010337 000004      4$:  MOV      R3,@#4      ;RESTORE TIMEOUT VECTOR
1348
1349
1350
1351      ;:*****
1352      ;*TEST 2      TEST ABILITY TO REFERENCE TBUF
1353      ;:*****
1354 003674 000004      TST2:  SCOPE
1355 003676 013703 000004      MOV      @#4,R3      ;SAVE TIMEOUT VECTOR
1356 003702 012737 003716 000004      MOV      #1$,@#4      ;SET UP TIMEOUT VECTOR
1357 003710 005777 176440      TST      @TBUF      ;REFERENCE THE XMIT BUFFER
1358 003714 000412      BR      4$      ;GO TO END OF TEST
1359
1360 003716 022626      1$:  CMP      (SP)+,(SP)+      ;RESTORE SP AFTER TIMEOUT
1361 003720 005737 002336      TST      CTSTFL      ;CHECK IF DEVICE IS CONSOLE
1362 003724 001002      BNE      2$      ;IF YES, SKIP ERROR TYPEOUT
1363 003726 104002      ERROR   2      ;REPORT ERROR TO APT & TTY
1364 003730 000404      BR      4$      ;BR TO END OF TEST
1365
1366 003732 004737 013170      2$:  JSR      PC,$ATY4      ;: ONLY REPORT A FATAL ERROR
1367 003736 000002      2      ;: THE ERROR NUMBER (FROM APT LIST)
1368 003740 000000      3$:  HALT
1369 003742 010337 000004      4$:  MOV      R3,@#4      ;RESTORE TIMEOUT VECTOR
```

```
1370
1371
1372
1373
1374
1375 003746 000004
1376 003750 005077 176400
1377 003754 105777 176372
1378 003760 100016
1379
1380
1381 003762 005077 176366
1382 003766 105777 176360
1383 003772 100011
1384
1385 003774 005737 002336
1386 004000 001002
1387 004002 104003
1388 004004 000404
1389 004006
1390 004006 004737 013170
1391 004012 000003
1392 004014 000000
1393 004016 005000
1394 004020 105777 176326
1395 004024 100414
1396 004026 005200
1397 004030 001373
1398
1399 004032 005737 002336
1400 004036 001002
1401 004040 104004
1402 004042 000405
1403 004044
1404 004044 004737 013170
1405 004050 000004
1406 004052 000000
1407 004054 000424
1408
1409
1410 004056 023737 000042 000046 ID:
1411 004064 001412
1412 004066 005737 001074
1413 004072 001007
1414 004074 005737 001076
1415 004100 001004
1416 004102 104401
1417 004104 022140
1418 004106 104401
1419 004110 022152
1420 004112 032777 000020 174720 6$:
1421 004120 001402
1422 004122 000137 005056

;*****
;*TEST 3 TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED
;*****
TST3: SCOPE
CLR @TBUF ;LOAD XBUF
TSTB @TCSR ;CHECK DONE
BPL 3$ ;BR IF CLEAR
;FILL SECOND BUFFER, BECAUSE REFRESH COULD CAUSE
; FIRST TEST TO FAIL
CLR @TBUF ;FILL DOUBLE BUFFER
TSTB @TCSR ;CHECK DONE
BPL 3$ ;BR IF CLEAR

TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 1$ ;IF YES, SKIP ERROR TYPEOUT
ERROR 3 ;DONE NOT CLEARED WITH TBUF FULL
BR 2$ ;BR TO END OF TEST

1$: JSR PC,$ATY4 ;: ONLY REPORT A FATAL ERROR
3 ;: THE ERROR NUMBER (FROM APT LIST)
2$: HALT ;TCSR "DONE" NOT CLEARED WITH TBUF FULL
3$: CLR R0 ;CLEAR TIMER
4$: TSTB @TCSR ;CHECK FOR XMIT DONE
BMI ID ;IF DONE SETS, BR TO END OF TEST
INC R0 ;INCREMENT TIMER
BNE 4$ ;BR IF TIMER NOT DONE

TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 5$
ERROR 4 ;TCSR "DONE" DOES NOT SET
BR ID ;BR TO END OF TEST

5$: JSR PC,$ATY4 ;: ONLY REPORT A FATAL ERROR
4 ;: THE ERROR NUMBER (FROM APT LIST)
HALT
BR TST4 ;BR TO NEXT TEST, AND SKIP THE TYPEOUT THAT FOLLOWS
; BECAUSE OF THIS FAILURE

ID: CMP @#42,@#46 ;UNDER ACT11?
BEQ 6$ ;IF YES, SKIP IDENT. TYPEOUT
TST $PASS ;IS THIS THE FIRST PASS?
BNE 6$ ;IF NOT BR TO NEXT TEST & SKIP THE IDENTIFICATION TYPEOU
TST $DEVCT ;IS THIS THE FIRST SUBPASS?
BNE 6$ ;IF NOT, BR TO NEXT TEST
TYPE ;TYPE PROGRAM IDENTIFICATION
M1 ;TYPE NUMBER OF DEVICES UNDER TEST
TYPE
M2
BIT #BIT4,@SWR ;CLOCK TEST ONLY?
BEQ TST4 ;BR IF NOT
JMP TCLOCK ;ELSE, JUMP TO TEST CLOCK
```


TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED

```
1423
1424
1425
1426      ;*****
1427      ;*TEST 4      TEST THAT TCSR 'DONE' SETS WITH RESET
1428      ;*****
1428 004126 000004
1429 004130 005077 176220
1430 004134 105777 176212
1431 004140 100375
1432 004142 005077 176206
1433 004146 000240
1434 004150 000005
1435 004152 105777 176174
1436 004156 100401
1437
1438 004160 104005
1439
1440
1441
1442      ;*****
1443      ;*TEST 5      TEST ABILITY TO ACCESS RCSR
1444      ;*****
1445 004162 000004
1446 004164 013703 000004
1447 004170 012737 004204 000004
1448 004176 005777 176144
1449 004202 000402
1450
1451 004204 022626
1452 004206 104006
1453 004210 010337 000004

TST4:  SCOPE
      CLR @TBUF ;LOAD TRANSMIT BUFFER
1$:   TSTB @TCSR ;WAIT FOR DONE
      BPL 1$
      CLR @TBUF ;LOAD SECOND BUFFER
      NOP
      RESET ;CLEAR DONE WITH RESET
      TSTB @TCSR ;CHECK FOR DONE SET
      BMI TST5 ;BR TO NEXT TEST IF DONE SET

      ERROR 5 ;TCSR 'DONE' DOES NOT SET WITH RESET

TST5:  SCOPE
      MOV @#4,R3 ;SAVE TIMEOUT VECTOR
      MOV #1$,@#4 ;SET UP TIMEOUT VECTOR
      TST @RCSR ;ACCESS RCSR
      BR 2$ ;BR TO END OF TEST

1$:   CMP (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
      ERROR 6 ;CAN NOT ACCESS RCSR
2$:   MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
```

T5 TEST ABILITY TO ACCESS RCSR

1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471

```
*****  
*TEST 6 TEST ABILITY TO ACCESS RBUF  
*****  
TST6: SCOPE  
MOV @#4,R3 ;SAVE TIMEOUT VECTOR  
MOV #1$,@#4 ;SET UP TIMEOUT VECTOR  
TST @RBUF ;ACCESS RBUF  
BR 2$ ;BR TO END OF TEST  
  
1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT  
ERROR 7 ;CAN NOT ACCESS RBUF  
2$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
```

TEST ABILITY TO ACCESS RBUF

```
1472
1473
1474
1475          ;:*****
1476          ;*TEST 7      TEST THAT BITO(BREAK BIT) CAN BE SET & CLEARED & RESET
1477          ;:*****
1477 004246 000004          TST7:  SCOPE
1478 004250 032777 000400 174562      BIT      #BIT8,@SWR      ;IS BREAK FUNCTION ENABLED?
1479 004256 001545          BEQ      TST11          ;BR TO NEXT TEST, IF NOT
1480 004260 000005          RESET          ;CLEAR EVERYTHING
1481 004262 032777 000001 176062      BIT      #BIT0,@TCSR    ;CHECK BIT0 OF TCSR CLEAR
1482 004270 001411          BEQ      3$              ;BR IF CLEAR
1483 004272 005737 002336          TST      CTSTFL
1484 004276 001002          BNE      1$              .
1485 004300 104011          ERROR    11              ;BIT0 WAS NOT CLEAR AFTER RESET
1486 004302 000404          BR       3$
1487 004304          1$:
1488 004304 004737 013170          JSR      PC,$ATY4          ;;ONLY REPORT A FATAL ERROR
1489 004310 000011          11              ;;THE ERROR NUMBER (FROM APT LIST)
1490 004312 000000          2$:  HALT
1491
1492 004314 052777 000001 176030      3$:  BIS      #BIT0,@TCSR    ;SET BIT0 IN TCSR
1493 004322 032777 000001 176022      BIT      #BIT0,@TCSR    ;TEST BIT0 OF TCSR
1494 004330 001001          BNE      4$              ;BR IF SET
1495
1496 004332 104012          ERROR    12              ;BIT0 OF TCSR WILL NOT SET
1497
1498 004334 042777 000001 176010      4$:  BIC      #BIT0,@TCSR    ;CLEAR BIT0 OF TCSR
1499 004342 032777 000001 176002      BIT      #BIT0,@TCSR    ;TEST BIT0 OF TCSR
1500 004350 001411          BEQ      7$              .
1501 004352 005737 002336          TST      CTSTFL
1502 004356 001002          BNE      5$              .
1503 004360 104013          ERROR    13              ;BIT0 OF TCSR WILL NOT CLEAR
1504 004362 000404          BR       7$
1505 004364          5$:
1506 004364 004737 013170          JSR      PC,$ATY4          ;;ONLY REPORT A FATAL ERROR
1507 004370 000013          13              ;;THE ERROR NUMBER (FROM APT LIST)
1508 004372 000000          6$:  HALT
1509
1510 004374 052777 000001 175750      7$:  BIS      #BIT0,@TCSR    ;SET BIT0 IN TCSR
1511 004402 000005          RESET          ;CLEAR BIT0 WITH RESET
1512 004404 032777 000001 175740      BIT      #BIT0,@TCSR    ;TEST BIT0 CLEAR
1513 004412 001467          BEQ      TST11          ;BR IF CLEAR
1514 004414 042777 000001 175730      BIC      #BIT0,@TCSR    ;CLEAR BIT0, TO PRINT ERROR
1515 004422 104014          ERROR    14              ;RESET DID NOT CLEAR BIT0 OF TCSR
```

TEST THAT BIT0(BREAK BIT) CAN BE SET & CLEARED & RESET

```
1516
1517
1518
1519
1520
1521 004424 000004
1522 004426 000005
1523 004430 032777 000004 175714
1524 004436 001411
1525
1526 004440 005737 002336
1527 004444 001002
1528 004446 104015
1529 004450 000404
1530
1531 004452
1532 004452 004737 013170
1533 004456 000015
1534 004460 000000
1535
1536 004462 052777 000004 175662
1537 004470 032777 000004 175654
1538 004476 001001
1539
1540 004500 104016
1541
1542 004502 042777 000004 175642
1543 004510 032777 000004 175634
1544 004516 001411
1545
1546 004520 005737 002336
1547 004524 001002
1548 004526 104017
1549 004530 000404
1550 004532
1551 004532 004737 013170
1552 004536 000017
1553 004540 000000
1554
1555 004542 052777 000004 175602
1556 004550 000005
1557 004552 032777 000004 175572
1558 004560 001461
1559
1560 004562 042777 000004 175562
1561 004570 104020
1562

::*****
:*TEST 10 TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET
::*****
TST10: SCOPE
RESET ;CLEAR EVERYTHING
BIT #BIT2,@TCSR ;TEST FOR BIT2 OF TCSR CLEAR
BEQ 3$ ;BR IF CLEAR

TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 1$ ;IF YES, SKIP ERROR TYPEOUT
ERROR 15 ;BIT2 OF TCSR NOT CLEAR AFTER RESET
BR 3$

1$: JSR PC,$ATY4 ;:ONLY REPORT A FATAL ERROR
15 ;:THE ERROR NUMBER (FROM APT LIST)
2$: HALT

3$: BIS #BIT2,@TCSR ;SET BIT2 OF TCSR
BIT #BIT2,@TCSR ;TEST FOR BIT2 SET
BNE 4$ ;BR IF SET

ERROR 16 ;BIT2 OF TCSR WILL NOT SET

4$: BIC #BIT2,@TCSR ;CLEAR BIT2 OF TCSR
BIT #BIT2,@TCSR ;TEST BIT2 CLEAR
BEQ 7$ ;BR IF CLEAR

TST CTSTFL ;CHECK IF DEVICE IS CONSOLE
BNE 5$ ;IF YES, SKIP ERROR TYPEOUT
ERROR 17
BR 7$

5$: JSR PC,$ATY4 ;:ONLY REPORT A FATAL ERROR
17 ;:THE ERROR NUMBER (FROM APT LIST)
6$: HALT ;BIT0 OF TCSR WILL NOT CLEAR

7$: BIS #BIT2,@TCSR ;SET BIT2 OF TCSR
RESET ;CLEAR BIT2 WITH RESET
BIT #BIT2,@TCSR ;TEST FOR BIT2 CLEAR
BEQ TST12 ;IF CLEAR, GO TO NEXT TEST

BIC #BIT2,@TCSR ;CLEAR BIT2, TO PRINT ERROR
ERROR 20 ;RESET DID NOT CLEAR BIT2 OF TCSR
```

T10 TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET

```
1563
1564
1565
1566
1567
1568 004572 000004
1569 004574 000005
1570 004576 017703 175560
1571 004602 012777 004632 175552
1572 004610 004737 012242
1573 004614 000340
1574 004616 032777 000100 175526
1575 004624 001404
1576 004626 104021
1577
1578 004630 000402
1579
1580 004632 022626 1$:
1581 004634 104022
1582
1583
1584 004636 052777 000100 175506 2$:
1585 004644 032777 000100 175500
1586 004652 001001
1587
1588 004654 104023
1589
1590
1591 004656 042777 000100 175466 3$:
1592 004664 032777 000100 175460
1593 004672 001401
1594 004674 104024
1595
1596
1597 004676 052777 000100 175446 4$:
1598 004704 000005
1599 004706 032777 000100 175436
1600 004714 001401
1601
1602 004716 104025
1603
1604 004720 010377 175436 5$:
```

*TEST 11 TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET

TST11: SCOPE
RESET ;CLEAR EVERYTHING
MOV @TVECT,R3 ;SAVE XMIT VECTOR
MOV #1\$,@TVECT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
JSR PC,WRPSW ;SET PSW TO PRIORITY=7
 .WORD 340
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BEQ 2\$;BR IF ZERO
ERROR 21
 ;BIT6 IN TCSR NOT CLEAR AFTER RESET
BR 2\$
1\$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 22
 ;XMIT INTERRUPT OCCURRED PRIO=7
2\$: BIS #BIT6,@TCSR ;SET BIT6 OF TCSR
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BNE 3\$;BR, IF SET
ERROR 23
 ;CANNOT SET BIT6 OF TCSR
3\$: BIC #BIT6,@TCSR ;CLEAR BIT6 OF TCSR
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BEQ 4\$;BR IF CLEAR
ERROR 24
 ;CANNOT CLEAR BIT6 OF TCSR
4\$: BIS #BIT6,@TCSR ;SET BIT6 OF TCSR
RESET ;CLEAR BIT6 WITH RESET
BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
BEQ 5\$;BR IF CLEAR
ERROR 25
 ;CANNOT CLEAR BIT6 OF TCSR WITH RESET
5\$: MOV R3,@TVECT ;RESTORE XMIT VECTOR

```
1605
1606
1607
1608
1609
1610 004724 000004
1611 004726 000005
1612 004730 017703 175422
1613 004734 012777 004764 175414
1614 004742 004737 012242
1615 004746 000340
1616 004750 032777 000100 175370
1617 004756 001404
1618 004760 104026
1619
1620 004762 000402
1621
1622 004764 022626 1$:
1623 004766 104027 ERROR 27
1624
1625
1626 004770 052777 000100 175350 2$:
1627 004776 032777 000100 175342 BIT #BIT6,@RCSR
1628 005004 001001 BNE 3$ ;SET BIT6 OF RCSR
1629 ;TEST BIT6 OF RCSR
1630 005006 104030 ERROR 30 ;BR, IF SET
1631 ;CANNOT SET BIT6 OF RCSR
1632
1633 005010 042777 000100 175330 3$:
1634 005016 032777 000100 175322 BIC #BIT6,@RCSR
1635 005024 001401 BIT #BIT6,@RCSR
1636 ;CLEAR BIT6 OF RCSR
1637 005026 104031 ERROR 31 ;TEST BIT6 OF RCSR
1638 ;BR, IF CLEAR
1639 ;CANNOT CLEAR BIT6 OF RCSR
1640 005030 052777 000100 175310 4$:
1641 005036 000005 BIS #BIT6,@RCSR
1642 005040 032777 000100 175300 RESET ;SET BIT6 OF RCSR
1643 005046 001401 BIT #BIT6,@RCSR ;CLEAR BIT6 OF RCSR WITH RESET
1644 ;TEST BIT6 OF RCSR
1645 005050 104032 ERROR 32 ;BR, IF CLEAR
1646 ;CANNOT CLEAR BIT6 OF RCSR WITH RESET
1647 005052 010377 175300 5$:
1648 MOV R3,@RVECT ;RESTORE RECEIVE VECTOR
```

TEST THAT BIT6 OF RCSR CAN BE SET & RESET

