

KD11-Z
DL11-W

DL11-W/1144 MFM SLU
CZDLDE0

AH-8529E-MC
FICHE 1 OF 1

NOV 1980
COPYRIGHT © 75-80
MADE IN USA



.REM @

IDENTIFICATION

PRODUCT CODE: AC-8528E-MC
PRODUCT NAME: CZDLDEO DL11-W/1144 MFM SLU
PRODUCT DATE: JUNE 1980
MAINTAINER: DIAGNOSTIC ENGINEERING

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 COMPANIES.

COPYRIGHT (C) 1975, 1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

HISTORY SECTION

CZDLDDO WAS RELEASED OCT 79

THE FOLLOWING CHANGES WERE MADE. ALL CHANGES ARE INDICATED
BY ;** IN THE COMMENT FIELD:

1. USE THE MFPT INSTRUCTION, AND IF THE CPU IS A 11/44:
 - A. DO NOT PERFORM READER ENABLE AND RECEIVER ACTIVE TESTS.
 - B. IF ERROR FLAGS TESTS AND BREAK TESTS ARE ENABLED, PERFORM THESE TESTS ONLY FOR THE SLU AT 176500. PERFORM THESE TESTS FOR THE CONSOLE SLU IF BIT03 OF SWR IS ADDITIONALLY SET.
 - C. WHILE IN MAINTENANCE MODE DO NOT WRITE AND READ A *P OR AN ASCII 220. THIS WILL FORCE THE CONSOLE INTO 'CONSOLE MODE'.
 - D. IN THE CLOCK REPEATABILITY TEST, ALLOW FOR A TOLERANCE OF 2 BETWEEN CLOCK COUNTS. THIS IS TO ALLOW ENOUGH TOLERANCE WHEN MOS MEMORY IS USED AND REFRESH CYCLES ARE OCCURRING.
2. BECAUSE 11/44 SLU INTERRUPT REQUESTS ARE DEPENDANT ON A MFM CLOCK ENOUGH TIME MUST BE GIVEN FOR THEM TO OCCUR. THEREFORE, ALL TESTS ASSOCIATED WITH XMIT & RECEIVE INTERRUPTS SHOULD HAVE A MINIMUM OF 4 NOP'S ACTING AS WAIT FOR INTERRUPT.
3. IT WAS FOUND THAT AN ATTACHED TUS8 WOULD BE ACTIVATED BY THE DIAGNOSTIC, AND, AS A RESULT, CHARACTERS WOULD BE SENT TO THE RECEIVER BUFFER. THIS WOULD CAUSE SOME TESTS TO FAIL. THEREFORE TO INHIBIT ANY COMMUNICATION TO THE TUS8 FROM THE DIAGNOSTIC:
 - A. ENABLE MAINTENANCE MODE IN ALL TESTS THAT WRITE TO THE XMITTER.
 - B. IN ALL TESTS THAT WRITE TO XMITTER AND BEFORE LEAVING TEST: DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS THAT MIGHT BE IN THE PROCESS OF BEING XMITTED TO FINISH.
 - C. DISABLE MAINTENANCE MODE FOR PURPOSE OF ERROR PRINTING ONLY TO THE SLU WHICH THE TERMINAL IS ATTACHED.
4. IT WAS FOUND THAT UPON POWERUP AND WITH THE TUS8 ATTACHED, CHARACTERS WOULD BE SENT TO THE RECEIVER FROM THE TUS8. SOME RECEIVER TESTS WOULD FAIL DUE TO THIS. THEREFORE, ON ALL TESTS THAT PERFORM RECEIVER TESTS AND FOLLOWING MAINTENANCE MODE BEING ENABLED, DELAY ENOUGH TIME TO ALLOW TO ALLOW ANY CHARACTERS THAT MIGHT BE IN THE PROCESS OF BEING RECEIVED TO FINISH.

5. ADD SOFTWARE THAT WILL IMPLEMENT AUTO INITIATION OF 11/44 T/A CONSOLE TEST VIA WRAP CABLE FROM TU58 TO CONSOLE PORTS. IT IS SELECTED BY SWR BIT02 AND IS PERFORMED ONLY AFTER ALL SLUS ARE TESTED.

CZDLDE0 - EXTENDS DIAGNOSTIC SOFTWARE VECTOR ADDRESSING CAPABILITY TO 776 (OCTAL).