```
1649
1650 005056 012737 000012 001002 TCLOCK: MOV #12,$STNM ;ADJUST TEST NUMBER TO (NEXT TEST - 1)
1651 ;*****
1652 ;*TEST 13 TEST ABILITY TO ACCESS LKS
1653 ;*****
1654 005064 000004 TST13: SCOPE
1655 005066 005737 002336 TST CTSTFL ;IS CONSOLE UNDER TEST?
1656 005072 001420 BEQ TST14 ;IF NOT, SKIP THIS TEST
1657 005074 032777 000100 173736 BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
1658 005102 001014 BNE TST14 ;IF YES, SKIP THIS TEST
1659 005104 013703 000004 MOV @#4,R3 ;SAVE TIMEOUT VECTOR
1660 005110 012737 005124 000004 MOV #1$,@#4 ;SET UP TIMEOUT VECTOR
1661 005116 005777 175264 TST @LKS ;ACCESS LKS
1662 005122 000402 BR 2$ ;NO TIMEOUT - BR TO END OF TEST
1663
1664 005124 022626 1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
1665 005126 104010 ERROR 10 ;CAN NOT ACCESS LKS
1666
1667 005130 010337 000004 2$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
1668
1669 ;*****
1670 ;*TEST 14 TEST THAT BIT6 OF LKS CAN BE SET & RESET
1671 ;*****
1672 005134 000004 TST14: SCOPE
1673 005136 005737 002336 TST CTSTFL ;IS CONSOLE UNDER TEST?
1674 005142 001460 BEQ TST15 ;IF NOT, SKIP THIS TEST
1675 005144 032777 000100 173666 BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
1676 005152 001054 BNE TST15 ;IF YES, SKIP THIS TEST
1677 005154 000005 RESET
1678 005156 017703 175226 MOV @ARTCVT,R3 ;SAVE LINE CLOCK VECTOR
1679 005162 012777 005212 175220 MOV #1$,@ARTCVT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
1680 005170 004737 012242 JSR PC,WRPSW ;SET PSW TO PRIORITY 7
1681 005174 000340 .WORD 340
1682 005176 032777 000100 175202 BIT #BIT6,@LKS ;TEST BIT6 OF LKS
1683 005204 001404 BEQ 2$
1684 005206 104033 ERROR 33
1685 ;BIT6 OF LKS NOT CLEAR AFTER RESET
1686 005210 000402 BR 2$
1687
1688 005212 022626 1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1689 005214 104034 ERROR 34 ;LKS INTERRUPT WITH PRIORITY=7
1690
1691
1692 005216 052777 000100 175162 2$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
1693 005224 032777 000100 175154 BIT #BIT6,@LKS ;TEST BIT6 OF LKS
1694 005232 001001 BNE 3$ ;BR IF SET
1695
1696 005234 104035 ERROR 35
1697 ;CANNOT SET BIT6 OF LKS
1698
1699 005236 042777 000100 175142 3$: BIC #BIT6,@LKS ;CLEAR BIT6 OF LKS
1700 005244 032777 000100 175134 BIT #BIT6,@LKS ;TEST BIT6 OF LK
1701 005252 001401 BEQ 4$
1702 005254 104036 ERROR 36
1703 ;CANNOT CLEAR BIT6 OF LKS
1704 005256 052777 000100 175122 4$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
```

TEST THAT BIT6 OF LKS CAN BE SET & RESET

1705	005264	000005			RESET			;CLEAR BIT6 OF LKS WITH RESET
1706	005266	032777	000100	175112	BIT	#BIT6,@LKS		;TEST BIT6 OF LKS
1707	005274	001401			BEQ	5\$;BR IF CLEAR
1708								
1709	005276	104037			ERROR	37		
1710								
1711	005300	010377	175104	5\$:	MOV	R3,@RTCVT		;CANNOT CLEAR BIT6 OF LKS WITH RESET ;RESTORE LINE CLOCK VECTOR

.P
CZ


```

1712
1713
1714
1715
1716
1717 005304 000004
1718 005306 013703 000004
1719 005312 013704 000006
1720 005316 012737 005450 000004
1721 005324 012737 000340 000006
1722 005332 000005
1723 005334 012700 000002
1724 005340 032777 000020 173472
1725 005346 001404
1726 005350 013737 002406 001020
1727 005356 000403
1728 005360 013737 002346 001020 1$:
1729 005366 013737 001020 001022 2$:
1730 005374 040037 001022
1731 005400 023737 001020 001022
1732 005406 001002
1733 005410 050037 001022
1734 005414 017737 173402 001024 3$:
1735 005422 052777 000100 173372
1736 005430 032777 000100 173362
1737 005436 001011
1738 005440 013777 001024 173354
1739 005446 000401
1740 005450 022626 4$:
1741 005452 006300 5$:
1742 005454 105700
1743 005456 100343
1744 005460 000401
1745
1746 005462 104040 6$:
1747
1748
1749
1750 005464 010337 000004 7$:
1751 005470 010437 000006

```

```

*****
: *TEST 15      TEST FOR DUAL ADDRESSING OF REGISTERS
*****
TST15:  CCOPE
        MOV     @#4,R3      ;SAVE TIMEOUT VECTOR
        MOV     @#6,R4      ;SAVE TIMEOUT PSW
        MOV     #45,@#4     ;SET UP TIMEOUT VECTOR
        MOV     #340,@#6    ;KEEP PRIO=7
        RESET                    ;CLEAR EVERYTHING
        MOV     #BIT1,R0    ;SET UP BIT MASK
        BIT     #BIT4,@SWR   ;CLOCK TEST ONLY?
        BEQ     1$          ;BR IF NOT
        MOV     LKS,$GDADR  ;ELSE, MOVE GOOD LKS ADDRESS INTO $GDADR
        BR     2$
        MOV     RCSR,$GDADR ;MOVE GOOD RCSR ADDRESS INTO $GDADR
        MOV     $GDADR,$BDADR ;MOVE GOOD ADDRESS INTO TEST ADDRESS LOCATION
        BIC     R0,$BDADR   ;CREATE BAD ADDRESS BY COMPLEMENTING ONE BIT
        CMP     $GDADR,$BDADR ;ARE ADDRESSES IDENTICAL?
        BNE     3$          ;IF NOT, TEST THIS ADDRESS
        BIS     R0,$BDADR   ;ELSE, BIT SET THIS BIT POSITION TO GENERATE BAD ADDRESS
        MOV     @$BDADR,$GDDAT ;SAVE CONTENTS OF BAD ADDRESS IF IT EXISTS
        BIS     #BIT6,$BDADR ;SET BIT6 USING BAD ADDRESS
        BIT     #BIT6,$GDADR ;CHECK TO SEE IF GOOD ADDRESS CONTAINS BIT6
        BNE     6$          ;BR IF SET ---> ERROR
        MOV     $GDDAT,$BDADR ;RESTORE ANY MEMORY LOCATION THAT WAS ALTERED
        BR     5$          ;BR TO CONTINUE TEST
        CMP     (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
        ASL     R0          ;SHIFT BIT MASK TO NEXT POSITION
        TSTB   R0          ;COMPLEMENTED ALL BITS FROM 1 - 7?
        BPL     2$          ;BR, IF NOT.
        BR     7$          ;BR TO NEXT TEST

        ERROR 40          ;DUAL ADDRESSING ERROR
                          ;$BDADR = DUAL ADDRESS
                          ;$GDADR = GOOD ADDRESS

        MOV     R3,@#4     ;RESTORE TIMEOUT VECTOR
        MOV     R4,@#6     ;RESTORE TIMEOUT PSW

```

T15 TEST FOR DUAL ADDRESSING OF REGISTERS

```

1752
1753
1754
1755
1756
1757 005474 000004
1758 005476 005737 002336
1759 005502 001437
1760 005504 032777 000100 173326
1761 005512 001033
1762 005514 000005
1763 005516 105777 174664
1764 005522 100401
1765
1766 005524 104041
1767
1768 005526 042777 000200 174652
1769 005534 032777 000200 174644
1770 005542 001410
1771 005544 042777 000200 174634
1772 005552 032777 000200 174626
1773 005560 001401
1774
1775 005562 104042
1776
1777 005564 005000
1778 005566 105777 174614
1779 005572 100403
1780 005574 005200
1781 005576 001373
1782
1783 005600 104043
1784

```

```

*****
*TEST 16 TEST THAT BIT7 OF LKS SETS & CAN BE CLEARED
*****
TST16: SCOPE
TST CTSTFL ;IS CONSOLE UNDER TEST?
BEQ TST17 ;IF NOT, SKIP THIS TEST
BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
BNE TST17 ;IF YES, SKIP THIS TEST
RESET ;CLEAR EVERYTHING & SET BIT7 OF LKS
1$: TSTB @LKS ;TEST FOR BIT7 OF LKS
BMI 2$ ;BR IF SET
ERROR 41 ;BIT7 OF LKS DID NOT SET WITH RESET
2$: BIC #BIT7,@LKS ;CLEAR BIT7 OF LKS
BIT #BIT7,@LKS ;TEST BIT7 OF LKS
BEQ 3$
1771 005544 042777 000200 174634 BIC #BIT7,@LKS ;TRY ONE MORE TIME BECAUSE THE CLOCK
1772 005552 032777 000200 174626 BIT #BIT7,@LKS ; MAY HAVE SET IMMEDIATELY AFTER THE FIRST CLEAR
1773 005560 001401 DEQ 3$
ERROR 42 ;CAN NOT CLEAR BIT7 OF LKS
3$: CLR R0 ;CLEAR TIMER
CONT: TSTB @LKS ;TEST FOR BIT7 OF LKS
BMI TST17 ;BR, IF SET
INC R0 ;INCREMENT TIMER
BNE CONT ;CONTINUE UNTIL TIME EXPIRES
ERROR 43 ;BIT7 OF LKS DOES NOT SET

```

T16 TEST THAT BIT7 OF LKS SETS & CAN BE CLEARED

```
1785
1786
1787 ::*****
1788 :*TEST 17 TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY
1789 :*****
1789 005602 000004 TST17: SCOPE
1790 005604 005737 002336 TST CTSTFL ;IS CONSOLE UNDER TEST?
1791 005610 001503 BEQ TST20 ;IF NOT, SKIP THIS TEST
1792 005612 032777 000100 173220 BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
1793 005620 001077 BNE TST20 ;IF YES, SKIP THIS TEST
1794 005622 004737 012242 JSR PC,WRPSW ;SET PSW TO PRIORITY 7
1795 005626 000340 .WORD 340
1796 005630 017703 174554 MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
1797 005634 017704 174552 MOV @RTCP SW,R4 ;SAVE LINE CLOCK PSW VECTOR
1798 005640 012777 005702 174542 MOV #2,@RTCVT ;SET RTC INTERRUPT VECTOR TO ERROR REPORT
1799 005646 012777 000340 174536 MOV #340,@RTCP SW ;KEEP PRIORITY AT 7
1800 005654 042777 000200 174524 BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG
1801 005662 052777 000100 174516 BIS #BIT6,@LKS ;SET INTERRUPT ENABLE
1802 005670 105777 174512 1$: TSTB @LKS ;WAIT FOR RTC DONE(INTERRUPT REQUEST)
1803 005674 100375 BPL 1$
1804 005676 000240 NOP ;GIVE TIME FOR ANY INTERRUPTS
1805 005700 000402 BR 3$ ;BR, IF NO INTERRUPT OCCURS
1806
1807 005702 022626 2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1808 005704 104044 ERROR 44 ;RTC INTERRUPTS AT PRIORITY 7
1809
1810 005706 005077 174474 3$: CLR @LKS ;DISABLE RTC INTERRUPTS & CLEAR DONE
1811 005712 012777 005740 174470 MOV #4,@RTCVT ;SET RTC INTERRUPT VECTOR FOR ERROR
1812 005720 004737 012242 JSR PC,WRPSW ;CHANGE PSW TO PRIORITY 5
1813 005724 000240 .WORD 240
1814 005726 105777 174454 20$: TSTB @LKS ;WAIT FOR DONE (INTERRUPT REQUEST)
1815 005732 100375 BPL 20$
1816 005734 000240 NOP ;GIVE TIME FOR ANY INTERRUPT
1817 005736 000402 BR 5$ ;IF NO INTERRUPT - BR TO CONTINUE TEST
1818
1819 005740 022626 4$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1820 005742 104045 ERROR 45 ;RTC INTERRUPTS WITH INTERRUPTS DISABLED
1821
1822 005744 012777 006000 174436 5$: MOV #7,@RTCVT ;POINT RTC VECTOR TO END OF TEST
1823 005752 042777 000200 174426 BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG
1824 005760 052777 000100 174420 BIS #BIT6,@LKS ;ALLOW INTERRUPTS
1825 005766 105777 174414 6$: TSTB @LKS ;WAIT FOR RTC DONE
1826 005772 100375 BPL 6$
1827 005774 000240 NOP ;GIVE TIME FOR INTERRUPT
1828
1829 005776 104046 ERROR 46 ;RTC INTERRUPT DID NOT OCCUR
1830
1831 006000 022626 7$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
1832 006002 042777 000100 174376 BIC #BIT6,@LKS ;DISABLE INTERRUPTS
1833 006010 010377 174374 MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR
1834 006014 010477 174372 MOV R4,@RTCP SW ;RESTORE LINE CLOCK PSW VECTOR
1835
1836
```

TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY

1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876

006020 000004
006022 005737 002336
006026 001457
006030 032777 000100 173002
006036 001053
006040 000005
006042 017703 174342
006046 017704 174340
006052 012777 006122 174330
006060 012777 000340 174324
006066 004737 012242
006072 000240
006074 042777 000200 174304
006102 052777 000100 174276
006110 105777 174272
006114 100375
006116 000240
006120 104047
006122 022626
006124 012777 006144 174256
006132 004737 012242
006136 000240
006140 000240
006142 000402
006144 022626
006146 104050
006150 042777 000100 174230
006156 010377 174226
006162 010477 174224

```
:::*****  
:*TEST 20 TEST RTC FOR DOUBLE INTERRUPTS  
:::*****  
TST20: SCOPE  
TST CTSTFL ;IS CONSOLE UNDER TEST?  
BEQ TST21 ;IF NOT, SKIP THIS TEST  
BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?  
BNE TST21 ;IF YES, SKIP THIS TEST  
RESET ;CLEAR EVERYTHING  
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR  
MOV @RTCPSW,R4 ;SAVE LINE CLOCK PSW VECTOR  
MOV #2,@RTCVT ;SET UP RTC INTERRUPT VECTOR  
MOV #340,@RTCPSW ;DISALLOW INTERRUPTS AFTER THE INTERRUPT  
JSR PC,WRPSW ;SET PRIORITY TO 5  
 .WORD 240  
BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG  
BIS #BIT6,@LKS ;ENABLE CLOCK INTERRUPTS  
1$: TSTB @LKS ;WAIT FOR DONE  
BPL 1$  
NOP ;GIVE TIME FOR ANY INTERRUPT  
ERROR 47 ;RTC INTERRUPT DID NOT OCCUR  
2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
MOV #3,@RTCVT ;POINT RTC VECTOR TO ERROR REPORT  
JSR PC,WRPSW ;SET PSW TO PRIORITY 5  
 .WORD 240  
NOP ;GIVE SOME TIME FOR AN INTERRUPT  
BR 4$ ;NO INTERRUPT - BR TO END OF TEST  
3$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
ERROR 50 ;INTERRUPT SEQUENCE DID NOT CLEAR  
 ;INTERRUPT REQUEST  
4$: BIC #BIT6,@LKS ;DISABLE CLOCK INTERRUPTS  
MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR  
MOV R4,@RTCPSW ;RESTORE LINE CLOCK PSW VECTOR
```

```

1877
1878
1879
1880
1881
1882 006166 000004
1883 006170 005737 002336
1884 006174 001442
1885 006176 032777 000100 172634
1886 006204 001036
1887 006206 004737 012242
1888 006212 000340
1889 006214 017703 174170
1890 006220 012777 006272 174162
1891 006226 042777 000200 174152
1892 006234 052777 000100 174144
1893 006242 105777 174140 1$:
1894 006246 100375
1895 006250 000005
1896 006252 004737 012242
1897 006256 000240
1898 006260 000240
1899 006262 042777 000100 174116
1900 006270 000402
1901
1902 006272 022626 2$:
1903 006274 104051
1904
1905 006276 010377 174106 3$:

```

```

*****
*TEST 21 TEST THAT RTC INTERRUPT CLEARS WITH RESET
*****
TST21: SCOPE
TST CTSTFL ;IS CONSOLE UNDER TEST?
BEQ TST22 ;IF NOT, SKIP THIS TEST
BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
BNE TST22 ;IF YES, SKIP THIS TEST
JSR PC,WRPSW ;SET PRIORITY TO 7
.WORD 340
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV #2$,@RTCVT ;POINT RTC VECTOR TO ERROR REPORT
BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG
BIS #BIT6,@LKS ;ENABLE CLOCK INTERRUPTS
1$: TSTB @LKS ;WAIT FOR DONE (INTERRUPT REQUEST)
BPL 1$
RESET ;CLEAR PENDING INTERRUPT WITH RESET
JSR PC,WRPSW ;SET PRIORITY TO 5
.WORD 240
NOP ;GIVE TIME FOR ANY INTERRUPT
BIC #BIT6,@LKS ;DISALLOW INTERRUPTS
BR 3$ ;BR TO END OF TEST
2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 51 ;RESET DID NOT CLEAR INTERRUPT
3$: MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR

```

T21 TEST THAT RTC INTERRUPT CLEARS WITH RESET

```
1906
1907
1908
1909
1910
1911 006302 000004
1912 006304 005737 002336
1913 006310 001447
1914 006312 032777 000100 172520
1915 006320 001043
1916 006322 004737 012242
1917 006326 000340
1918 006330 017703 174054
1919 006334 012777 006412 174046
1920 006342 042777 000200 174036
1921 006350 052777 000100 174030
1922 006356 105777 174024
1923 006362 100375
1924 006364 042777 000200 174014
1925 006372 004737 012242
1926 006376 000240
1927 006400 000240
1928 006402 042777 000100 173776
1929 006410 000402
1930
1931
1932 006412 022626
1933 006414 104052
1934
1935 006416 010377 173766
1936 006422 004737 012242
1937 006426 000340

::*****
:*TEST 22 TEST THAT RTC INTERRUPT CLEARS BY CLEARING BIT7 OF LKS
:*****
TST22: SCOPE
TST CTSTFL ;IS CONSOLE UNDER TEST?
BEQ TST23 ;IF NOT, SKIP THIS TEST
BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
BNE TST23 ;IF YES, SKIP THIS TEST
JSR PC,WRPSW ;SET PRIORITY TO 7
.WORD 340
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV #2,@RTCVT ;POINT RTC VECTOR TO ERROR REPORT
BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG
BIS #BIT6,@LKS ;ENABLE CLOCK INTERRUPTS
1$: TSTB @LKS ;WAIT FOR DONE (INTERRUPT REQUEST)
BP 1$
BIC #BIT7,@LKS ;CLEAR DONE & INTERRUPT
JSR PC,WRPSW ;ALLOW INTERRUPTS
.WORD 240
NOP ;GIVE TIME FOR ANY INTERRUPT
BIC #BIT6,@LKS ;DISALLOW INTERRUPTS
BR 3$ ;BR TO END OF TEST

2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 52 ;CLEARING BIT7 OF LKS DID NOT CLEAR INTERRUPT

3$: MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR
JSR PC,WRPSW ;RESTORE PRIORITY TO 7
.WORD 340
```

TEST THAT RTC INTERRUPT CLEARS BY CLEARING BIT7 OF LKS

```
1938
1939
1940
1941      ;:*****
1942      ;*TEST 23      TEST CLOCK REPEATABILITY
1943      ;:*****
1943 006430 000004      TST23: SCOPE
1944 006432 005737 002336      TST      CTSTFL      ; IS CONSOLE UNDER TEST?
1945 006436 001462      BEQ      TST24      ; IF NOT, SKIP THIS TEST
1946 006440 032777 000100 172372      BIT      #BIT6,@SWR      ; ARE LINE CLOCK TESTS INHIBITED?
1947 006446 001056      BNE      TST24      ; IF YES, SKIP THIS TEST
1948 006450 042777 000100 173730      BIC      #BIT6,@LKS      ; DISALLOW INTERRUPTS
1949
1950 006456 005000      CLR      R0      ; CLEAR A TIMER
1951 006460 012701 177777      MOV      #-1,R1      ; SET A FLAG INDICATING FIRST PASS THRU THIS LOOP
1952 006464 005002      1$: CLR      R2      ; CLEAR CLOCK COUNTER
1953 006466 005077 173714      CLR      @LKS      ; CLEAR DONE
1954 006472 105777 173710      2$: TSTB     @LKS      ; SYNC ON DONE
1955 006476 100375      BPL      2$
1956 006500 005077 173702      CLR      @LKS      ; CLEAR DONE
1957 006504 105777 173676      3$: TSTB     @LKS      ; IS CLOCK DONE?
1958 006510 100003      BPL      4$      ; BR IF NOT , TO INCREMENT TIMER
1959 006512 005202      INC      R2      ; IF DONE, INCREMENT CLOCK COUNT
1960 006514 005077 173666      CLR      @LKS      ; CLEAR DONE
1961 006520 005200      4$: INC      R0      ; INCREMENT TIMER
1962 006522 001370      BNE      3$      ; BR IF TIME REMAINS
1963 006524 005201      INC      R1      ; INCREMENT LOOP PASS FLAG
1964 006526 001003      BNE      CMPARE     ; BR IF TWO PASSES HAVE BEEN MADE
1965 006530 010237 006600      MOV      R2,FIRST   ; IF NOT, STORE FIRST CLOCK COUNT
1966 006534 000753      BR       1$      ; DO LOOP AGAIN
1967 006536 013701 006600      CMPARE: MOV     FIRST,R1 ; RECALL FIRST CLOCK COUNT
1968 006542 160201      SUB      R2,R1      ; CALCULATE DIFFERENCE OF TWO COUNTS
1969 006544 100001      BPL      TOLER      ; IF POSITIVE,SKIP NEGATION OF DIFFERENCE
1970 006546 005401      NEG      R1      ; MAKE DIFFERENCE A POSITIVE NUMBER
1971 006550 020127 000001      TOLER: CMP     R1,#1  ; COMPARE DIFFERENCE WITH DESIRED TOLERANCE
1972 006554 003403      BLE      5$      ; BR, IF LOWER/EQUAL TO TOLERANCE
1973
1974 006556 010237 006602      MOV      R2,SECND   ; STORE SECOND COUNT
1975 006562 104053      ERROR   53      ; CLOCK REPEATABILITY ERROR
1976
1977 006564 032777 000020 172246 5$: BIT      #BIT4,@SWR   ; CLOCK TESTS ONLY?
1978 006572 001404      BEQ      TST24      ; BR IF NOT
1979 006574 000137 012102      JMP      $EOP      ; ELSE, JUMP TO END OF PASS ROUTINE
1980
1981 006600 000000      FIRST: .WORD 0
1982 006602 000000      SECND:  .WORD 0
```

```
1983
1984
1985
1986      :*****
1987      :*TEST 24      TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
1988      :*****
1988 006604 000004      TST24: SCOPE
1989 006606 042777 000100 173536      BIC      #BIT6,@TCSR      ;CLEAR TRANSMIT INTERRUPT ENABLE
1990 006614 017703 173542      MOV      @TVECT,R3      ;SAVE XMIT VECTOR
1991 006620 012777 006644 173534      MOV      #2$,@TVECT      ;POINT XMIT VECTOR TO ERROR REPORT
1992 006626 105777 173520      1$:      TSTB      @TCSR      ;WAIT FOR DONE
1993 006632 100375      BPL      1$
1994 006634 004737 012242      JSR      PC,WRPSW      ;SET PSW TO PRIORITY 3
1995 006640 000140      .WORD   140
1996 006642 000402      BR      3$
1997
1998 006644 022626      2$:      CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
1999 006646 104054      ERROR   54
2000
2001 006650 012777 006670 173504      3$:      MOV      #4$,@TVECT      ;XMIT INTERRUPTS WITH INTERRUPT ENABLE CLEAR
2002 006656 052777 000100 173466      BIS      #BIT6,@TCSR      ;SET XMIT VECTOR TO END OF TEST
2003 006664 000240      NOP
2004
2005 006666 104055      ERROR   55      ;XMIT DID NOT INTERRUPT
2006
2007 006670 042777 000100 173454      4$:      BIC      #BIT6,@TCSR      ;DISABLE INTERRUPTS
2008 006676 022626      CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
2009 006700 010377 173456      MOV      R3,@TVECT      ;RESTORE XMIT VECTOR
2010
```


TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED

```
2011
2012
2013      ;*****
2014      ;*TEST 25      TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED
2015      ;*****
2015 006704 000004      TST25: SCOPE
2016 006706 042777 000100 173436      BIC      #BIT6,@TCSR      ;DISABLE INTERRUPTS
2017 006714 004737 012242      JSR      PC,WRPSW      ;SET PSW TO PRIORITY 7
2018 006720 000340      .WORD    340
2019 006722 017703 173434      MOV      @TVECT,R3      ;SAVE XMIT VECTOR
2020 006726 012777 006754 173426      MOV      #2$,@TVECT      ;POINT XMIT VECTOR TO ERROR REPORT
2021 006734 105777 173412      1$:     TSTB     @TCSR      ;WAIT FOR DONE
2022 006740 100375      BPL      1$
2023 006742 052777 000100 173402      BIS      #BIT6,@TCSR      ;ENABLE INTERRUPT
2024 006750 000240      NOP
2025 006752 000402      BR       3$      ;CONTINUE TEST
2026
2027 006754 022626      2$:     CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
2028 006756 104056      ERROR   56
2029
2030 006760 042777 000100 173364      3$:     BIC      #BIT6,@TCSR      ;XMIT INTERRUPTS AT PRIORITY=7
2031 006766 012777 007006 173366      MOV      #4$,@TVECT      ;CLEAR INTERRUPT ENABLE
2032 006774 004737 012242      JSR      PC,WRPSW      ;POINT XMIT VECTOR TO ERROR REPORT
2033 007000 000140      .WORD    140      ;SET PSW TO PRIORITY 3
2034 007002 000240      NOP
2035 007004 000402      BR       5$      ;BR TO END OF TEST-NO INTERRUPT
2036
2037 007006 022626      4$:     CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
2038 007010 104057      ERROR   57
2039
2040 007012 010377 173344      5$:     MOV      R3,@TVECT      ;XMIT INTERRUPT OCCURES WITH BIT6 CLEAR
      ;RESTORE XMIT VECTOR
```

TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED

```
2041
2042
2043
2044      ;*****
2045      ;*TEST 26      TEST TRANSMITTER FOR DOUBLE INTERRUPTS
2046      ;*****
2046      TST26:  SCOPE
2047      007016 000004      BIC      #BIT6,@TCSR      ;CLEAR INTERRUPT ENABLE
2048      007020 042777 000100 173324      MOV      @TVECT,R3      ;SAVE XMIT VECTOR
2049      007026 017703 173330      MOV      @TPSW,R4      ;SAVE XMIT PSW VECTOR
2050      007032 017704 173326      MOV      #2$,@TVECT      ;SET UP XMIT VECTOR
2051      007036 012777 007100 173316      MOV      #340,@TPSW      ;SET PIO 7 AFTER INTERRUPT
2052      007044 012777 000340 173312      JSR      PC,WRPSW      ;SET PSW TO PRIORITY 3
2053      007052 004737 012242      .WORD   140
2054      007056 000140
2055      007060 105777 173266      1$:     TSTB   @TCSR      ;WAIT FOR DONE
2056      007064 100375      BPL
2057      007066 052777 000100 173256      BIS     #BIT6,@TCSR      ;ENABLE INTERRUPTS
2058      007074 000240      NOP
2059      007076 104060      ERROR  60
2060
2061      007100 022626      ;XMIT INTERRUPT DID NOT OCCUR
2062      007102 012777 007130 173252      2$:     CMP     (SP)+,(SP)+  ;RESTORE SP AFTER INTERRUPT
2063      007110 004737 012242      MOV     #4$,@TVECT      ;POINT XMIT VECTOR TO ERROR
2064      007114 000140      JSR     PC,WRPSW      ;SET PSW TO PRIORITY 3
2065      007116 000240      .WORD  140
2066      007120 042777 000100 173224      NOP     ;GIVE TIME FOR ANY INTERRUPTS
2067      007126 000402      BIC     #BIT6,@TCSR      ;DISABLE INTERRUPTS
2068      007130 022626      BR      5$             ;BR TO END OF TEST
2069      007132 104061      4$:     CMP     (SP)+,(SP)+  ;RESTORE SP AFTER INTERRUPT
2070      007134 010377 173222      ERROR  61
2071      007140 010477 173220      5$:     MOV     R3,@TVECT    ;XMIT RE-INTERRUPTED
2072      007144 010477 173220      MOV     R4,@TPSW      ;RESTORE XMIT VECTOR
2073      007146 010477 173220      ;RESTORE XMIT PSW VECTOR
2074
2075      ;*****
2076      ;*TEST 27      TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF
2077      ;*****
2078      TST27:  SCOPE
2079      007144 000004      BIC     #BIT6,@TCSR      ;DISABLE INTERRUPTS
2080      007146 042777 000100 173176      JSR     PC,WRPSW      ;SET PSW TO PRIORITY 7
2081      007154 004737 012242      .WORD  340
2082      007160 000340      MOV     @TVECT,R3      ;SAVE XMIT VECTOR
2083      007162 017703 173174      MOV     #2$,@TVECT      ;POINT XMIT VECTOR TO ERROR
2084      007166 012777 007240 173166      BIS     #BIT6,@TCSR      ;ENABLE INTERRUPTS
2085      007174 052777 000100 173150      CLR     @TBUF          ;LOAD TBUF
2086      007202 005077 173146      1$:     TSTB   @TCSR      ;WAIT FOR DONE (INTERRUPT)
2087      007206 105777 173140      BPL
2088      007212 100375      CLR     @TBUF          ;FILL SECOND BUFFER TO RESET INT.
2089      007214 005077 173134      JSR     PC,WRPSW      ;ALLOW INTERRUPTS
2090      007220 004737 012242      .WORD  140
2091      007224 000140      NOP
2092      007226 000240      ;GIVE TIME FOR ANY INTERRUPTS
2093      007230 042777 000100 173114      BIC     #BIT6,@TCSR      ;DISABLE INTERRUPTS
2094      007236 000402      BR      3$             ;BR TO END OF TEST
2095      007240 022626      2$:     CMP     (SP)+,(SP)+  ;RESTORE SP AFTER INTERRUPT
2096      007242 104062      ERROR  62
```

TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF

```
2097  
2098 007244 010377 173112 3$: MOV R3,@TVECT ;LOADING TBUF DID NOT CLEAR INTERRUPT.  
2099 ;RESTORE XMIT VECTOR
```

```
2100
2101
2102      ;:*****
2103      ;*TEST 30      TEST THAT RCVR ACTIVE (11) & DONE (?) SET & CLEAR PROPERLY
2104      ;:*****
2104 007250 000004      TST30: SCOPE
2105 007252 000005      RESET
2106 007254 052777 000004 173070      BIS #BIT2,@TCSR ;CLEAR EVERYTHING
2107 007262 005000      CLR RO ;SET MAINTENANCE WRAP
2108 007264 005077 173064      CLR @TBUF ;CLEAR A TIMER
2109 007270 032777 004000 173050 WACTV: BIT @TBUF ;LOAD TRANSMIT BUFFER
2110 007276 001006      BNE #BIT11,@RCSR ;TEST RCVR ACTIVE BIT
2111 007300 005200      INC 2$ ;BR IF SET
2112 007302 001372      BNE RO ;INCREMENT TIMER IF NOT SET
2113 007304 042777 000004 173040      WACTV WACTV ;CONTINUE WAIT IF TIME REMAINS
2114      BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
2115 007312 104063      ERROR 63 ;RCVR ACTIVE DID NOT SET WHILE RECEIVING
2116
2117 007314 000005      2$: RESET
2118 007316 032777 004000 173022      BIT #BIT11,@RCSR ;VERIFY "INIT" CLEARS RCV ACTIVE
2119 007324 001401      BEQ 3$
2120
2121 007326 104115      ERROR 115 ;INIT DID NOT CLEAR RCV ACTIVE
2122
2123 007330 005000      3$: CLR RO ;CLEAR A TIMER
2124 007332 052777 000004 173012      BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
2125 007340 062700 000000      WT: ADD #0,RO ;WAIT AT LEAST ONE BIT TIME
2126 007344 005200      INC RO
2127 007346 001374      BNE WT
2128 007350 033777 004000 172770      BIT BIT11,@RCSR ;VERIFY RCV ACTIVE STILL CLEAR
2129 007356 001404      BEQ 4$ ;BR IF CLEAR
2130
2131 007360 042777 000004 172764      BIC #BIT2,@TCSR ;CLEAR MAINTENANCE WRAP
2132 007366 104116      ERROR 116 ;RCV ACTIVE WITHOUT "START" BIT
2133
2134 007370 005000      4$: CLR RO ;CLEAR TIMER
2135 007372 005077 172756      CLR @TBUF ;LOAD TRANSMIT BUFFER
2136 007376 105777 172744      WDONE: TSTB @RCSR ;CHECK FOR RECEIVER DONE
2137 007402 100406      BMI 5$ ;BR, IF DONE
2138 007404 005200      INC RO ;INCREMENT TIMER, IF NOT DONE
2139 007406 001373      BNE WDONE ;CONTINUE WAIT IF TIME REMAINS
2140 007410 042777 000004 172734      BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
2141 007416 104064      ERROR 64
2142      ;RECEIVER DONE NEVER SET
2143
2144 007420 032777 004000 172720 5$: BIT #BIT11,@RCSR ;CHECK FOR RCVR ACTIVE CLEAR
2145 007426 001404      BEQ 6$ ;BR, IF CLEAR
2146 007430 042777 000004 172714      BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
2147 007436 104065      ERROR 65
2148      ;RCVR ACTIVE DID NOT CLEAR WITH RCVR DONE
2149
2150 007440 000005      6$: RESET ;CLEAR DONE WITH RESET
2151 007442 105777 172700      TSTB @RCSR ;CHECK FOR DONE CLEAR
2152 007446 001404      BEQ 7$
2153
2154 007450 042777 000004 172674      BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
2155 007456 104066      ERROR 66
```

TEST THAT RCVR ACTIVE (11) & DONE (7) SET & CLEAR PROPERLY

2156 ;RESET DID NOT CLEAR RCVR DONE
2157
2158 007460 042777 000004 172660 7S: BIC #BIT2,@RCR ;CLEAR MAINTENANCE BIT
2159 007466 000400 BR TST31 ;BR TO NEXT TEST

TEST THAT RCVR ACTIVE (11) & DONE (7) SET & CLEAR PROPERLY

2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200

```

*****
*TEST 31      TEST THAT RDR ENABLE CLEARS RECEIVER DONE FLAG
*****
TST31:  SCOPE
        RESET                ;CLEAR EVERYTHING
        BIS      #BIT0,@RCSR  ;SET RDR ENABLE
        BIS      #BIT2,@TCSR  ;SET MAINTENANCE WRAP
        CLR      @TBUF        ;LOAD TRANSMITTER
1$:     TSTB      @RCSR        ;WAIT FOR RECEIVER DONE
        BPL      1$
        BIT      #BIT0,@RCSR  ;VERIFY RCV ACTIVE CLEARED RDR ENABLE
        BEQ      2$          ;BR IF CLEAR
        ERROR    117        ;RDR ENABLE NOT CLEARED WITH RCV ACTIVE
2$:     BIS      #BIT0,@RCSR  ;CLEAR DONE BY SETTING RDR ENABLE
        TSTB      @RCSR        ;CHECK FOR DONE CLEAR
        BEQ      TST32       ;BR, IF CLEAR TO NEXT TEST
        BIC      #BIT2,@TCSR  ;CLEAR MAINTENANCE BIT
        ERROR    67
                                ;SETTING RDR ENABLE DID NOT CLEAR RCVR DONE

```

```

*****
*TEST 32      TEST THAT READING RBUF CLEARS RECEIVER DONE
*****
TST32:  SCOPE
        RESET                ;CLEAR EVERYTHING
        BIS      #BIT2,@TCSR  ;SET MAINTENANCE WRAP
        CLR      @TBUF        ;LOAD TRANSMITTER
1$:     TSTB      @RCSR        ;WAIT FOR RECEIVER DONE
        BPL      1$
        MOV      @RBUF,R0     ;READ RECEIVE BUFFER
        BIC      #BIT2,@TCSR  ;CLEAR MAINTENANCE BIT
        TSTB      @RCSR        ;CHECK FOR RECEIVE DONE CLEAR
        BEQ      TST33       ;BR, IF CLEAR TO NEXT TEST
        ERROR    70
                                ;READING RBUF DID NOT CLEAR RCVR DONE

```