1.0 GENERAL INFORMATION

1.1 ABSTRACT

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE
THE FOLLOWING MODULES:

1. DL11-W SERIAL LINE/REAL TIME CLOCK INTERFACE
2. 11/44 MULTIFUNCTION MODULE

THE PROGRAM WILL RUN WITHOUT ANY SPECIAL TEST FIXTURES BY DEFAULT.
HOWEVER, THE FOLLOWING OPTIONAL TESTING IS PROVIDED:

1. TEST TO VERIFY XMIT AND RECEIVE OF THE UARTS WITH
A WRAP CABLE. THE UART UNDER TEST IS LOOPED BACK ON
ITSELF. THIS IS SELECTED VIA OPERATIONAL SWITCH SETTING
BIT 7, AND IS APPLICABLE TO THE DL11W AND 11/44 MFM.
2. AUTOMATIC INITIATION OF THE T/A CONSOLE TEST OF THE 11/44
MFM. THE TUSB PORTS ARE LOOPED TO THE CONSOLE PORTS
THIS IS SELECTED VIA OPERATIONAL SWITCH SETTING BIT 2
AND IS APPLICABLE TO 11/44 MFM ONLY.
(SEE CKKFBA0 FOR T/A CONSOLE TEST EXPLANATION)

THIS DIAGNOSTIC OPERATES ON THE CONSOLE SERIAL LINE AND CLOCK INTERFACES
AS WELL AS UP TO FIFTEEN(15) ADDITIONAL IDENTICALLY CONFIGURED
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
. IT CAN BE RUN UNDER XXDP,APT, AND ACT MONITORS,
AND ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER,
SOFTWARE 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.)

OPTIONAL:

1. LOOP CABLE FOR UART LOOP BACK ON ITSELF
2. WRAP CABLE FOR 11/4 MFM T/E TESTING
(SEE DWG. #7016942 WRAP AROUND CABLE
AND SECTION 5.6 OF THIS DOCUMENT)

1.2.2 STORAGE

THE PROGRAM USES 5K WORDS OF MEMORY

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. FOR THE DL11-W:

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).

FOR THE 11/44:

THE PROGRAM ASSUMES THAT THE ERROR FLAG BITS AND THE BREAK FUNCTION ARE DISABLED, AND WILL NOT TEST THESE FUNCTIONS HOWEVER, IF BIT 10 (FOR ERROR FLAGS) AND BIT 08 (FOR BREAK) OF SWR ARE SET, THEN ERROR FLAGS AND BREAK TESTS ARE PERFORMED FOR THE TUSB SLU ONLY. IF BIT03 OF SWR IS ALSO SET THEN THESE TESTS WILL BE ENABLED FOR THE CONSOLE.

READER ENABLE AND RECEIVER ACTIVE TESTS ARE NOT PERFORMED SINCE THE 11/44 MFM DOES NOT IMPLEMENT THESE FUNCTIONS.

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

NOTE: IF USING A CPU WITHOUT HARDWARE SWITCH REGISTER
SOFTWARE SWITCH REGISTER LOCATION = 176.
(FOR A 11/44 CPU USE MFM CONSOLE FOR DEPOSITING
SWITCH REGISTER. TYPE ^P TO ENTER CONSOLE)

- 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) [LCWER CASE ALPHA]

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

@ABCDEFGHIJKLMN O PQRSTU VWXYZ[\] [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

THE DIAGNOSTIC WILL CHECK FOR EXISTENCE OF SWITCH REGISTER AT 177570.

IF NO 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(LOC. 176) IS DESIRED, LOAD ALL 1'S INTO LOCATION 177570. (ALL SWITCHES UP IF PHYSICAL SWITCHES ARE AVAILABLE)

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)
BIT03	- FOR 11/44 MFM: ENABLE BOTH 'BREAK TESTS' AND 'ERROR FLAG TESTS' FOR THE CONSOLE SLU. (THIS BIT IS VALID ONLY IF BIT10 OR BIT08 IS SET.)
BIT02	- 11/44 MFM OPTION: SELECT AUTO INITIATION OF T/A CONSOLE TEST VIA WRAP CABLE

FOR DL11-W:

IF THE SOFTWARE SWITCH REGISTER IS USED(LOC. 176) THEN 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.

FOR 11/44 CPU:

SINCE THE 11/44 HAS A HARDWARE SWITCH REGISTER LOADED BY THE MFM CONSOLE THEN THE DIAGNOSTIC SHOULD ALWAYS FIND EXISTENCE OF 177570. WHEN OPERATING ON THE 11/44 CPU, DYNAMIC CHANGING OF SWREG(177570) DURING PROGRAM EXECUTION CAN BE ACCOMPLISHED BY USING THE MFM CONSOLE. TYPING ^P<CR> ON THE CONSOLE TTY WILL ENTER THE CONSOLE. THIS IS CONSIDERED "CONSOLE MODE". TO EXAMINE SWREG TYPE ' E SW<CR> ' TO THE CONSOLE PROMPT. TO LOAD THE SWREG TYPE ' D SW DATA<CR> ' WHERE "DATA" IS AN OCTAL NUMBER. IN ORDER FOR THE DIAGNOSTIC TO TYPE TO THE TTY IT IS NECESSARY TO HAVE THE 11/44 MFM IN "PROGRAM I/O MODE". THIS CAN BE ACCOMPLISHED BY TYPING ' C<CR> ' TO THE CONSOLE PROMPT. THEREFORE, WHEN CONSOLE USE IS COMPLETED, PLACE THE MFM IN "PROGRAM I/O MODE". BECAUSE THIS DIAGNOSTIC USES THE MAINTENANCE BIT, ^P WILL NOT BE ACKNOWLEDGED DURING THESE TESTS. THEREFORE, ISSUE ^P DURING PROGRAM TYPEOUTS. AT THIS TIME THE MAINTENANCE BIT WILL BE CLEARED.