```

2201
2202
2203
2204
2205
2206 007626 000004
2207 007630 042777 000100 172514
2208 007636 042777 000100 172502
2209 007644 052777 000004 172500
2210 007652 017703 172500
2211 007656 012777 007714 172472
2212 007664 004737 012242
2213 007670 000140
2214 007672 005077 172456
2215 007676 105777 172444 1$:
2216 007702 100375
2217 007704 042777 000004 172440
2218 007712 000405
2219
2220 007714 042777 000004 172430 2$:
2221 007722 022626
2222 007724 104071
2223
2224
2225 007726 012777 007754 172422 3$:
2226 007734 052777 000100 172404
2227 007742 000240
2228 007744 042777 000004 172400
2229 007752 104072
2230
2231
2232 007754 042777 000100 172364 4$:
2233 007762 042777 000004 172362
2234 007770 022626
2235 007772 010377 172360

```

```

*****
*TEST 33 TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
*****
TST33: SCOPE
      BIC #BIT6,@TCSR ;DISABLE TRANSMIT INTERRUPTS
      BIC #BIT6,@RCSR ;DISABLE RECEIVER INTERRUPTS
      BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
      MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
      MOV #2$,@RVECT ;POINT RCV VECTOR TO ERROR REPORT
      JSR PC,WRPSW ;SET PSW TO PRIORITY 3
           .WORD 140
      CLR @TBUF ;SEND A CHARACTER
      TSTB @RCSR ;WAIT FOR RECEIVER DONE
      BPL 1$
      BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
      BR 3$ ;CONTINUE TEST
2$: BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
    CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
    ERROR 71
;RECEIVER INTERRUPTS WITH INT. ENABLE CLEAR
3$: MOV #4$,@RVECT ;POINT RCV VECTOR TO END OF TEST
    BIS #BIT6,@RCSR ;ENABLE RCV INTERRUPTS
    NOP ;GIVE ANY INTERRUPTS TIME
    BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
    ERROR 72
;RCVR DID NOT INTERRUPT
4$: BIC #BIT6,@RCSR ;DISABLE INTERRUPTS
    BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
    CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
    MOV R3,@RVECT ;RESTORE RECEIVE VECTOR

```

TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED

```
2236
2237
2238
2239      ;*****
2240      ;*TEST 34      TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED
2241      ;*****
2241 007776 000004
2242 010000 000005
2243 010002 004737 012242
2244 010006 000340
2245 010010 017703 172342
2246 010014 012777 010054 172334
2247 010022 052777 000004 172322
2248 010030 005077 172320
2249 010034 105777 172306 1$:
2250 010040 100375
2251 010042 052777 000100 172276
2252 010050 000240
2253 010052 000405
2254 010054 042777 000004 172270 2$:
2255 010062 022626
2256 010064 104073
2257
2258
2259 010066 042777 000100 172252 3$:
2260 010074 012777 010122 172254
2261 010102 004737 012242
2262 010106 000140
2263 010110 000240
2264 010112 042777 000004 172232
2265 010120 000405
2266
2267 010122 042777 000004 172222 4$:
2268 010130 022626
2269 010132 104074
2270
2271 010134 010377 172216 5$:
```

SCOPE
RESET ;CLEAR EVERYTHING
JSR PC,WRPSW ;SET PSW TO PRIORITY 7
.WORD 340
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
MOV #2,@RVECT ;POINT RCVR VECTOR TO ERROR REPORT
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
CLR @TBUF ;SEND A CHARACTER
TSTB @RCSR ;WAIT FOR RECEIVER DONE
BPL 1\$
BIS #BIT6,@RCSR ;ENABLE INTERRUPTS
NOP ;GIVE TIME FOR INTERRUPT
BR 3\$;CONTINUE TEST
BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 73
;RCVR INTERRUPTS AT PRIORITY 7
BIC #BIT6,@RCSR ;CLEAR INTERRUPT ENABLE
MOV #4,@RVECT ;POINT RCVR VECTOR TO ERROR REPORT
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
NOP ;GIVE TIME FOR ANY INTERRUPT
BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
BR 5\$;BR TO END OF TEST, IF NO INTERRUPT
BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 74
;RCVR INTERRUPT REQUEST PASSED WITH BIT6 CLEAR
MOV R3,@RVECT ;RESTORE RECEIVE VECTOR

TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED

2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309

010140 000004
010142 000005
010144 017703 172206
010150 017704 172204
010154 012777 010236 172174
010162 012777 000340 172170
010170 004737 012242
010174 000140
010176 052777 000004 172146
010204 005077 172144
010210 105777 172132
010214 100375
010216 042777 000004 172126
010224 052777 000100 172114
010232 000240
010234 104075
010236 022626
010240 012777 010276 172110
010246 004737 012242
010252 000140
010254 000240
010256 042777 000100 172062
010264 010377 172066
010270 010477 172064
010274 000402
010276 022626
010300 104076
010302 010377 172050

```
*****  
*TEST 35 TEST RECEIVER FOR DOUBLE INTERRUPTS  
*****  
TST35: SCOPE  
RESET ;CLEAR EVERYTHING  
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR  
MOV @RPSW,R4 ;SAVE RECEIVE PSW VECTOR  
MOV #2,@RVECT ;POINT RCV VECTOR TO CONTINUE TEST  
MOV #340,@RPSW ;SET PRIORITY TO 7 AFTER INTERRUPT  
JSR PC,WRPSW ;SET PSW TO PRIORITY 3  
 .WORD 140  
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP  
CLR @TBUF ;SEND A CHARACTER  
1$: TSTB @RCSR ;WAIT FOR RCVR DONE  
BPL 1$  
BIC #BIT2,@TCSR ;CLEAR MAINTENANCE BIT  
BIS #BIT6,@RCSR ;ENABLE RCV INTERRUPTS  
NOP ;GIVE SOME TIME  
ERROR 75  
 ;RCVR INTERRUPT DID NOT OCCUR  
2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
MOV #3,@RVECT ;POINT RCV VECTOR TO ERROR REPORT  
JSR PC,WRPSW ;RESET PSW TO PRIORITY 3  
 .WORD #140  
NOP ;GIVE SOME TIME  
BIC #BIT6,@RCSR ;CLEAR INTERRUPT ENABLE  
MOV R3,@RVECT ;RESTORE RECEIVE VECTOR  
MOV R4,@RPSW ;RESTORE RECEIVE PSW VECTOR  
BR 4$ ;BR TO END OF TEST  
3$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
ERROR 76  
 ;RECEIVER RE-INTERRUPTED  
4$: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR
```

T35 TEST RECEIVER FOR DOUBLE INTERRUPTS

```
2310
2311
2312
2313      ;:*****
2314      ;*TEST 36      TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF
2315      ;:*****
2315 010306 000004      TST36:  SCOPE
2316 010310 000005      RESET      ;CLEAR EVERYTHING
2317 010312 004737 012242      JSR      PC,WRPSW      ;SET PSW PRIORITY TO 7
2318 010316 000340      .WORD 340
2319 010320 017703 172032      MOV      @RVECT,R3      ;SAVE RECEIVE VECTOR
2320 010324 012777 010412 172024      MOV      #2$,@RVECT      ;POINT RCV VECTOR TO ERROR REPORT
2321 010332 052777 000100 172006      BIS      #BIT6,@RCSR      ;SET RCVR INTERRUPT ENABLE
2322 010340 052777 000004 1720C4      BIS      #BIT2,@TCSR      ;SET MAINTENANCE WRAP
2323 010346 005077 172002      CLR      @TBUF      ;SEND A CHARACTER
2324 010352 105777 171770      1$:  TSTB      @RCSR      ;WAIT FOR DONE (INTERRUPT)
2325 010356 100375      BPL      1$
2326 010360 042777 000004 171764      BIC      #BIT2,@TCSR      ;CLEAR MAINTENANCE BIT
2327 010366 005077 171756      CLR      @RBUF      ;READ RBUF TO CLEAR PENDING INTERRUPT
2328 010372 004737 012242      JSR      PC,WRPSW      ;SET PSW TO PRIORITY 3
2329 010376 000140      .WORD 140
2330 010400 000240      NOP
2331 010402 042777 000100 171736      BIC      #BIT6,@RCSR      ;NO INTERRUPT-CLEAR INT. ENABLE
2332 010410 000402      BR      3$
2333
2334 010412 022626      2$:  CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
2335 010414 104077      ERROR 77      ;READING RBUF DID NOT CLEAR INTERRUPT
2336
2337 010416 010377 171734      3$:  MOV      R3,@RVECT      ;RESTORE RECEIVE VECTOR
2338
```

TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF

```
2339
2340
2341
2342
2343
2344 010422 000004
2345 010424 000005
2346 010426 004737 012242
2347 010432 000340
2348 010434 017703 171716
2349 010440 012777 010520 171710
2350 010446 052777 000100 171672
2351 010454 052777 000004 171670
2352 010462 012777 000377 171664
2353 010470 105777 171652 1$:
2354 010474 100375
2355 010476 000005
2356 010500 004737 012242
2357 010504 000140
2358 010506 000240
2359 010510 042777 000100 171630
2360 010516 000402
2361
2362
2363 010520 022626 2$:
2364 010522 104100
2365
2366 010524 010377 171626 3$:

;*****
;*TEST 37 TEST THAT RESET CLEARS RECEIVE INTERRUPT
;*****
TST37: SCOPE
RESET ;CLEAR EVERYTHING
JSR PC,WRPSW ;SET PSW TO PRIORITY 7
.WORD 340
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
MOV #2$,@RVECT ;POINT RCV VECTOR TO ERROR REPORT
BIS #BIT6,@RCSR ;SET RCV INTERRUPT ENABLE
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
MOV #377,@TBUF ;SEND AN ALL 1'S CHARACTER
1$: TSTB @RCSR ;WAIT FOR RCV DONE
BPL 1$
RESET ;CLEAR RCV INTERRUPT & RBUF
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
NOP ;ALLOW TIME FOR AN ERRONEOUS INTERRUPT
BIC #BIT6,@RCSR ;NO INTERRUPT-CLEAR INT. ENABLE
BR 3$ ;CONTINUE TEST

2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR 100 ;RESET DID NOT CLEAR RCVR INTERRUPT

3$: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR
```

T37 TEST THAT RESET CLEARS RECEIVE INTERRUPT

```
2367
2368
2369
2370          ;:*****
          ;*TEST 40      TEST THAT THE 'OR' ERROR (BIT14) & 'ERROR' (BIT15) CAN BE SET
          ;:*****
2371          TST40:  SCOPE
2372 010530 000004          BIT      #BIT10,@SWR      ;IS THIS TEST ENABLED
2373 010532 032777 002000 170300          BEQ      TST41      ;IF NOT ENABLED, BR TO NEXT TEST
2374 010540 001432          RESET          ;CLEAR EVERYTHING
2375 010542 000005          BIS      #BIT2,@TCSR      ;SET MAINTENANCE WRAP
2376 010544 052777 000004 171600          MOV      #3,R0      ;SET CHARACTER COUNT TO SEND 3 CHAR.
2377 010552 012700 000003          CLR      @TBUF      ;LOAD TRANSMIT BUFFER
2378 010556 005077 171572          1$:  TSTB   @TCSR      ;WAIT FOR TRANSMIT DONE
2379 010562 105777 171564          BPL      2$
2380 010566 100375          DEC      R0          ;DECREMENT CHARACTER COUNT
2381 010570 005300          BNE     1$          ;BR IF ALL CHARACTERS NOT TRANSMITTED
2382 010572 001371          BIC     #BIT2,@TCSR ;CLEAR MAINTENANCE BIT
2383 010574 042777 000004 171550          BIT     #BIT14,@RBUF ;TEST FOR 'OR' ERROR FLAG
2384 010602 032777 040000 171540          BNE     3$          ;BR, IF SET
2385 010610 001001          ERROR  101
2386 010612 104101          ;'OR' ERROR FLAG DID NOT SET
2387
2388
2389 010614 032777 100000 171526 3$:  BIT     #BIT15,@RBUF ;TEST 'ERROR' FLAG
2390 010622 001001          BNE     4$          ;BR, IF SET
2391 010624 104102          ERROR  102
2392
2393 010626          4$:  ;'ERROR' FLAG DID NOT SET WITH 'OR' FLAG
```

T40 TEST THAT THE 'OR' ERROR (BIT14) & 'ERROR' (BIT15) CAN BE SET

```
2394
2395
2396
2397
2398
2399 010626 000004
2400 010630 032777 000400 170202
2401 010636 001444
2402 010640 000005
2403 010642 052777 000004 171502
2404 010650 012777 177777 171476
2405 010656 105777 171464 1$:
2406 010662 100375
2407 010664 005077 171460
2408 010670 052777 000001 171454
2409 010676 005000
2410 010700 117737 171442 001026 2$:
2411 010706 100406
2412 010710 005200
2413 010712 001372
2414
2415 010714 042777 000005 171430
2416 010722 104103
2417
2418 010724 105777 171420
2419 010730 001404
2420 010732 042777 000005 171412
2421
2422 010740 104103
2423
2424 010742 042777 000005 171402 3$:
```

```

*****
*TEST 41 TEST THAT BREAK TRANSMITS ALL ZEROES
*****
TST41: SCOPE
BIT #BIT8,@SWR ;IS BREAK FUNCTION TEST ENABLED?
BEQ TST42 ;BR TO NEXT TEST, IF NOI ENABLED
RESET ;CLEAR EVERYTHING
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
MOV #-1,@TBUF ;TRANSMIT ALL ONES TO RCVR
TSTB @RCSR ;WAIT FOR RCVR DONE
BPL 1$
CLR @RBUF ;CLEAR DONE (LEAVING ALL ONES IN RBUF)
BIS #BIT0,@TCSR ;TRANSMIT BREAK
CLR R0 ;CLEAR A TIMER
MOVB @RCSR,$BDDAT ;WAIT FOR RCVR DONE
BMI CONT41 ;BR IF DONE
INC R0 ;IF NOT, INCREMENT TIMER
BNE 2$ ;BR IF TIME REMAINS

BIC #BIT0!BIT2,@TCSR ;CLEAR MAINTENANCE & BREAK BITS
ERROR 103 ;BREAK DID NOT TRANSMIT ANYTHING

CONT41: TSTB @RBUF ;CHECK RECEIVE BUFFER FOR ZERO
BEQ 3$ ;BR, IF ZERO
BIC #BIT0!BIT2,@TCSR ;CLEAR MAINTENANCE & BREAK BITS

ERROR 103 ;BREAK DID NOT TRANSMIT ALL ZEROES

BIC #BIT0.BIT2,@TCSR ;CLEAR MAINTENANCE & BREAK BITS
```

```
2425
2426
2427
2428
2429
2430 010750 000004
2431 010752 032777 002000 170060
2432 010760 001435
2433 010762 032777 000400 170050
2434 010770 001431
2435 010772 000005
2436 010774 052777 000004 171350
2437 011002 052777 000001 171342
2438 011010 005077 171340
2439 011014 105777 171326 1$:
2440 011020 100375
2441 011022 042777 000005 171322
2442 011030 032777 020000 171312
2443 011036 001001
2444
2445 011040 104104
2446
2447 011042 032777 100000 171300 2$:
2448 011050 001001
2449
2450 011052 104114
2451
2452 011054 3$:
```

```
*****
*TEST 42 TEST THAT 'FR' ERROR CAN BE SET DURING BREAK
*****
TST42: SCOPE
BIT #BIT10,@SWR ;IS THE 'TEST ERROR FLAGS' BIT SET
BEQ TST43 ;BR TO NEXT TEST, IF NOT SET
BIT #BIT8,@SWR ;IS BREAK FUNCTION ENABLED
BEQ TST43 ;BR TO NEXT TEST, IF NOT SET
RESET ;CLEAR EVERYTHING
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
BIS #BIT0,@TCSR ;SEND BREAK
CLR @TBUF ;TRANSMIT A CHARACTER TO TIME BREAK
TSTB @RCSR ;WAIT FOR RCVR DONE
BPL 1$
BIC #BIT0!BIT2,@TCSR ;CLEAR MAINTENANCE & BREAK BITS
BIT #BIT13,@RBUF ;CHECK FOR FRAMING ERROR FLAG
BNE 2$ ;BR, IF SET