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	10000000	400
7	01000000	200
6	00100000	100
5	00010000	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.

FOR DL11W:

THE DEFAULT EXECUTION TIMES PROVIDED(\$TSTM,\$PASTM) ARE FOR EXECUTION WITH AN 11/34 PROCESSOR, CORE MEMORY , AND 110 BAUD.

FOR 11/44:

THE EXECUTION TIMES PROVIDED IN THE APT SCRIPT THAT FOLLOWS AND IN SECT. 2.4 ARE FOR EXECUTION WITH A 11/44 PROCESSOR, CACHE, 16K CORE MEMORY, AND 300 BAUD.

THE FOLLOWING IS A PROGRAM LOAD FILE USED BY APT:

- E TABLE 'A' IS USED FOR APT DUMP MODE.
 - TWO SLU'S ARE TESTED.(\$SWREG BIT05=1 AND \$DEVN=3)
A SLU AT 177560(DEFAULT CONSOLE SLU) AND AT 176500
(BASE ADDRESS CODE).
 - THE ERROR FLAG AND BREAK TESTS ARE SELECTED FOR THE SLU AT 176500(TU58 SLU IN MFG.)
(\$SWREG BIT 10 AND 8 =1)

2. E TABLE 'B' IS USED FOR APT QV AND RUN TIME MODES. IT ACCOMPLISHES WHAT E TABLE 'A' DOES, BUT ADDITIONALLY IT SUPPRESSES ALL TYPEOUTS TO THE TERMINAL(\$ENVN=240) AND SELECTS AUTO TESTING OF T/A CONSOLE TEST VIA WRAP CABLE(\$SWREG BIT02=1).

1ST PASS RUN TIME	LONGEST TEST TIME	ADDITIONAL RUN TIME
60	50	45

.....	E TABLES
E-MODE/S-MODE (\$ENVM/\$ENV)	A 200/000	B 240/001
SWITCH REGISTER 1 (\$SWREG)	002440	002404
SWITCH REGISTER 2	000400	000400
CPU TYPE/OPTIONS	00/0000	00/0000
MEMORY MAP CODE 1	000/00000000	000/00000000
MEMORY MAP CODE 2	000/00000000	000/00000000
MEMORY MAP CODE 3	000/00000000	000/00000000
MEMORY MAP CODE 4	000/00000000	000/00000000
BUS PRIORITY/INTERRUPT 1	0000	0000
BUS PRIORITY/INTERRUPT 2	0000	0000
BUS ADDRESS CODE	176500	176500
DEVICE MAP CODE (\$DEVN)	000003	000003
CTLR. SPECIFIC WORD 1	000000	000000
CTLR. SPECIFIC WORD 2	000000	000000

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#15 --> 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 '\$DEVN' (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 '\$DEVN' 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

\$DEVN: 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 -

FOR DL11-W: (110 BAUD)
LONGEST SUBTEST TIME = 50 SECONDS
PASS TIME = 60 SECONDS
ADDITIONAL DEVICES = 55 SECONDS/DEVICE

FOR 11/44: (300 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 TIMEOUT (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 TIMEOUT. 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 TIMEOUT.

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 '1' 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.

5.0 AUTOMATIC INITIATION OF 11/44 T/A CONSOLE TEST
VIA WRAP CABLE

PURPOSE: THE T/E(OR T/A) CONSOLE TEST CAN BE IMPLEMENTED BY MANUALLY TYPING T/E(OR T/A) ON THE KEYBOARD OF THE TERMINAL WHEN IN CONSOLE MODE. IN ORDER TO IMPLEMENT THIS TEST IN AN AUTOMATIC WAY IN MANUFACTURING WHILE UNDER APT, THE USE OF A WRAP CABLE FROM THE TUSB TO THE CONSOLE CAN BE USED ALLOWING THIS DIAGNOSTIC TO ISSUE THE 'T/A' COMMAND TO THE CONSOLE. THE DIAGNOSTIC WILL ISSUE THE APPROPRIATE SEQUENCE OF COMMANDS TO THE CONSOLE VIA THE WRAP CABLE WHEN BIT02 OF SWR =1. THE DIAGNOSTIC WILL THEN MONITOR THE EXPECTED RESPONSE FROM THE CONSOLE VIA THE WRAP CABLE AND HALT IF THERE IS AN ERROR. THE T/A TEST IS DONE ONLY AFTER ALL SLUS ARE TESTED. IF T/A IS SUCCESSFUL 'END PASS' IS PRINTED AND THE DIAGNOSTIC STARTS AGAIN.