ERROR 104
;BREAK DID NOT SET FRAMING ERROR
;TEST 'ERROR' FLAG
;BR, IF SET

ERROR 114
;'ERROR' FLAG DID NOT SET WITH 'OR' FLAG
```

T42 TEST THAT 'FR' ERROR CAN BE SET DURING BREAK

```
2453
2454
2455      ;*****
2456      ;*TEST 43      TEST DATA PATH FROM TRANSMITTER TO RECEIVER USING MAINTENANCE WRAP
2457      ;*****
2458 011054 000004      TST43: SCOPE
2459
2460 011056 000005      RESET      ;CLEAR EVERYTHING
2461 011060 005001      CLR      R1      ;CLEAR REGISTER FOR TEST DATA
2462 011062 052777 000004 171262      BIS      #BIT2,@TCSR      ;SET MAINTENANCE WRAP
2463      ;TRANSMIT A BINARY COUNT PATTERN - UP
2464      ;TO THE BIT POSITION INDICATED BY THE
2465      ;CONTENTS OF LOCATION '$USW '
2466 011070 005201      1$:      INC      R1      ;INCREMENT THE TEST DATA
2467 011072 010177 171256      MOV      R1,@TBUF      ;XMIT A CHARACTER
2468 011076 105777 171244      2$:      TSTB     @RCSR      ;WAIT FOR RECEIVER DONE
2469 011102 100375      BPL      2$
2470 011104 017702 171240      MOV      @RBUF,R2      ;GET RECEIVED CHARACTER
2471 011110 043701 001112      BIC      @#$USWR,R1      ;CLEAR LOWEST UNUSED DATA BIT POSITITON IN TEST DATA
2472 011114 020102      CMP      R1,R2      ;COMPARE DATA
2473 011116 001003      BNE      3$      ;BR, IF NON-COMPARE
2474 011120 105701      TSTB     R1      ;TEST XMIT DATA FOR ZERO
2475 011122 001411      BEQ      4$      ;BR, IF FINISHED
2476 011124 000761      BR       1$      ;CONTINUE IF NOT
2477 011126 010137 001024      3$:      MOV      R1,$GDDAT      ;STORE THE EXPECTED DATA
2478 011132 010237 001026      MOV      R2,$BDDAT      ;STORE RECEIVED DATA
2479 011136 042777 000004 171206      BIC      #BIT2,@TCSR      ;CLEAR MAINTENANCE BIT
2480 011144 104105      ERROR    105      ;DATA COMPARE DATA
2481
2482 011146 042777 000004 171176 4$:      BIC      #BIT2,@TCSR      ;CLEAR MAINTENANCE BIT
2483
2484
2485
2486      ;*****
2487      ;*TEST 44      TEST DATA PATHS USING WRAP CABLE
2488      ;*****
2489 011154 000004      TST44: SCOPE
2490 011156 032777 000200 167654      BIT      #BIT7,@SWR      ;IS THIS TEST ENABLED
2491 011164 001432      BEQ      TST45      ;BR, IF NOT
2492 011166 000005      RESET
2493 011170 005001      CLR      R1      ;CLEAR EVERYTHING
2494      ;CLEAR REGISTER FOR TEST DATA
2495      ;TRANSMIT A BINARY COUNT PATTERN - UP
2496      ;TO THE BIT POSITION INDICATED BY THE
2497      ;CONTENTS OF LOCATION '$USWR'
2497 011172 105201      1$:      INCB     R1      ;INCREMENT THE TEST DATA
2498 011174 010177 171154      MOV      R1,@TBUF      ;XMIT A CHARACTER
2499 011200 005000      CLR      R0      ;CLEAR A TIMER
2500 011202 105777 171140      2$:      TSTB     @RCSR      ;WAIT FOR RECEIVER DONE
2501 011206 100403      BMI      3$      ;BR IF DONE
2502 011210 005200      INC      R0      ;INCREMENT TIMER IF NOT
2503 011212 001373      BNE      2$      ;BR IF TIME RFMAINS
2504
2505 011214 104064      ERROR    64      ;RECEIVER DONE NOT SET
2506
2507 011216 017702 171126      3$:      MOV      @RBUF,R2      ;GET RECEIVED CHARACTER
2508 011222 043701 001112      BIC      @#$USWR,R1      ;CLEAR LOWEST UNUSED DATA BIT POSITITON IN TEST DATA
```

TEST DATA PATHS USING WRAP CABLE

```
2509 011226 020102      CMP      R1,R2      ;COMPARE DATA
2510 011230 001003      BNE      4$         ;BR, IF NON-COMPARE
2511 011232 105701      TSTB     R1         ;TEST XMIT DATA FOR ZERO
2512 011234 001406      BEQ      TST45     ;BR, IF FINISHED
2513 011236 000755      BR       1$         ;CONTINUE IF NOT
2514 011240 010137 001024 4$:      MOV      R1,$GDDAT ;STORE EXPECTED DATA
2515 011244 010237 001026      MOV      R2,$BDDAT ;STORE RECEIVED DATA
2516
2517 011250 104106      ERROR   106       ;DATA COMPARE ERROR WITH WRAP CABLE
```


TEST DATA PATHS USING WRAP CABLE

```
2518
2519
2520
2521      ;:*****
;*TEST 45      TEST DL11-W LOGIC BY EXERCISING THE XMIT,RECEIVE, & CLOCK(IF AVAILABLE)
2522      ;:*****
2523 011252 000004      TST45: SCOPE
2524 011254 000005      RESET          ;CLEAR EVERYTHING
2525 011256 004737 012242      JSR      PC,WRPSW      ;SET PRIORITY TO 7
2526 011262 000340      .WORD      340
2527 011264 017703 171072      MOV      @TVECT,R3      ;SAVE XMIT VECTOR
2528 011270 017704 171062      MOV      @RVECT,R4      ;SAVE RECEIVE VECTOR
2529 011274 017705 171110      MOV      @RTCVT,R5      ;SAVE CLOCK VECTOR
2530 011300 017737 171060 011734      MOV      @TPSW,STPSW     ;SAVE XMIT PSW VECTOR
2531 011306 017737 171046 011736      MOV      @RPSW,SRPSW     ;SAVE RECEIVE PSW VECTOR
2532 011314 017737 171072 011740      MOV      @RTCPSW,SCPSW   ;SAVE CLOCK PSW VECTOR
2533 011322 012777 011650 171032      MOV      #XMIT,@TVECT   ;POINT TRANSMIT VECTOR TO TRANSMIT ROUTINE
2534 011330 005077 171030      CLR      @TPSW          ;ALLOW INTERRUPTS AFTER XMIT INTERRUPT
2535 011334 012777 011704 171014      MOV      #RCV,@RVECT    ;POINT RECEIVE VECTOR TO RECEIVE ROUTINE
2536 011342 005077 171012      CLR      @RPSW          ;ALLOW INTERRUPTS AFTER RCVR INTERRUPT
2537 011346 005737 002336      TST      CTSTFL         ;IS CONSOLE UNDER TEST?
2538 011352 001414      BEQ      1$             ;IF NOT SKIP CLOCK SET UP
2539 011354 032777 000100 167456      BIT      #BIT6,@SWR     ;IF YES, ARE CLOCK TEST DISABLED?
2540 011362 001010      BNE      1$             ;IF YES, SKIP CLOCK SET UP
2541 011364 012777 011720 171016      MOV      #CLK,@RTCVT    ;POINT VECTOR TO CLOCK INTERRUPT ROUTINE
2542 011372 005077 171014      CLR      @RTCPSW        ;ALLOW INTERRUPTS AFTER CLOCK INTERRUPT
2543 011376 052777 000100 171002      BIS      #BIT6,@LKS     ;ENABLE CLOCK INTERRUPTS
2544 011404 052777 000004 170740 1$: BIS      #BIT2,@TCSR     ;SET MAINTENANCE WRAP
2545 011412 052777 000100 170732      BIS      #BIT6,@TCSR    ;ENABLE TRANSMIT INTERRUPTS
2546 011420 052777 000100 170720      BIS      #BIT6,@RCR     ;ENABLE RECEIVE INTERRUPTS
2547 011426 005037 011730      CLR      XMTCNT         ;CLEAR XMIT INTERRUPT COUNTER
2548 011432 005037 011726      CLR      RCVCNT         ;CLEAR RCV INTERRUPT COUNTER
2549 011436 005001      CLR      R1             ;CLEAR A REGISTER FOR TEST DATA USE
2550 011440 005000      CLR      R0             ;CLEAR TIMER
2551 011442 012702 011742      MOV      #BUF,R2        ;POINT R2 TO RECEIVE DATA STORAGE
2552 011446 005077 170702      CLR      @TBUF          ;SEND FIRST CHARACTER
2553 011452 004737 012242      JSR      PC,WRPSW        ;SET PSW TO PRIORITY 3
2554 011456 000140      .WORD      140
2555
2556 011460 062700 000000      2$: ADD      #0,R0         ;WAIT FOR INTERRUPTS
2557 011464 062700 000001      ADD      #1,R0         ;ADD INSTRUCTIONS ARE USED TO LENGTHEN LOOP TIME
2558 011470 001373      BNE      2$            ; TO COVER THE SLOWEST BAUD RATE ON THE FASTEST CPU
2559 011472 032777 000100 170652      BIT      #BIT6,@TCSR    ;FINISHED ENTIRE TRANSMISSION
2560 011500 001402      BEQ      3$            ;BR, IF INTERRUPTS ARE DISABLED (FINISHED)
2561 011502 000005      RESET          ;CLEAR EVERYTHING
2562
2563 011504 104107      ERROR     107          ;TRANSMIT INTERRUPT TIMEOUT IN MAIN. DATA TEST
2564
2565 011506 023737 011730 011726 3$: CMP      XMTCNT,RCVCNT   ;COMPARE THE NUMBER OF INTERRUPTS
2566 011514 001402      BEQ      4$            ;BR, IF EQUAL
2567 011516 000005      RESET          ;CLEAR EVERYTHING
2568
2569 011520 104110      ERROR     110          ;RECEIVER DID NOT GET FULL TRANSMISSION
2570
2571      ; IF RCVCNT=0, NO DATA RECEIVED
2572      ; IF RCVCNT=?, THEN (XMTCNT-RCVCNT)
2573      ; EQUALS THE NO. OF INTERRUPTS LOST.
```

```
2574
2575
2576 011522 005737 002336      4$:  TST      CTSTFL      ;IS CONSOLE UNDER TEST?
2577 011526 001410              BEQ      5$          ;IF NOT, SKIP CLOCK COUNT CHECK
2578 011530 032777 000100 167302  BIT      #BIT6,@SWR   ;IF YES, ARE CLOCK TESTS DISABLED?
2579 011536 001004              BNE      5$          ;IF YES, SKIP CLOCK COUNT CHECK
2580 011540 005737 011732      TST      CLKCNT      ;CHECK FOR AT LESST ONE CLOCK INTERRUPT
2581 011544 001001              BNE      5$          ;BR IF INTERRUPTS OCCURRED
2582
2583 011546 104113              ERROR    113         ;NO CLOCK INTERRUPTS IN EXERCISER
2584
2585 011550 000005      5$:  RESET          ;CLEAR EVERYTHING
2586 011552 012700 011742      MOV      #BUF,R0     ;LOAD RECEIVED DATA POINTER TO R0
2587 011556 005001              CLR      R1          ;SET UP REGISTER FOR COMPARISON
2588 011560 022001      COMP:  CMP      (R0)+,R1 ;COMPARE XMIT & RCV DATA
2589 011562 001005              BNE      6$          ;BR, IF NOT EQUAL
2590 011564 105201              INCB    R1          ;INCREMENT COMPARE DATA
2591 011566 032701 000040      BIT      #BIT5,R1   ;FINISHED CHECKING RECEIVED DATA?
2592 011572 001772              BEQ      COMP        ;BR, IF NOT FINISHED
2593 011574 000405              BR      7$          ;BR TO END OF TEST
2594
2595 011576 014037 001026      6$:  MOV      -(R0),%BDDAT ;STORE BAD DATA FOR ERROR REPORT
2596 011602 010137 001024      MOV      R1,%GDDAT  ;STORE GOOD DATA FOR ERROR REPORT
2597 011606 104111              ERROR    111         ;DATA COMPARE ERROR IN EXERCISER
2598
2599 011610 010377 170546      7$:  MOV      R3,@TVECT  ;RESTORE XMIT VECTOR
2600 011614 010477 170536      MOV      R4,@RVECT  ;RESTORE RECEIVE VECTOR
2601 011620 010577 170564      MOV      R5,@RTCVT  ;RESTORE CLOCK VECTOR
2602 011624 013777 011734 170532  MOV      STPSW,@TPSW ;RESTORE XMIT PSW VECTOR
2603 011632 013777 011736 170520  MOV      SRPSW,@RPSW ;RESTORE RECEIVE PSW VECTOR
2604 011640 013777 011740 170544  MOV      SCPSW,@RTCPSW ;RESTORE CLOCK PSW VECTOR
2605 011646 000501              BR      ENDEV       ;BR TO END OF DEVICE TEST ROUTINE
2606
2607 011650 005237 011730      XMIT:  INC      XMITCNT ;INCREMENT XMIT INTERRUPT COUNTER
2608 011654 105201              INCB    R1          ;INCREMENT TEST DATA
2609 011656 032701 000040      BIT      #BIT5,R1   ;SEND DATA PATTERN FROM 00 --> 37
2610 011662 001404              BEQ      XCONT      ;BR, IF MORE DATA TO BE SENT
2611 011664 042777 000100 170460  BIC      #BIT6,@TCSR ;CLEAR XMIT INTERRUPT ENABLE
2612 011672 000402              BR      XRET        ;RETURN, WITHOUT SENDING ANY MORE DATA
2613 011674 110177 170454      XCONT:  MOV      R1,@TBUF ;SEND NEW CHARACTER
2614 011700 005000      XRET:  CLR      R0    ;CLEAR TIMER
2615 011702 000002              RTI          ;RETURN
2616
2617 011704 017722 170440      RCV:   MOV      @RBUF,(R2)+ ;STORE RECEIVED DATA
2618 011710 005237 011726      INC      RCVCNT     ;INCREMENT RCV INTERRUPT COUNTER
2619 011714 005000              CLR      R0        ;CLEAR TIMER
2620 011716 000002              RTI          ;RETURN
2621
2622 011720 005237 011732      CLK:   INC      CLKCNT ;INCREMENT CLOCK INTERRUPT COUNT
2623 011724 000002              RTI          ;RETURN
2624
2625 011726 000000      RCVCNT: .WORD    0
2626 011730 000000      XMITCNT: .WORD   0
2627 011732 000000      CLKCNT:  .WORD   0
2628 011734 000000      STPSW:  .WORD   0
2629 011736 000000      SRPSW:  .WORD   0
```

.MAIN. MACY11 30A(1052) 19-APR-78 11:44 PAGE 66
CZDLDC.P11 19-APR-78 11:41 T45

TEST DL11-W LOGIC BY EXERCISING THE XMIT, RECEIVE, & CLOCK (IF AVAILABLE) SIMULTAN
SEQ 0066

2630 011740 000000
2631 011742 000044
2632

SCPSW: .WORD 0
BUF: .BLKW 44

```
2633  
2634 ;END OF DEVICE PASS ROUTINE  
2635  
2636 012052 005037 001002 ENDEV: CLR $STSTM ;CLEAR TEST NO. COUNT FOR SCOPE ROUTINE  
2637 012056 005237 001076 INC $DEVCT ;INCREMENT DEVICE COUNTER  
2638 012062 023737 002342 001076 CMP TMP2,$DEVCT ;ALL DEVICES TESTED  
2639 012070 001404 BEQ $EOP ;BR, IF YES  
2640 012072 005037 002336 CLR CTSTFL ;CLEAR CONSOLE UNDER TEST FLAG  
2641 012076 000137 003500 JMP TSTDEV ;GO TEST NEXT DEVICE
```

.SBTTL END OF PASS ROUTINE

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

```
2651  
2652 012102 $EOP:  
2653 012102 000004 SCOPE  
2654 012104 005037 001002 CLR $STSTM ;;ZERO THE TEST NUMBER  
2655 012110 005237 001074 INC $PASS ;;INCREMENT THE PASS NUMBER  
2656 012114 042737 100000 001074 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER  
2657 012122 005327 DEC (PC)+ ;;LOOP?  
2658 012124 000001 $EOPCT: .WORD 1  
2659 012126 003015 BGT $DOAGN ;;YES  
2660 012130 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER  
2661 012132 000001 $ENDCT: .WORD 1  
2662 012134 012124 $EOPCT  
2663 012136 104401 012172 TYPE ,ENDMG ;;TYPE 'END PASS'  
2664 012142 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS  
2665 012146 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR  
2666 012150 000005 RESET ;;CLEAR THE WORLD  
2667 012152 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR  
2668 012154 000240 NOP ;;SAVE ROOM  
2669 012156 000240 NOP ;;FOR  
2670 012160 000240 NOP ;;ACT11  
2671 012162 $DOAGN:  
2672 012162 000137 JMP @(PC)+ ;;RETURN  
2673 012164 012206 $RTNAD: .WORD GOAGIN  
2674 012166 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING  
2675 012172 .EVEN  
2676 012172 005015 047105 020104 ENDMG: .ASCIZ <CR><LF>/END PASS /  
2677 012200 040520 051523 000040
```

```
2678
2679 012206 005037 001076      GOAGIN: CLR      $DEVCT      ;CLEAR DEVICE COUNT
2680 012212 022737 000001 002342      CMP      #1,TMP2      ;IS THERE ONLY ONE DEVICE UNDER TEST?
2681 012220 001004              BNE      RSTRT        ;BR, IF NOT
2682 012222 012706 001000              MOV      #1000,SP     ;RESET STACK POINTER
2683 012226 000137 003622              JMP      TST1         ;GO DO ANOTHER PASS
2684
2685 012232 005037 001100      RSTRT: CLR      $UNIT      ;CLEAR UNIT NUMBER
2686 012236 000137 003424              JMP      BEGIN
2687
2688 012242 011646      WRPSW: MOV(SP),-(SP)      ;COPY RETURN PC
2689 012244 013616              MOV      @ (SP)+,(SP) ;MOVE NEW PSW TO STACK
2690 012246 062746 000002              ADD      #2,-(SP)     ;ADJUST JSR RETURN
2691 012252 000002              RTI                  ;POP RETURN PC & NEW PSW
2692
2693      ;SUBROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
2694
2695 012254 012600      CATCH: MOV      (SP)+,R0    ;GET ADDRESS OF TRAP VECTOR + 4
2696 012256 162700 000004              SUB      #4,R0        ;ADJUST TO POINT TO TRAP ADDRESS
2697 012262 010037 012302              MOV      R0,BDVECT    ;STORE TRAP OR INTERRUPT ADDRESS
2698 012266 016637 000002 012300              MOV      2(SP),OLDPC  ;GET PC WHERE TRAP OR INTERRUPT OCCURRED
2699 012274 104112              ERROR   112          ;REPORT ERROR
2700
2701 012276 000000              HALT                    ;PROGRAM MUST BE RESTARTED AT THIS POINT
2702 012300 000000      OLDPC: .WORD   0
2703 012302 000000      BDVECT: .WORD  0
2704
```

```
2705  
2706  
2707  
2708  
2709  
2710  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718 012304 $ERROR:  
2719 012304 105237 001003 7$: INCB $ERFLG ;SET THE ERROR FLAG  
2720 012310 001775 BEQ 7$ ;DON'T LET FLAG GO TO ZERO  
2721 012312 013777 001002 166522 MOV $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG  
2722 012320 005237 001012 INC $ERTTL ;INCREMENT ERROR COUNT  
2723 012324 011637 001016 MOV (SP),$ERRPC ;GET ADDRESS OF ERROR INSTRUCTION  
2724 012330 162737 000002 001016 SUB #2,$ERRPC  
2725 012336 117737 166454 001014 MOVB @$ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE  
2726 012344 032777 020000 166466 BIT #BIT13,@SWR ;SKIP TYPEOUT IF SET  
2727 012352 001004 BNE 20$ ;SKIP TYPEOUTS  
2728 012354 004737 012466 JSR PC,$ERRTYP ;GO TO USER ERROR ROUTINE  
2729 012360 104401 001063 TYPE ,$CRLF  
2730 012364 20$:  
2731 012364 122737 000001 001106 CMPB #APTENV,$ENV ;RUNNING IN APT MODE  
2732 012372 001007 BNE 2$ ;NO, SKIP APT ERROR REPORT  
2733 012374 113737 001014 012406 MOVB $ITEMB,21$ ;SET ITEM NUMBER AS ERROR NUMBER  
2734 012402 004737 013170 JSR PC,$ATY4 ;REPORT FATAL ERROR TO APT  
2735 012406 000 .BYTE 0  
2736 012407 000 .BYTE 0  
2737 012410 000777 22$: BR 22$ ;APT ERROR LOOP  
2738 012412 005777 166422 2$: TST @SWR ;HALT ON ERROR  
2739 012416 100001 BPL 3$ ;SKIP IF CONTINUE  
2740 012420 000000 HALT ;HALT ON ERROR!  
2741 012422 104406 3$: CKSWR ;TEST FOR CHANGE IN $JFT-SWR  
2742 012424 032777 001000 166406 BIT #BIT09,@SWR ;LOOP ON ERROR SWITCH SET?  
2743 012432 001402 BEQ 4$ ;BR IF NO  
2744 012434 013716 001010 MOV $LPERR,(SP) ;FUDGE RETURN FOR LOOPING  
2745 012440 005737 001060 4$: TST $ESCAPE ;CHECK FOR AN ESCAPE ADDRESS  
2746 012444 001402 BEQ 5$ ;BR IF NONE  
2747 012446 013716 001060 MOV $ESCAPE,(SP) ;FUDGE RETURN ADDRESS FOR ESCAPE  
2748 012452 5$:  
2749 012452 022737 012152 000042 CMP #SENDAD,@#42 ;ACT-11 AUTO-ACCEPT?  
2750 012460 001001 BNE 6$ ;BR IF NO  
2751 012462 000000 HALT ;YES  
2752 012464 6$:  
2753 012464 000002 RTI ;RETURN  
2754
```

2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

```
$ERRTYP:
      TYPE      , $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
      MOV       RO, -(SP)    ;; SAVE RO
      CLR       RO          ;; PICKUP THE ITEM INDEX
      BISB      @#$ITEMB, RO
      BNE       1$         ;; IF ITEM NUMBER IS ZERO, JUST
                          ;; TYPE THE PC OF THE ERROR
                          ;; SAVE $ERRPC FOR TYPEOUT
                          ;; ERROR ADDRESS
                          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
                          ;; GET OUT
      MOV       $ERRPC, -(SP)
      TYPOC
      BR        6$         ;; ADJUST THE INDEX SO THAT IT WILL
                          ;; WORK FOR THE ERROR TABLE
1$:   DEC       RO
      ASL       RO
      ASL       RO
      ASL       RO
      ADD       #$ERRTB, RO  ;; FORM TABLE POINTER
      MOV       (RO)+, 2$   ;; PICKUP "ERROR MESSAGE" POINTER
      BEQ       3$         ;; SKIP TYPEOUT IF NO POINTER
                          ;; TYPE THE "ERROR MESSAGE"
                          ;; "ERROR MESSAGE" POINTER GOES HERE
2$:   .WORD     0          ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPE      , $CRLF      ;; PICKUP "DATA HEADER" POINTER
                          ;; SKIP TYPEOUT IF 0
                          ;; TYPE THE "DATA HEADER"
                          ;; "DATA HEADER" POINTER GOES HERE
3$:   MOV       (RO)+, 4$   ;; "CARRIAGE RETURN" & "LINE FEED"
      BEQ       5$         ;; PICKUP "DATA TABLE" POINTER
                          ;; GO TYPE THE DATA
                          ;; RESTORE RO
4$:   .WORD     0          ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPE      , $CRLF      ;; PICKUP "DATA TABLE" POINTER
      MOV       (RO), RO    ;; GO TYPE THE DATA
      BNE       7$         ;; RESTORE RO
5$:   MOV       (SP)+, RO   ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPE      , $CRLF      ;; RETURN
      RTS      PC
7$:   MOV       @ (RO)+, -(SP) ;; SAVE @ (RO)+ FOR TYPEOUT
      TYPOC
      TST      (RO)        ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
                          ;; IS THERE ANOTHER NUMBER?
      BEQ       6$         ;; BR IF NO
      TYPE     , 8$        ;; TYPE TWO(2) SPACES
      BR        7$        ;; LOOP
8$:   .ASCIZ   / /         ;; TWO(2) SPACES
      .EVEN
```

ERROR MESSAGE TYPEOUT ROUTINE

```
2804
2805      .SBTTL POWER DOWN AND UP ROUTINES
2806      ::*****
2807      ;*POWER DOWN ROUTINE
2808      ::*****
2809 012622 012737 012762 000024 $PWRDN: MOV    #SILLUP,@#PWRVEC ;SET FOR FAST UP
2810 012630 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;PRIO:7
2811 012636 010046      MOV    R0,-(SP) ;PUSH R0 ON STACK
2812 012640 010146      MOV    R1,-(SP) ;PUSH R1 ON STACK
2813 012642 010246      MOV    R2,-(SP) ;PUSH R2 ON STACK
2814 012644 010346      MOV    R3,-(SP) ;PUSH R3 ON STACK
2815 012646 010446      MOV    R4,-(SP) ;PUSH R4 ON STACK
2816 012650 010546      MOV    R5,-(SP) ;PUSH R5 ON STACK
2817 012652 017746 166162      MOV    @SWR,-(SP) ;PUSH @SWR ON STACK
2818 012656 010637 012766      MOV    SP,$SAVR6 ;SAVE SP
2819 012662 012737 012674 000024      MOV    #SPWRUP,@#PWRVEC ;SET UP VECTOR
2820 012670 000000      HALT
2821 012672 000776      BR     .-2 ;HANG UP
2822
2823
2824      ::*****
2825      ;*POWER UP ROUTINE
2826      ::*****
2827 012674 012737 012762 000024 $PWRUP: MOV    #SILLUP,@#PWRVEC ;SET FOR FAST DOWN
2828 012702 013706 012766      MOV    $SAVR6,SP ;GET SP
2829 012706 012677 166126      MOV    (SP)+,@SWR ;POP STACK INTO @SWR
2830 012712 012605      MOV    (SP)+,R5
2831 012714 012604      MOV    (SP)+,R4 ;POP STACK INTO R4
2832 012716 012603      MOV    (SP)+,R3 ;POP STACK INTO R3
2833 012720 012602      MOV    (SP)+,R2 ;POP STACK INTO R2
2834 012722 012601      MOV    (SP)+,R1 ;POP STACK INTO R1
2835 012724 012600      MOV    (SP)+,R0 ;POP STACK INTO R0
2836 012726 012737 012622 000024      MOV    #SPWRDN,@#PWRVEC ;SET UP THE POWER DOWN VECTOR
2837 012734 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;PRIO:7
2838 012742 005037 012766      CLR    $SAVR6 ;WAIT LOOP FOR THE TTY
2839 012746 005237 012766      1$: INC    $SAVR6 ;WAIT FOR THE INC
2840 012752 001375      BNE    1$ ;OF WORD
2841 012754 104401      TYPE ;REPORT THE POWER FAILURE
2842 012756 012770      $PWRMG: .WORD $POWER ;POWER FAIL MESSAGE POINTER
2843 012760 000002      RTI
2844 012762 000000      $SILLUP: HALT ;THE POWER UP SEQUENCE WAS STARTED
2845 012764 000776      BR     .-2 ; BEFORE THE POWER DOWN WAS COMPLETE
2846 012766 000000      $SAVR6: 0 ;PUT THE SP HERE
2847 012770 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'
2848 012776 000122
2849
```



```
2850
2851      .SBTTL SCOPE HANDLER ROUTINE
2852
2853      ;*****
2854      ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2855      ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2856      ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2857      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2858      ;*SW14=1      LOOP ON TEST
2859      ;*SW09=1      LOOP ON ERROR
2860      ;*CALL
2861      ;*      SCOPE      ;;SCOPE=IOT
2862
2863      $SCOPE:
2864      013000      104406      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
2865      013002      032777      040000      166030      1$:      BIT      #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
2866      013010      001052      BNE      $OVER      ;;YES IF SW14=1
2867      ;*****START OF CODE FOR THE XOR TESTER*****
2868      013012      000416      $XTSTR: BR      6$      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
2869      ;THIS INSTRUCTION TO A 'NOP' (NOP=240)
2870      013014      013746      000004      MOV      @#ERRVEC,-(SP)      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2871      013020      012737      013040      000004      MOV      #5$,@#ERRVEC      ;;SET FOR TIMEOUT
2872      013026      005737      177060      TST      @#177060      ;;TIME OUT ON XOR?
2873      013032      012637      000004      MOV      (SP)+,@#ERRVEC      ;;RESTORE THE ERROR VECTOR
2874      013036      000421      BR      $SVLAD      ;;GO TO THE NEXT TEST
2875      013040      022626      5$:      CMP      (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
2876      013042      012637      000004      MOV      (SP)+,@#ERRVEC      ;;RESTORE THE ERROR VECTOR
2877      013046      000407      BR      7$      ;;LOOP ON THE PRESENT TEST
2878      013050      6$:;*****END OF CODE FOR THE XOR TESTER*****
2879      013050      105737      001003      2$:      TSTB      $ERFLG      ;;HAS AN ERROR OCCURRED?
2880      013054      001412      BEQ      $SVLAD      ;;BR IF NO
2881      013056      032777      001000      165754      BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
2882      013064      001404      BEQ      4$      ;;BR IF NO
2883      013066      013737      001010      001006      7$:      MOV      $LPERR,$LPADR      ;;SET LOOP ADDRESS TO LAST SCOPE
2884      013074      000420      BR      $OVER
2885      013076      105037      001003      4$:      CLR      $ERFLG      ;;ZERO THE ERROR FLAG
2886      013102      105237      001002      $SVLAD: INCB      $TSTNM      ;;COUNT TEST NUMBERS
2887      013106      113737      001002      001072      MOVB      $TSTNM,$TESTN      ;;SET TEST NUMBER IN APT MAILBOX
2888      013114      011637      001006      MOV      (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
2889      013120      011637      001010      MOV      (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
2890      013124      005037      001060      CLR      $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
2891      013130      112737      000001      001015      MOVB      #1,$ERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2892      013136      013777      001002      165676      $OVER: MOV      $TSTNM,@DISPLAY      ;;DISPLAY TEST NUMBER
2893      013144      013716      001006      MOV      $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
2894      013150      000002      RTI      ;;FIXES PS
2895
```

SCOPE HANDLER ROUTINE

```
2896
2897
2898
2899
2900
2901 013152 112737 000001 013416 SATY1: MOVB #1,$FFLG ;TO REPORT FATAL ERROR
2902 013160 112737 000001 013414 SATY3: MOVB #1,$MFLG ;TO TYPE A MESSAGE
2903 013166 000403 BR SATYC
2904 013170 112737 000001 013416 SATY4: MOVB #1,$FFLG ;TO ONLY REPORT FATAL ERROR
2905 013176 SATYC:
2906 013176 010046 MOV R0,-(SP) ;PUSH R0 ON STACK
2907 013200 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
2908 013202 105737 013414 TSTB $MFLG ;SHOULD TYPE A MESSAGE?
2909 013206 001450 BEQ 5$ ;IF NOT: BR
2910 013210 122737 000001 001106 CMPB #APTENV,$ENV ;OPERATING UNDER APT?
2911 013216 00;J31 BNE 3$ ;IF NOT: BR
2912 013220 132737 000100 001107 BITB #APTPOOL,$ENVM ;SHOULD SPOOL MESSAGE?
2913 013226 001425 BEQ 3$ ;IF NOT: BR
2914 013230 017600 000004 MOV @4(SP),R0 ;GET MESSAGE ADDRESS
2915 013234 062766 000002 000004 ADD #2,4(SP) ;BUMP RETURN ADDRESS
2916 013242 005737 001066 1$: TST $MSGTYPE ;SEE IF DONE W/ LAST XMISSION?
2917 013246 001375 BNE 1$ ;IF NOT: WAIT
2918 013250 010037 001102 MOV R0,$MSGAD ;PUT ADDRESS IN MAILBOX
2919 013254 105720 2$: TSTB (R0)+ ;FIND END OF MESSAGE
2920 013256 001376 BNE 2$
2921 013260 163700 001102 SUB $MSGAD,R0 ;SUB START OF MESSAGE
2922 013264 006200 ASR R0 ;GET MESSAGE LENGTH IN WORDS
2923 013266 010037 001104 MOV R0,$MSGGLT ;PUT LENGTH IN MAILBOX
2924 013272 012737 000004 001066 MOV #4,$MSGTYPE ;TELL APT TO TAKE MESSAGE
2925 013300 000413 BR 5$
2926 013302 017637 000004 013326 3$: MOV @4(SP),4$ ;PUT MSG ADDR IN JSR LINKAGE
2927 013310 062766 000002 000004 ADD #2,4(SP) ;BUMP RETURN ADDRESS
2928 013316 0.3746 177776 MOV 177776,-(SP) ;PUSH 177776 ON STACK
2929 013322 004737 013420 JSR PC,$TYPE ;CALL TYPE MACRO
2930 013326 000000 4$: .WORD 0
2931 013330 5$:
2932 013330 105737 013416 10$: TSTB $FFLG ;SHOULD REPORT FATAL ERROR?
2933 013334 001413 BEQ 12$ ;IF NOT: BR
2934 013336 005737 001106 TST $ENV ;RUNNING UNDER APT?
2935 013342 001410 BEQ 12$ ;IF NOT: BR
2936 013344 005737 001066 11$: TST $MSGTYPE ;FINISHED LAST MESSAGE?
2937 013350 001375 BNE 11$ ;IF NOT: WAIT
2938 013352 017637 000004 001070 MOV @4(SP),$FATAL ;GET ERROR #
2939 013360 005237 001066 INC $MSGTYPE ;TELL APT TO TAKE ERROR
2940 013364 062766 000002 000004 12$: ADD #2,4(SP) ;BUMP RETURN ADDRESS
2941 013372 105037 013416 CLRB $FFLG ;CLEAR FATAL FLAG
2942 013376 105037 013415 CLRB $LFLG ;CLEAR LOG FLAG
2943 013402 105037 013414 CLRB $MFLG ;CLEAR MESSAGE FLAG
2944 013406 012601 MOV (SP)+,R1 ;POP STACK INTO R1
2945 013410 012600 MOV (SP)+,R0 ;POP STACK INTO R1
2946 013412 000207 RTS PC ;RETURN
2947 013414 000 $MFLG: .BYTE 0
2948 013415 000 $LFLG: .BYTE 0 ;LOG FLAG
2949 013416 000 $FFLG: .BYTE 0 ;FATAL FLAG
2950
2951 013420 .EVEN
```

2952	000200	APTSIZE=200
2953	000001	APTENV=001
2954	000100	APTSPool=100
2955	000040	APTCSUP=040
2956		

2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012

013420 105737 001057
013424 100002
013426 000000
013430 000430
013432 010046
013434 017600 000002
013440 122737 000001 001106
013446 001011
013450 132737 000100 001107
013456 001405
013460 010037 013470
013464 004737 013160
013470 000000
013472 132737 000040 001107
013500 001003
013502 112046
013504 001005
013506 005726
013510 012600
013512 062716 000002
013516 000002
013520 122716 000011
013524 001430
013526 122716 000200
013532 001006
013534 005726
013536 104401
013540 001063
013542 105037 013676
013546 000755
013550 004737 013632
013554 123726 001056
013560 001350
013562 013746 001054
013566 105366 000001
013572 002770
013574 004737 013632

.SBTTL TYPE ROUTINE

```
*****  
;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.  
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.  
;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.  
;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.  
;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.  
*  
*CALL:  
*1) USING A TRAP INSTRUCTION  
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING  
*OR  
* TYPE  
* MESADR  
*  
$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?  
BPL 1$ ;;BR IF YES  
HALT ;;HALT HERE IF NO TERMINAL  
BR 3$ ;;LEAVE  
1$: MOV R0,-(SP) ;;SAVE R0  
MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING  
CMPB #APTENV,$ENV ;;RUNNING IN APT MODE  
BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE  
BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT  
BEQ 62$ ;;NO,GO CHECK FOR CONSOLE  
MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT  
JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT  
61$: .WORD 0 ;;MESSAGE ADDRESS  
62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED  
BNE 60$ ;;YES,SKIP TYPE OUT  
2$: MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK  
BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR  
TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK  
60$: MOV (SP)+,R0 ;;RESTORE R0  
3$: ADD #2,(SP) ;;ADJUST RETURN PC  
RTI ;;RETURN  
4$: CMPB #HT,(SP) ;;BRANCH IF <HT>  
BEQ 8$  
CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>  
BNE 5$  
TST (SP)+ ;;POP <CR><LF> EQUIV  
TYPE ;;TYPE A CR AND LF  
$CRLF  
CLRB $CHARCNT ;;CLEAR CHARACTER COUNT  
BR 2$ ;;GET NEXT CHARACTER  
5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER  
6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?  
BNE 2$ ;;IF NO GO GET NEXT CHAR.  
MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED  
 ;;AND THE NULL CHAR.  
7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?  
BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK  
JSR PC,$TYPEC ;;GO TYPE A NULL
```