FIGURE 1 SHOWS THE PROPER WRAP CABLE SETUP AND REQUIREMENTS FOR AUTOMATICALLY INITIATING T/A FROM THE THE DIAGNOSTIC. NOTICE THAT THIS ARRANGEMENT ALLOWS FOR THE TERMINAL TO MONITOR ALL COMMUNICATION FROM THE CONSOLE OUTPUT DURING EXECUTION OF THE DIAGNOSTIC IN 'WRAP MODE'.

TYPEOUT EXAMPLES

1. WITH THE CONFIGURATION OF FIGURE 1 AND 'WRAP MODE' SELECTED THE PROGRAM MAY BE LOADED BY APT. IF 'E TABLE B' OF SECTION 2.3.3 WERE USED WITH THE EXCEPTION OF ALLOWING TYPEOUTS(\$ENV=200) THE FOLLOWING WOULD BE SEEN ON THE LOCAL TERMINAL:

```
CZDLDE0 DL11-W/1144 MFM SLU
02 DEVICES UNDER TEST ^P
CONSOLE
>>>T/A
```

```
CONSOLE-TESTB
END PASS ^P
CONSOLE
>>>T/A
```

```
CONSOLE-TESTB
END PASS ^P
CONSOLE
>>>T/A
```

```
CONSOLE-TESTB
END PASS ^P .....ETC.
```

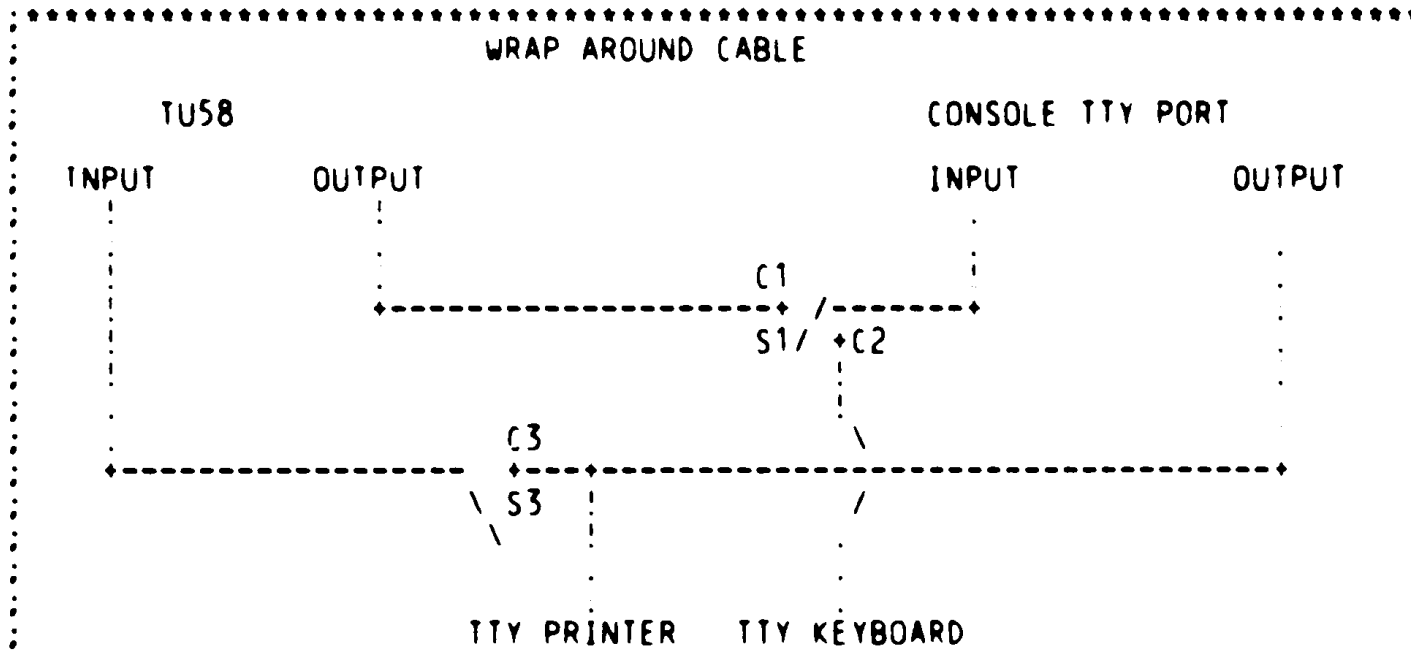
DESCRIPTION: REFERING TO THE ABOVE PRINTOUTS:

- A. THE DIAGNOSTIC IS LOADED WITH THE TITLE BEING PRINTED.
- B. TWO SLU DEVICES(CONSOLE,TU58) ARE SPECIFIED
- C. BOTH SLUS ARE TESTED COMPLETELY. AT THIS POINT THE THE DIAGNOSTIC ISSUES ^P TO THE WRAP CABLE. THE CONSOLE ENTERS CONSOLE MODE AND THE ^P TYPED ON THE TERMINAL IS THE MFM CONSOLE ECHO.
- D. THE CONSOLE PERFORMS ITS OWN SELF TEST BY TYPING 'CONSOLE' FOLLOWED BY >>>, THE CONSOLE PROMPT.
- E. THE DIAGNOSTIC THEN ISSUES T/A AND THE TERMINAL SHOWS THE CONSOLE ECHO OF THIS.
- F. THE MFM THEN PERFORMS THE T/A TEST SHOWN BY THE TERMINAL TYPING 'CONSOLE-TESTB'.THE DIAGNOSTIC LOOKS FOR THE 'B' IN THIS TYPEOUT,AND WHEN FOUND , CONSIDERS THE TEST A SUCCESS. THE DIAGNOSTIC WILL TYPE END PASS INDICATING A SUCCESSFUL PASS OF THE DIAGNOSTIC.
- G. THE DIAGNOSTIC CONTINUES WITH FURTHER PASSES OF THE DIAGNOSTIC.

2. IF 'E TABLE B' OF 2.3.3 WERE USED WITH TYPEOUTS SUPPRESSED (\$ENV=240) AS STATED, THEN THE FOLLOWING TYPEOUTS WOULD BE NOTICED:

```
^P  
CONSOLE  
>>>T/A  
  
CONSOLE-TESTB^P  
CONSOLE  
>>>T/A  
  
CONSOLE-TESTB^P  
CONSOLE  
>>>T/A  
  
CONSOLE-TESTB^P .....ETC.
```

FIG. 1- WRAPAROUND CONFIGURATION - AUTO INITIATION OF
11/44 T/A CONSOLE TEST



WHEN THE SWITCH IS IN 'NORMAL' MODE CONTACTS ARE:
S1-C2 IS CLOSED
S1-C1 IS OPEN
S3-C3 IS OPEN

THIS ALLOWS THE USE OF THE TERMINAL WITH THE CONSOLE

WHEN THE SWITCH IS IN 'WRAP' MODE CONTACTS ARE
S1-C1 CLOSED
S1-C2 OPEN
S2-C3 CLOSED

THIS ALLOWS THE TU58 PORT TO COMMUNICATE WITH THE CONSOLE PORT AND FOR
THE TTY TO MONITOR THE CONSOLE OUTPUT

THIS IS ALL DONE WITH A DPDT SWITCH

BEFORE THIS TEST CAN BE RUN THE CONSOLE UART, TTY, TU58 MUST BE AT
THE SAME BAUD RATE. THE CONSOLE AND TU58 UART, AND THE TTY SHOULD BE
SET UP WITH THE SAME STOP BIT SETTING,
WITH PARITY INHIBITED AND WITH A WORD LENGTH OF 7 OR 8 BITS

FOR MORE INFORMATION SEE THE MFM PRINTS AND THE WRAP AROUND
CABLE DRAWING NUMBER: 7016942

788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843

001100
104000
000004

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

000000
000040
000100
000140
000200
000240
000300
000340

100000
040000

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

STACK= 1100
ERROR=EMT
SCOPE=IOT

;*MISCELLANEOUS DEFINITIONS

HT= 11 ;;CODE FOR HORIZONTAL TAB
LF= 12 ;;CODE FOR LINE FEED
CR= 15 ;;CODE FOR CARRIAGE RETURN
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 ;;PROCESSOR STATUS WORD
PSW=PS
STKLMT= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570 ;;HARDWARE SWITCH REGISTER
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

R0= X0 ;;GENERAL REGISTER
R1= X1 ;;GENERAL REGISTER
R2= X2 ;;GENERAL REGISTER
R3= X3 ;;GENERAL REGISTER
R4= X4 ;;GENERAL REGISTER
R5= X5 ;;GENERAL REGISTER
R6= X6 ;;GENERAL REGISTER
R7= X7 ;;GENERAL REGISTER
SP= X6 ;;STACK POINTER
PC= X7 ;;PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS

PR0= 0 ;;PRIORITY LEVEL 0
PR1= 40 ;;PRIORITY LEVEL 1
PR2= 100 ;;PRIORITY LEVEL 2
PR3= 140 ;;PRIORITY LEVEL 3
PR4= 200 ;;PRIORITY LEVEL 4
PR5= 240 ;;PRIORITY LEVEL 5
PR6= 300 ;;PRIORITY LEVEL 6
PR7= 340 ;;PRIORITY LEVEL 7

;*''SWITCH REGISTER'' SWITCH DEFINITIONS

SW15= 100000
SW14= 40000

844	020000	SW13=	20000
845	010000	SW12=	10000
846	004000	SW11=	4000
847	002000	SW10=	2000
848	001000	SW09=	1000
849	000400	SW08=	400
850	000200	SW07=	200
851	000100	SW06=	100
852	000040	SW05=	40
853	000020	SW04=	20
854	000010	SW03=	10
855	000004	SW02=	4
856	000002	SW01=	2
857	000001	SW00=	1
858	001000		SW9=SW09
859	000400		SW8=SW08
860	000200		SW7=SW07
861	000100		SW6=SW06
862	000040		SW5=SW05
863	000020		SW4=SW04
864	000010		SW3=SW03
865	000004		SW2=SW02
866	000002		SW1=SW01
867	000001		SW0=SW00

868			
869		;*DATA BIT DEFINITIONS (BIT00 TO BIT15)	
870	100000	BIT15=	100000
871	040000	BIT14=	40000
872	020000	BIT13=	20000
873	010000	BIT12=	10000
874	004000	BIT11=	4000
875	002000	BIT10=	2000
876	001000	BIT09=	1000
877	000400	BIT08=	400
878	000200	BIT07=	200
879	000100	BIT06=	100
880	000040	BIT05=	40
881	000020	BIT04=	20
882	000010	BIT03=	10
883	000004	BIT02=	4
884	000002	BIT01=	2
885	000001	BIT00=	1
886	001000		BIT9=BIT09
887	000400		BIT8=BIT08
888	000200		BIT7=BIT07
889	000100		BIT6=BIT06
890	000040		BIT5=BIT05
891	000020		BIT4=BIT04
892	000010		BIT3=BIT03
893	000004		BIT2=BIT02
894	000002		BIT1=BIT01
895	000001		BIT0=BIT00

896			
897		;*BASIC "CPU" TRAP VECTOR ADDRESSES	
898	000004	ERRVEC=	4 ;:TIME OUT AND OTHER ERRORS
899	000010	RESVEC=	10 ;:RESERVED AND ILLEGAL INSTRUCTIONS

```

900      000014      TBITVEC=14      ;; 'T' BIT
901      000014      TRTVEC= 14      ;; TRACE TRAP
902      000014      BPTVEC= 14      ;; BREAKPOINT TRAP (BPT)
903      000020      IOTVEC= 20      ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
904      000024      PWRVEC= 24      ;; POWER FAIL
905      000030      EMTVEC= 30      ;; EMULATOR TRAP (EMT) **ERROR**
906      000034      TRAPVEC=34      ;; "TRAP" TRAP
907      000060      TKVEC= 60      ;; TTY KEYBOARD VECTOR
908      000064      TPVEC= 64      ;; TTY PRINTER VECTOR
909      000240      PIRQVEC=240     ;; PROGRAM INTERRUPT REQUEST VECTOR
910      000007      MFPT=7
911      176500      ABASE= 176500
912      000300      AVECT1= 300
913      000400      AUSWR= 400
914      000001      $TN= 1
915      161000      $SWR= 161000
916      000003      BPT= 000003      ; THIS IS THE COMMAND FOR A TRAP
917                                     ; THROUGH 14 (BPT TRAP)
918
919      000000      .=0
920      ;; *****
921      ;; *ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A '.+2,BPT'
922      ;; *SEQUENCE TO CATCH ILLEGAL TRAPS & INTERRUPTS
923      ;; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
924
925
926      000014      .=14      ; THE BPT TRAP VECTOR POINTS TO THE
927 000014 014552   .WORD   CATCH ; ILLEGAL TRAP HANDLER 'CATCH'
928 000016 000340   .WORD   340
929
930      000042      .- 42
931 000042 000000   .WORD   0
932
933
934
935
936      000174      .= 174
937 000174 000000   DISPREG: .WORD 0
938 000176 000000   SWREG:  .WORD 0
939
940      000200      .=200
941 000200 000137 003046  JMP     START ; DO INTERFACE TEST
942 000204 000137 017324  JMP     ECHO  ; DO ECHO TEST
943 000210 000137 017544  JMP     OUTST ; DO OUTPUT TEST TO TERMINAL