```
3013 013600 105337 013676          DECB  $CHARCNT      ;;DO NOT COUNT AS A COUNT
3014 013604 000770          BR      7$          ;;LOOP
3015
3016          ;HORIZONTAL TAB PROCESSOR
3017
3018 013606 112716 000040      8$:   MOVB  #' ,(SP)      ;;REPLACE TAB WITH SPACE
3019 013612 004737 013632      9$:   JSR   PC,$TYPEC     ;;TYPE A SPACE
3020 013616 132737 000007 013676  BITB  #7,$CHARCNT     ;;BRANCH IF NOT AT
3021 013624 001372          BNE   9$          ;;TAB STOP
3022 013626 005726          TST   (SP)+        ;;POP SPACE OFF STACK
3023 013630 000724          BR    2$          ;;GET NEXT CHARACTER
3024 013632 105777 165212      $TYPEC: TSTB @ $TPS     ;;WAIT UNTIL PRINTER IS READY
3025 013636 100375          BPL   $TYPEC
3026 013640 116677 000002 165204  MOVB  2(SP),@ $TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3027 013646 122766 000015 000002  CMPB  #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
3028 013654 001003          BNE   1$          ;;BRANCH IF NO
3029 013656 105037 013676      CLRB  $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
3030 013662 000406          BR    $TYPEX     ;;EXIT
3031 013664 122766 000012 000002 1$:   CMPB  #LF,2(SP)     ;;IS CHARACTER A LINE FEED?
3032 013672 001402          BEQ   $TYPEX     ;;BRANCH IF YES
3033 013674 105227          INCB  (PC)+      ;;COUNT THE CHARACTER
3034 013676 000000      $CHARCNT: .WORD  0  ;;CHARACTER COUNT STORAGE
3035 013700 000207      $TYPEX: RTS      PC
3036
3037          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3038
3039          ;;*****
3040          ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3041          ;;*OCTAL (ASCII) NUMBER AND TYPE IT.
3042          ;;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3043          ;;*CALL:
3044          ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
3045          ;;*   TYPOS          ;;CALL FOR TYPEOUT
3046          ;;*   .BYTE  N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3047          ;;*   .BYTE  M          ;;M=1 OR 0
3048          ;;*                               ;;1=TYPE LEADING ZEROS
3049          ;;*                               ;;0=SUPPRESS LEADING ZEROS
3050          ;;*
3051          ;;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3052          ;;*$TYPOS OR $TYPOC
3053          ;;*CALL:
3054          ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
3055          ;;*   TYPON          ;;CALL FOR TYPEOUT
3056          ;;*
3057          ;;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3058          ;;*CALL:
3059          ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
3060          ;;*   TYPOC          ;;CALL FOR TYPEOUT
3061
3062 013702 017646 000000      $TYPOS: MOV  @(SP),-(SP)     ;;PICKUP THE MODE
3063 013706 116637 000001 014125  MOVB  1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
3064 013714 112637 014127  MOVB  (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
3065 013720 062716 000002  ADD   #2,(SP)        ;;ADJUST RETURN ADDRESS
3066 013724 000406          BR    $TYPON
3067 013726 112737 000001 014125  $TYPOC: MOVB #1,$OFILL  ;;SET THE ZERO FILL SWITCH
3068 013734 112737 000006 014127  MOVB  #6,$OMODE+1   ;;SET FOR SIX(6) DIGITS
```

BINARY TO OCTAL (ASCII) AND TYPE

```
3069 013742 112737 000005 014124 $TYPON: MOVB #5,$SOCNT ;;SET THE ITERATION CL NT
3070 013750 010346 MOV R3,-(SP) ;;SAVE R3
3071 013752 010446 MOV R4,-(SP) ;;SAVE R4
3072 013754 010546 MOV R5,-(SP) ;;SAVE R5
3073 013756 113704 014127 MOVB $OMODE+1,R4 ;;GET THE NUMBER OF DI'ITS TO TYPE
3074 013762 005404 NEG R4
3075 013764 062704 000006 ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
3076 013770 110437 014126 MOVB R4,$OMODE ;;SAVE IT FOR USE
3077 013774 113704 014125 MOVB $OFILL,R4 ;;GET THE ZERO FILL SWITCH
3078 014000 016605 000012 MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
3079 014004 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
3080 014006 006105 1$: ROL R5 ;;ROTATE MSB INTO 'C'
3081 014010 000404 BR 3$ ;;GO DO MSB
3082 014012 006105 2$: ROL R5 ;;FORM THIS DIGIT
3083 014014 006105 ROL R5
3084 014016 006105 ROL R5
3085 014020 010503 MOV R5,R3
3086 014022 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
3087 014024 105337 014126 DECB $OMODE ;;TYPE THIS DIGIT?
3088 014030 100016 BPL 7$ ;;BR IF NO
3089 014032 042703 177770 BIC #177770,R3 ;;GET RID OF JUNK
3090 014036 001002 BNE 4$ ;;TEST FOR 0
3091 014040 005704 TST R4 ;;SUPPRESS THIS 0?
3092 014042 001403 BEQ 5$ ;;BR IF YES
3093 014044 065204 4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
3094 014046 052703 000060 BIS #'0,R3 ;;MAKE THIS DIGIT ASCII
3095 014052 052703 000040 5$: BIS #' ,R3 ;;MAKE ASCII IF NOT ALREADY
3096 014056 110337 014122 MOVB R3,8$ ;;SAVE FOR TYPING
3097 014062 104401 014122 TYPE ,8$ ;;GO TYPE THIS DIGIT
3098 014066 105337 014124 7$: DECB $SOCNT ;;COUNT BY 1
3099 014072 003347 BGT 2$ ;;BR IF MORE TO DO
3100 014074 002402 BLT 6$ ;;BR IF DONE
3101 014076 005204 INC R4 ;;INSURE LAST DIGIT ISN'T A BLANK
3102 014100 000744 BR 2$ ;;GO DO THE LAST DIGIT
3103 014102 012605 6$: MOV (SP)+,R5 ;;RESTORE R5
3104 014104 012604 MOV (SP)+,R4 ;;RESTORE R4
3105 014106 012603 MOV (SP)+,R3 ;;RESTORE R3
3106 014110 016666 000002 000004 MOV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING
3107 014116 012616 MOV (SP)+,(SP)
3108 014120 000002 RTI ;;RETURN
3109 014122 000 8$: .BYTE 0 ;;STORAGE FOR ASCII DIGIT
3110 014123 000 .BYTE 0 ;;TERMINATOR FOR TYPE ROUTINE
3111 014124 000 $SOCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
3112 014125 000 $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
3113 014126 000000 $OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
3114
```

BINARY TO OCTAL (ASCII) AND TYPE

```
3115
3116          .SBTTL  TTY INPUT ROUTINE
3117
3118          ;:*****
3119          .ENABL  LSB
3120
3121          ;:*****
3122          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3123          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3124          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3125          ;*WHEN OPERATING IN TTY FLAG MODE.
3126 014130 022737 000176 001040 $CKSWR: CMP    #SWREG,SWR    ;; IS THE SOFT-SWR SELECTED?
3127 014136 001074          BNE    15$          ;; BRANCH IF NO
3128 014140 105777 164700          TSTB   @STKS          ;; CHAR THERE?
3129 014144 100071          BPL    15$          ;; IF NO, DON'T WAIT AROUND
3130 014146 117746 164674          MOVB  @STKB,-(SP)    ;; SAVE THE CHAR
3131 014152 042716 177600          BIC   # C177,(SP)    ;; STRIP-OFF THE ASCII
3132 014156 022726 000007          CMP   #7,(SP)+      ;; IS IT A CONTROL G?
3133 014162 001062          BNE    15$          ;; NO, RETURN TO USER
3134 014164 123727 001034 000001          CMPB  $AUTOB,#1     ;; ARE WE RUNNING IN AUTO-MODE?
3135 014172 001456          BEQ   15$          ;; BRANCH IF YES
3136
3137 014174 104401 014655          TYPE  ,SCNTLG      ;; ECHO THE CONTROL-G ( G)
3138 014200 104401 014662          $GTSWR: TYPE  ,SMSWR      ;; TYPE CURRENT CONTENTS
3139 014204 013746 000176          MOV   SWREG,-(SP)   ;; SAVE SWREG FOR TYPEOUT
3140 014210 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
3141 014212 104401 014675          TYPE  ,SMNEW      ;; PROMPT FOR NEW SWR
3142 014216 005046          19$: CLR   -(SP)    ;; CLEAR COUNTER
3143 014220 005046          CLR   -(SP)    ;; THE NEW SWR
3144 014222 105777 164616          7$:  TSTB  @STKS    ;; CHAR THERE?
3145 014226 100375          BPL   7$        ;; IF NOT TRY AGAIN
3146
3147 014230 117746 164612          MOVB  @STKB,-(SP)  ;; PICK UP CHAR
3148 014234 042716 177600          BIC   # C177,(SP)  ;; MAKE IT 7-BIT ASCII
3149
3150
3151
3152 014240 021627 000025          9$:  CMP   (SP),#25   ;; IS IT A CONTROL-U?
3153 014244 001005          BNE   10$        ;; BRANCH IF NOT
3154 014246 104401 014650          TYPE  ,SCNTLU     ;; YES, ECHO CONTROL-U ( U)
3155 014252 062706 000006          20$: ADD  #6,SP     ;; IGNOKE PREVIOUS INPUT
3156 014256 000757          BR   19$        ;; LET'S TRY IT AGAIN
3157
3158
3159 014260 021627 000015          10$: CMP   (SP),#15   ;; IS IT A <CR>?
3160 014264 001022          BNE   16$        ;; BRANCH IF NO
3161 014266 005766 000004          TST   4(SP)      ;; YES, IS IT THE FIRST CHAR?
3162 014272 001403          BEQ   11$        ;; BRANCH IF YES
3163 014274 016677 000002 164536          MOV   2(SP),@SWR   ;; SAVE NEW SWR
3164 014302 062706 000006          11$: ADD  #6,SP     ;; CLEAR UP STACK
3165 014306 104401 001063          14$: TYPE  ,SCRLF   ;; ECHO <CR> AND <LF>
3166 014312 123727 001035 000001          CMPB  $INTAG,#1    ;; RE-ENABLE TTY KBD INTERRUPTS?
3167 014320 001003          BNE   15$        ;; BRANCH IF NOT
3168 014322 012777 000100 164514          MOV   #100,@STKS  ;; RE-ENABLE TTY KBD INTERRUPTS
3169 014330 000002          15$: RTI          ;; RETURN
3170 014332 004737 013632          16$: JSR   PC,$TYPEC ;; ECHO CHAR
```