```

```

944
945          000500
946          .SBTTL  .=      500
947          .SBTTL  ACT11 HOOKS
948          ;*****
949          ;HOOKS REQUIRED BY ACT11
950          $SVPC=.          ;SAVE PC
951          .=46
952 000046 014450          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
953          .=52
954 000052 000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
955          .= $SVPC          ;; RESTORE PC
956          .SBTTL  APT PARAMETER BLOCK
957
958          ;*****
959          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
960          ;*****
961          . $X=.          ;;SAVE CURRENT LOCATION
962          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
963 000024 000200          200          ;;FOR APT START UP
964          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
965 000044 000500          $APTHDR          ;;POINT TO APT HEADER BLOCK
966          .= $X          ;;RESET LOCATION COUNTER
967          ;*****
968          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
969          ;INTERFACE SPEC.
970
971 000500          $APTHD:
972 000500 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
973 000502 001066          $MBADR: .WORD $MAIL          ;;ADDRESS OF APT MAILBOX (BITS 0-15)
974 000504 000050          $TSTM: .WORD 50          ;;RUN TIM OF LONGEST TEST
975 000506 000060          $PASTM: .WORD 60          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
976 000510 000055          $UNITM: .WORD 55          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
977 000512 000030          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
978

```



```
979          .SBTTL  COMMON TAGS
980
981          ;:*****
982          ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
983          ;*USED IN THE PROGRAM.
984
985          001000          .=1000
986          001000          $CMTAG:          ;;START OF COMMON TAGS
987          001000          000000          .WORD          0          ;;CONTAINS THE TEST NUMBER
988          001002          000          $*STNM: .BYTE          0          ;;CONTAINS ERROR FLAG
989          001003          000          $ERFLG: .BYTE          0          ;;CONTAINS SUBTEST ITERATION COUNT
990          001004          000000          $!CNT: .WORD          0          ;;CONTAINS SCOPE LOOP ADDRESS
991          001006          000000          $PADR: .WORD          J          ;;CONTAINS SCOPE RETURN FOR ERRORS
992          001010          000000          $LPERR: .WORD          0          ;;CONTAINS TOTAL ERRORS DETECTED
993          001012          000000          $ERTTL: .WORD          0          ;;CONTAINS ITEM CONTROL BYTE
994          001014          000          $ITEMB: .BYTE          0          ;;CONTAINS MAX. ERRORS PER TEST
995          001015          001          $ERMAX: .BYTE          1          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
996          001016          000000          $ERRPC: .WORD          0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
997          001020          000000          $GDADR: .WORD          0          ;;CONTAINS ADDRESS OF 'BAD' DATA
998          001022          000000          $BDADR: .WORD          0          ;;CONTAINS 'GOOD' DATA
999          001024          000000          $GDDAT: .WORD          0          ;;CONTAINS 'BAD' DATA
1000         001026          000000          $BDDAT: .WORD          0          ;;RESERVED--NOT TO BE USED
1001         001030          000000          .WORD          0
1002         001032          000000          .WORD          0
1003         001034          000          $AUTOB: .BYTE          0          ;;AUTOMATIC MODE INDICATOR
1004         001035          000          $INTAG: .BYTE          0          ;;INTERRUPT MODE INDICATOR
1005         001036          000000          .WORD          0
1006         001040          177570          $SWR: .WORD          DSWR          ;;ADDRESS OF SWITCH REGISTER
1007         001042          177570          $DISPLAY: .WORD          DDISP          ;;ADDRESS OF DISPLAY REGISTER
1008         001044          177560          $TKS: 177560          ;;TTY KBD STATUS
1009         001046          177562          $TKB: 177562          ;;TTY KBD BUFFER
1010         001050          177564          $TPS: 177564          ;;TTY PRINTER STATUS REG. ADDRESS
1011         001052          177566          $TPB: 177566          ;;TTY PRINTER BUFFER REG. ADDRESS
1012         001054          000          $NULL: .BYTE          0          ;;CONTAINS NULL CHARACTER FOR FILLS
1013         001055          002          $FILLS: .BYTE          2          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
1014         001056          012          $FILLC: .BYTE          12          ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
1015         001057          000          $TPFLG: .BYTE          0          ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
1016         001060          000000          $ESCAPE:0          ;;ESCAPE ON ERROR ADDRESS
1017         001062          077          $QUES: .ASCII          /?/          ;;QUESTION MARK
1018         001063          015          $CRLF: .ASCII          <15>          ;;CARRIAGE RETURN
1019         001064          000012          $LF: .ASCII           <12>          ;;LINE FEED
1020         ;:*****
1021         .SBTTL  APT MAILBOX-ETABLE
1022
1023         ;:*****
1024         .EVEN
1025         001066          $MAIL:          ;;APT MAILBOX
1026         001066          000000          $MSGTY: .WORD          MSGTY          ;;MESSAGE TYPE CODE
1027         001070          000J00          $FATAL: .WORD          AFATAL          ;;FATAL ERROR NUMBER
1028         001072          000000          $TESTN: .WORD          ATESTN          ;;TEST NUMBER
1029         001074          000000          $PASS: .WORD          APASS          ;;PASS COUNT
1030         001076          000000          $DEVCT: .WORD          ADEVCT          ;;DEVICE COUNT
1031         001100          000000          $UNIT: .WORD          AUNIT          ;;I/O UNIT NUMBER
1032         001102          000000          $MSGAD: .WORD          AMSGAD          ;;MESSAGE ADDRESS
1033         001104          000000          $MSGLG: .WORD          AMGLG          ;;MESSAGE LENGTH
1034         001106          $ETABLE:          ;;APT ENVIRONMENT TABLE
```