```
3171 014336 021627 000060      CMP      (SP),#60      ;;CHAR < 0?
3172 014342 002420      BLT      18$          ;;BRANCH IF YES
3173 014344 021627 000067      CMP      (SP),#67      ;;CHAR > 7?
3174 014350 003015      BGT      18$          ;;BRANCH IF YES
3175 014352 042726 000060      BIC      #60,(SP)+     ;;STRIP-OFF ASCII
3176 014356 005766 000002      TST      2(SP)        ;;IS THIS THE FIRST CHAR
3177 014362 001403      BEQ      17$          ;;BRANCH IF YES
3178 014364 006316      ASL      (SP)         ;;NO, SHIFT PRESENT
3179 014366 006316      ASL      (SP)         ;;  CHAR OVER TO MAKE
3180 014370 006316      ASL      (SP)         ;;  ROOM FOR NEW ONE.
3181 014372 005266 000002      17$: INC      2(SP)        ;;KEEP COUNT OF CHAR
3182 014376 056616 177776      BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
3183 014402 000707      BR       7$           ;;GET THE NEXT ONE
3184 014404 104401 001062      18$: TYPE  ,%QUES     ;;TYPE ?<CR><LF>
3185 014410 000720      RR       20$         ;;SIMULATE CONTROL-U
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197 014412 011646      $RDCHR: MOV      (SP),-(SP)  ;;PUSH DOWN THE PC
3198 014414 016666 000004 000002      MOV      4(SP),2(SP)  ;;SAVE THE PS
3199 014422 105777 164416      1$: TSTB     @%TKS     ;;WAIT FOR
3200 014426 100375      BPL      1$           ;;A CHARACTER
3201 014430 117766 164412 000004      MOVB     @%TKB,4(SP)  ;;READ THE TTY
3202 014436 042766 177600 000004      BIC      # C<177>,4(SP) ;;GET RID OF JUNK IF ANY
3203 014444 026627 000004 000023      CMP      4(SP),#23   ;;IS IT A CONTROL-S?
3204 014452 001013      BNE      3$           ;;BRANCH IF NO
3205 014454 105777 164364      2$: TSTB     @%TKS     ;;WAIT FOR A CHARACTER
3206 014460 100375      BPL      2$           ;;LOOP UNTIL ITS THERE
3207 014462 117746 164360      MOVB     @%TKB,-(SP)  ;;GET CHARACTER
3208 014466 042716 177600      BIC      # C177,(SP)  ;;MAKE IT 7-BIT ASCII
3209 014472 022627 000021      CMP      (SP)+,#21   ;;IS IT A CONTROL-Q?
3210 014476 001366      BNE      2$           ;;IF NOT DISCARD IT
3211 014500 000750      BR       1$           ;;YES, RESUME
3212 014502 026627 000004 000140      3$: CMP      4(SP),#140 ;;IS IT UPPER CASE?
3213 014510 002407      BLT      4$           ;;BRANCH IF YES
3214 014512 026627 000004 000175      CMP      4(SP),#175  ;;IS IT A SPECIAL CHAR?
3215 014520 003003      BGT      4$           ;;BRANCH IF YES
3216 014522 042766 000040 000004      BIC      #40,4(SP)   ;;MAKE IT UPPER CASE
3217 014530 000002      4$: RTI              ;;GO BACK TO USER
3218
3219
3220
3221
3222
3223
3224
3225 014532 010346      $RDLIN: MOV      R3,-(SP)  ;;SAVE R3
3226 014534 012703 014640      1$: MOV      #%TTYIN,R3  ;;GET ADDRESS
```


3256
3257
3258 .SBTTL TRAP DECODER
3259
3260
3261 ;*****
3262 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3263 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3264 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3265 ;*GO TO THAT ROUTINE.

3266 014704 010046 \$TRAP: MOV R0,-(SP) ;;SAVE R0
3267 014706 016600 000002 MOV 2(SP),R0 ;;GET TRAP ADDRESS
3268 014712 005740 TST -(R0) ;;BACKUP BY 2
3269 014714 111000 MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
3270 014716 006300 ASL R0 ;;POSITION FOR INDEXING
3271 014720 016000 014740 MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE
3272 014724 000200 RTS R0 ;;GO TO ROUTINE

3273
3274
3275 ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

3276
3277 014726 011646 \$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
3278 014730 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
3279 014736 000002 RTI ;;RESTORE THE PSW

3280
3281 .SBTTL TRAP TABLE

3282
3283 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3284 ;*BY THE "TRAP" INSTRUCTION.

3285 ; ROUTINE
3286 ; -----
3287 ;
3288 014740 014726 \$TRPAD: .WORD \$TRAP2
3289 014742 013420 \$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
3290 014744 013726 \$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3291 014746 013702 \$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3292 014750 013742 \$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
3293
3294 014752 014200 \$GTSWR ;;CALL=GTSWR TRAP+5(104405) GET SOFT-SWR SETTING
3295
3296 014754 014130 \$CKSWR ;;CALL=CKSWR TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
3297 014756 014412 \$RDCHR ;;CALL=RDCHR TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
3298 014760 014532 \$RDLIN ;;CALL=RDLIN TRAP+10(104410) TTY TYPEIN STRING ROUTINE
3299

3390						
3391						
3392	015266	040503	020116	047516	EM1:	.ASCIZ /CAN NOT ACCESS TCSR/
3393	015274	020124	041501	042503		
3394	015302	051523	052040	051503		
3395	015310	000122				
3396	015312	040503	020116	047516	EM2:	.ASCIZ /CAN NOT ACCESS TBUF/
3397	015320	020124	041501	042503		
3398	015326	051523	052040	052502		
3399	015334	000106				
3400	015336	041524	051123	042040	EM3:	.ASCIZ /TCSR DONE NOT CLEARED WITH TBUF FULL/
3401	015344	047117	020105	047516		
3402	015352	020124	046103	040505		
3403	015360	042522	020104	044527		
3404	015366	044124	052040	052502		
3405	015374	020106	052506	046114		
3406	015402	000				
3407	015403	124	051503	020122	EM4:	.ASCIZ /TCSR DONE NOT SET/
3408	015410	047504	042516	047040		
3409	015416	052117	051440	052105		
3410	015424	000				
3411	015425	124	051503	020122	EM5:	.ASCIZ /TCSR DONE NOT SET WITH RESET/
3412	015432	047504	042516	047040		
3413	015440	052117	051440	052105		
3414	015446	053440	052111	020110		
3415	015454	042522	042523	000124		
3416	015462	040503	020116	047516	EM6:	.ASCIZ /CAN NOT ACCESS RCSR/
3417	015470	020124	041501	042503		
3418	015476	051523	051040	051503		
3419	015504	000122				
3420	015506	040503	020116	047516	EM7:	.ASCIZ /CAN NOT ACCESS RBUF/
3421	015514	020124	041501	042503		
3422	015522	051523	051040	052502		
3423	015530	000106				
3424	015532	040503	020116	047516	EM10:	.ASCIZ /CAN NOT ACCESS LKS/
3425	015540	020124	041501	042503		
3426	015546	051523	046040	051513		
3427	015554	000				
3428	015555	102	052111	020060	EM11:	.ASCIZ /BITO OF TCSR NOT CLEAR AFTER RESET/
3429	015562	043117	052040	051503		
3430	015570	020122	047516	020124		
3431	015576	046103	040505	020122		
3432	015604	043101	042524	020122		
3433	015612	042522	042523	000124		
3434	015620	040503	020116	047516	EM12:	.ASCIZ /CAN NOT SET BITO OF TCSR/
3435	015626	020124	042523	020124		
3436	015634	044502	030124	047440		
3437	015642	020106	041524	051123		
3438	015650	000				
3439	015651	103	047101	047040	EM13:	.ASCIZ /CAN NOT CLEAR BITO OF TCSR/
3440	015656	052117	041440	042514		
3441	015664	051101	041040	052111		
3442	015672	020060	043117	052040		
3443	015700	051503	000122			
3444	015704	042522	042523	020124	EM14:	.ASCIZ /RESET DID NOT CLEAR BITO OF TCSR/
3445	015712	044504	020104	047516		

.M
CZ
EM
EM
EM
EM
EM
EM
EM
EM
ENI
ENI
ER
FI
GN
GO
GT
HT
ID
IN
IO
LF
LK
MA
MF
MO
MP
MS
MS
M1
M2
M2
OL
OU
PI
PI
PR
PR
PR
PR
PR
PS
PW
RB
RC

3502	016376	020122	047516	020124	
3503	016404	046103	040505	020122	
3504	016412	043101	042524	020122	
3505	016420	042522	042523	000124	
3506	016426	041522	051126	044440	EM27: .ASCIZ /RCVR INTERRUPT WITH PRIORITY 7/
3507	016434	052116	051105	052522	
3508	016442	052120	053440	052111	
3509	016450	020110	051120	047511	
3510	016456	044522	054524	033440	
3511	016464	000			
3512	016465	103	047101	047040	EM30: .ASCIZ /CAN NOT SET BIT6 OF RCSR/
3513	016472	052117	051440	052105	
3514	016500	041040	052111	020066	
3515	016506	043117	051040	051503	
3516	016514	000122			
3517	016516	040503	020116	047516	EM31: .ASCIZ /CAN NOT CLEAR BIT6 OF RCSR/
3518	016524	020124	046103	040505	
3519	016532	020122	044502	033124	
3520	016540	047440	020106	041522	
3521	016546	051123	000		
3522	016551	103	047101	047040	EM32: .ASCIZ /CAN NOT CLEAR BIT6 OF RCSR WITH RESET/
3523	016556	052117	041440	042514	
3524	016564	051101	041040	052111	
3525	016572	020066	043117	051040	
3526	016600	051503	020122	044527	
3527	016606	044124	051040	051505	
3528	016614	052105	000		
3529	016617	102	052111	020066	EM33: .ASCIZ /BIT6 OF LKS NOT CLEAR AFTER RESET/
3530	016624	043117	046040	051513	
3531	016632	047040	052117	041440	
3532	016640	042514	051101	040440	
3533	016646	052106	051105	051040	
3534	016654	051505	052105	000	
3535	016661	114	051513	044440	EM34: .ASCIZ /LKS INTERRUPT WITH PRIORITY 7/
3536	016666	052116	051105	052522	
3537	016674	052120	053440	052111	
3538	016702	020110	051120	047511	
3539	016710	044522	054524	033440	
3540	016716	000			
3541	016717	103	047101	047040	EM35: .ASCIZ /CAN NOT SET BIT6 OF LKS/
3542	016724	052117	051440	052105	
3543	016732	041040	052111	020066	
3544	016740	043117	046040	051513	
3545	016746	000			
3546	016747	103	047101	047040	EM36: .ASCIZ /CAN NOT CLEAR BIT6 OF LKS/
3547	016754	052117	041440	042514	
3548	016762	051101	041040	052111	
3549	016770	020066	043117	046040	
3550	016776	051513	000		
3551	017001	122	051505	052105	EM37: .ASCIZ /RESET DID NOT CLEAR BIT6 OF LKS/
3552	017006	042040	042111	047040	
3553	017014	052117	041440	042514	
3554	017022	051101	041040	052111	
3555	017030	020066	043117	046040	
3556	017036	051513	000		
3557	017041	104	040525	020114	EM40: .ASCIZ /DUAL ADDRESSING ERROR/

3558	017046	042101	051104	051505	
3559	017054	044523	043516	042440	
3560	017062	051122	051117	000	
3561	017067	102	052111	020066	EM41: .ASCIZ /BIT6 OF LKS NOT SET AFTER RESET/
3562	017074	043117	046040	051513	
3563	017102	047040	052117	051440	
3564	017110	052105	040440	052106	
3565	017116	051105	051040	051505	
3566	017124	052105	000		
3567	017127	103	047101	047040	EM42: .ASCIZ /CAN NOT CLEAR BIT7 OF LKS/
3568	017134	052117	041440	042514	
3569	017142	051101	041040	052111	
3570	017150	020067	043117	046040	
3571	017156	051513	000		
3572	017161	102	052111	020067	EM43: .ASCIZ /BIT7 OF LKS DOES NOT SET/
3573	017166	043117	046040	051513	
3574	017174	042040	042517	020123	
3575	017202	047516	020124	042523	
3576	017210	000124			
3577	017212	052122	020103	047111	EM44: .ASCIZ /RTC INTERRUPT AT PRIORITY 7/
3578	017220	042524	051122	050125	
3579	017226	020124	052101	050040	
3580	017234	044522	051117	052111	
3581	017242	020131	000067		
3582	017246	052122	020103	047111	EM45: .ASCIZ /RTC INTERRUPTS WHEN DISABLED/
3583	017254	042524	051122	050125	
3584	017262	051524	053440	042510	
3585	017270	020116	044504	040523	
3586	017276	046102	042105	000	
3587	017303				EM47:
3588	017303	122	041524	044440	EM46: .ASCIZ /RTC INTERRUPT DID NOT OCCUR/
3589	017310	052116	051105	052522	
3590	017316	052120	042040	042111	
3591	017324	047040	052117	047440	
3592	017332	041503	051125	000	
3593	017337	122	041524	042040	EM50: .ASCIZ /RTC DOUBLE INTERRUPT/
3594	017344	052517	046102	020105	
3595	017352	047111	042524	051122	
3596	017360	050125	000124		
3597	017364	042522	042523	020124	EM51: .ASCIZ /RESET DID NOT INTERRUPT/
3598	017372	044504	020104	047516	
3599	017400	020124	047111	042524	
3600	017406	051122	050125	000124	
3601	017414	052122	020103	047111	EM52: .ASCIZ /RTC INTERRUPT DID NOT CLEAR WITH BIT7 OF LKS/
3602	017422	042524	051122	050125	
3603	017430	020124	044504	020104	
3604	017436	047516	020124	046103	
3605	017444	040505	020122	044527	
3606	017452	044124	041040	052111	
3607	017460	020067	043117	046040	
3608	017466	051513	000		
3609	017471	103	047514	045503	EM53: .ASCIZ /CLOCK REPEATABILITY/
3610	017476	051040	050105	040505	
3611	017504	040524	044502	044514	
3612	017512	054524	000		
3613	017515	130	044515	020124	EM54: .ASCIZ /XMIT INTERRUPTS WHEN DISABLED/

.M
CZ
TVE
TYF
TYF
TYF
VCT
VCT
VEC
WAC
WAI
WDC
WRF
WT
XCC
XMI
XMI
XRE
SAF
SAS
SAT
SAT
SAT
SAT
SAL
SBA
SBE
SBE
SCA
SCA
SCA
SCA
SCA
SCA
SCF
SCF
SDI
SDE
SDC
SEM
SEM
SEM
SEM
SEC
SEC
SEF
SEF
SEF
SEF

3726	020637	047	051105	047522	EM102: .ASCIZ /'ERROR' NOT SET WITH 'OR' FLAG/
3727	020644	023522	047040	052117	
3728	020652	051440	052105	053440	
3729	020660	052111	020110	047447	
3730	020666	023522	043040	040514	
3731	020674	000107			
3732	020676	051102	040505	020113	EM103: .ASCIZ /BREAK DID NOT XMIT ALL ZEROES/
3733	020704	044504	020104	047516	
3734	020712	020124	046530	052111	
3735	020720	040440	046114	055040	
3736	020726	051105	042517	000123	
3737	020734	051102	040505	020113	EM104: .ASCIZ /BREAK DID NOT SET 'FR' ERROR/
3738	020742	044504	020104	047516	
3739	020750	020124	042523	020124	
3740	020756	043047	023522	042440	
3741	020764	051122	051117	000	
3742	020771	104	052101	020101	EM105: .ASCIZ /DATA COMPARE ERROR/
3743	020776	047503	050115	051101	
3744	021004	020105	051105	047522	
3745	021012	000122			
3746	021014	040504	040524	041440	EM106: .ASCIZ /DATA COMPARE ERROR WITH CABLE/
3747	021022	046517	040520	042522	
3748	021030	042440	051122	051117	
3749	021036	053440	052111	020110	
3750	021044	040503	046102	000105	
3751	021052	044524	042515	052517	EM107: .ASCIZ /TIMEOUT IN EXERCISER TEST/
3752	021060	020124	047111	042440	
3753	021066	042530	041522	051511	
3754	021074	051105	052040	051505	
3755	021102	000124			
3756	021104	047111	047503	051122	EM110: .ASCIZ /INCORRECT RECEIVE COUNT/
3757	021112	041505	020124	042522	
3758	021120	042503	053111	020105	
3759	021126	047503	047125	000124	
3760	021134	040504	040524	041440	EM111: .ASCIZ /DATA COMPARE ERROR IN EXERCISER/
3761	021142	046517	040520	042522	
3762	021150	042440	051122	051117	
3763	021156	044440	020116	054105	
3764	021164	051105	044503	042523	
3765	021172	000122			
3766	021174	051124	050101	041440	EM112: .ASCIZ /TRAP CATCHER/
3767	021202	052101	044103	051105	
3768	021210	000			
3769	021211	116	020117	046103	EM113: .ASCIZ /NO CLK INTERRUPT IN EXERCISER/
3770	021216	020113	047111	042524	
3771	021224	051122	050125	020124	
3772	021232	047111	042440	042530	
3773	021240	041522	051511	051105	
3774	021246	000			
3775	021247	042	051105	047522	EM114: .ASCIZ /'ERROR' NOT SET WITH 'FR' FLAG/
3776	021254	021122	047040	052117	
3777	021262	051440	052105	053440	
3778	021270	052111	020110	043042	
3779	021276	021122	043040	040514	
3780	021304	000107			
3781	021306	041522	020126	041501	EM115: .ASCIZ /RCV ACTIVE NOT CLEAR WITH INIT/

TERMINAL OUTPUT TEST

3838	021757	124	051505	021524	DH105:	.ASCIZ	/TEST#	FRR PC	RCSR	GOOD	BAD/
3839	021764	020040	042440	051122							
3840	021772	050040	020103	051040							
3841	022000	051503	020122	020040							
3842	022006	043440	047517	020104							
3843	022014	020040	041040	042101							
3844	022022	000									
3845	022023	124	051505	021524	DH110:	.ASCIZ	/TEST#	ERR PC	RCSR	TRANS	RCV/
3846	022030	020040	042440	051122							
3847	022036	050040	020103	051040							
3848	022044	051503	020122	020040							
3849	022052	052040	040522	051516							
3850	022060	020040	051040	053103							
3851	022066	000									
3852	022067	124	051505	021524	DH112:	.ASCIZ	/TEST#	ERR PC	RCSR	OLDPC	TRAP ADR/
3853	022074	020040	042440	051122							
3854	022102	050040	020103	051040							
3855	022110	051503	020122	020040							
3856	022116	047440	042114	041520							
3857	022124	020040	052040	040522							
3858	022132	020120	042101	000122							
3859						.EVEN					
3860	022140	005015	055103	046104	M1:	.ASCIZ	<CR><LF>/CZDLD-C/				
3861	022146	026504	000103								
3862						.EVEN					
3863	022152	005015			M2:	.ASCII	<CR><LF>				
3864	022154	020040	042040	053105	M2A:	.ASCIZ	/ DEVICES UNDER TEST /				
3865	022162	041511	051505	052440							
3866	022170	042116	051105	052040							
3867	022176	051505	020124	000040							
3868											
3869						.EVEN					
3870	022204	001072	001016	002352	DT1:	.WORD	\$TESTN,\$ERRPC,\$TCSR,0				
3871	022212	000000									
3872	022214	001072	001016	002354	DT2:	.WORD	\$TESTN,\$ERRPC,\$TBUF,0				
3873	022222	000000									
3874	022224	001072	001016	002346	DT6:	.WORD	\$TESTN,\$ERRPC,\$RCSR,0				
3875	022232	000000									
3876	022234	001072	001016	002350	DT7:	.WORD	\$TESTN,\$ERRPC,\$RBUF,0				
3877	022242	000000									
3878	022244	001072	001016	002406	DT10:	.WORD	\$TESTN,\$ERRPC,\$LKS,0				
3879	022252	000000									
3880	022254	001072	001016	001020	DT40:	.WORD	\$TESTN,\$ERRPC,\$GDADR,\$BDADR,0				
3881	022262	001022	000000								
3882	022266	001072	001016	002406	DT53:	.WORD	\$TESTN,\$ERRPC,\$LKS,FIRST,SECND,0				
3883	022274	006600	006602	000000							
3884	022302	001072	001016	002346	DT103:	.WORD	\$TESTN,\$ERRPC,\$RCSR,\$BDDAT,0				
3885	022310	001026	000000								
3886	022314	001072	001016	002346	DT105:	.WORD	\$TESTN,\$ERRPC,\$RCSR,\$GDDAT,\$BDDAT,0				
3887	022322	001024	001026	000000							
3888	022330	001072	001016	002346	DT110:	.WORD	\$TESTN,\$ERRPC,\$RCSR,\$XMCNT,\$RCVCNT,0				
3889	022336	011730	011726	000000							
3890	022344	001072	001016	002346	DT112:	.WORD	\$TESTN,\$ERRPC,\$RCSR,\$OLDPC,\$BDVECT,0				
3891	022352	012300	012302	000000							
3892		000001				.END					

CROSS REFERENCE TABLE -- USER SYMBOLS

ABASE = 176500	537#	650	691	
ACDW1 = 000000	650			
ACDW2 = 000000	650			
ACPUOP= 000000	650	665		
ADDW0 = 000000	650			
ADDW1 = 000000	650			
ADDW10= 000000	650			
ADDW11= 000000	650			
ADDW12= 000000	650			
ADDW13= 000000	650			
ADDW14= 000000	650			
ADDW15= 000000	650			
ADDW2 = 000000	650			
ADDW3 = 000000	650			
ADDW4 = 000000	650			
ADDW5 = 000000	650			
ADDW6 = 000000	650			
ADDW7 = 000000	650			
ADDW8 = 000000	650			
ADDW9 = 000000	650			
ADEVCT= 000000	650	656		
ADEVM = 000000	650	692		
ADR 003562	1314#	1317		
ADRTBL 002414	1139#	1145	1312	
AENV = 000000	650	661		
AENVM = 000000	650	662		
AFATAL= 000000	650	653		
AMADR1= 000000	650	678		
AMADR2= 000000	650	682		
AMADR3= 000000	650	685		
AMADR4= 000000	650	688		
AMAMS1= 000000	650	672		
AMAMS2= 000000	650	680		
AMAMS3= 000000	650	683		
AMAMS4= 000000	650	686		
AMSGAD= 000000	650	658		
AMSGLG= 000000	650	659		
AMSGTY= 000000	650	652		
AMTYP1= 000000	650	673		
AMTYP2= 000000	650	681		
AMTYP3= 000000	650	684		
AMTYP4= 000000	650	687		
APASS = 000000	650	655		
APRIOR= 000000	650			
APTCSU= 000040	2955#	2988		
APTENV= 000001	2731	2910	2953#	2981
APTSIZ= 000200	1207	2952#		
APTSPO= 000100	2912	2954#	2983	
APTSZD 003266	1218	1220	1245#	
ASWREG= 000000	650	663		
ATESTN= 000000	650	654		
AUNIT = 000000	650	657		
AUSWR = 000400	539#	650	664	
AVECT1= 000300	538#	650	689	
AVECT2= 000000	650	690		
BDVECT 012302	2697*	2703#	3890	

CROSS REFERENCE TABLE -- USER SYMBOLS

EM113	021211	1083	3769#
EM114	021247	1088	3775#
EM115	021306	1093	3781#
EM116	021345	1098	3787#
EM117	021404	1103	3793#
EM12	015620	757	3434#
EM13	015651	762	3439#
EM14	015704	767	3444#
EM15	015745	772	3450#
EM16	016010	777	3456#
EM17	016041	782	3461#
EM2	015312	717	3396#
EM20	016074	787	3466#
EM21	016135	792	3472#
EM22	016201	797	3479#
EM23	016236	802	3484#
EM24	016267	807	3489#
EM25	016322	812	3494#
EM26	016363	817	3500#
EM27	016426	822	3506#
EM3	015336	722	3400#
EM30	016465	827	3512#
EM31	016516	832	3517#
EM32	016551	837	3522#
EM33	016617	842	3529#
EM34	016661	847	3535#
EM35	016717	852	3541#
EM36	016747	857	3546#
EM37	017001	862	3551#
EM4	015403	727	3407#
EM40	017041	867	3557#
EM41	017067	872	3561#
EM42	017127	877	3567#
EM43	017161	882	3572#
EM44	017212	887	3577#
EM45	017246	892	3582#
EM46	017303	897	3588#
EM47	017303	902	3587#
EM5	015425	732	3411#
EM50	017337	907	3593#
EM51	017364	912	3597#
EM52	017414	917	3601#
EM53	017471	922	3609#
EM54	017515	927	3613#
EM55	017653	932	3631#
EM56	017553	937	3619#
EM57	017611	942	3625#
EM6	015462	737	3416#
EM60	017653	947	3632#
EM61	017702	952	3636#
EM62	017726	957	3640#
EM63	017773	962	3647#
EM64	020017	967	3651#
EM65	020043	972	3655#
EM66	020111	977	3662#
EM67	020147	982	3668#

\$40CAT= ***** U	2865													
= 022360	545#	551	552#	556#	562#	566#	571#	576	577#	579#	581#	587	588#	
	590#	592#	611#	646	1139#	1140#	1174	1188	1189	2631#	2674	2675#	2802#	
	2821	2845	2895	2951#	3037	3119	3248#	3249	3255	3358#				
.\$ERRT 001146	711#													
.\$X = 000500	587#	592												

. ABS. 022360 000

ERRORS DETECTED: 0

DSKZ:CZDLDC,DSKZ:CZDLDC.SEQ=DSKZ:CZDLDC.P11

RUN-TIME: 25 14 1 SECONDS

RUN-TIME RATIO: 357/41=8.6

CORE USED: 21K (41 PAGES)

DOCUMENT PAGES: 103