1035	001106	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
1036	001107	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
1037	001110	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
1038	001112	000400	\$USWR:	.WORD	AUSWR	::USER SWITCHES
1039	001114	000000	\$CPUOP:	.WORD	ACPUOP	::CPU TYPE,OPTIONS
1040			*			BITS 15-11=CPU TYPE
1041			*			11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
1042			*			11/70=06,PDQ=07,Q=10
1043			*			BIT 10=REAL TIME CLOCK
1044			*			BIT 9=FLOATING POINT PROCESSOR
1045			*			BIT 8=MEMORY MANAGEMENT
1046	001116	000	\$MAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
1047	001117	000	\$MTYP1:	.BYTE	AMTYP1	::MEM. TYPE,BLK#1
1048			*			MEM.TYPE BYTE -- (HIGH BYTE)
1049			*			900 NSEC CORE=001
1050			*			300 NSEC BIPOLAR=002
1051			*			500 NSEC MOS=003
1052	001120	000000	\$MADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
1053			*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
1054	001122	000	\$MAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
1055	001123	000	\$MTYP2:	.BYTE	AMTYP2	::MEM. TYPE,BLK#2
1056	001124	000000	\$MADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
1057	001126	000	\$MAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
1058	001127	000	\$MTYP3:	.BYTE	AMTYP3	::MEM. TYPE,BLK#3
1059	001130	000000	\$MADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
1060	001132	000	\$MAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
1061	001133	000	\$MTYP4:	.BYTE	AMTYP4	::MEM. TYPE,BLK#4
1062	001134	000000	\$MADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
1063	001136	000300	\$VECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
1064	001140	000000	\$VECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
1065	001142	176500	\$BASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
1066	001144	000000	\$DEVM:	.WORD	ADEVM	::DEVICE MAP
1067	001146		\$ETEND:			
1068			.MEXIT			

ERROR POINTER TABLE

1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

.\$ERRTB:

EM1	::'CAN NOT ACCESS TCSR'
DH1	::'TEST# ERR PC TCSR'
DT1	::\$TESTN,\$ERRPC,TCSR
0	
EM2	::'CAN NOT ACCESS TBUF'
DH2	::'TEST# ERR PC TBUF'
DT2	::\$TESTN,\$ERRPC,TBUF
0	
EM3	::'TCSR DONE NOT CLEARED WITH TBUF FULL'
DH1	::'TEST# ERR PC TCSR'
DT1	::\$TESTN,\$ERRPC,TCSR
0	
EM4	::'TCSR DONE NOT SET'
DH1	::'TEST# ERR PC TCSR'
DT1	::\$TESTN,\$ERRPC,TCSR
0	
EM5	::TCSR DONE NOT SET WITH RESET
DH1	::'TEST# ERR PC TCSR'
DT1	::\$TESTN,\$ERRPC,TCSR
0	
EM6	::'CAN NOT ACCESS RCSR'
DH6	::'TEST# ERR PC RCSR'
DT6	::\$TESTN,\$ERRPC,RCSR
0	
EM7	::'CAN NOT ACCESS RBUF'
DH7	::'TEST# ERR PC RBUF'
DT7	::\$TESTN,\$ERRPC,RBUF
0	
EM10	::'CAN NOT ACCESS LKS'
DH10	::'TEST# ERR PC LKS'
DT10	::\$TESTN,\$ERRPC,LKS
0	

1125					
1126	001246	020117	EM11	:	'BIT0 OF TCSR NOT CLEAR AFTER RESET''
1127	001250	024012	DH1	:	'TEST# ERR PC TCSR''
1128	001252	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1129	001254	000000	0		
1130					
1131	001256	020162	EM12	:	'CAN NOT SET BIT0 OF TCSR''
1132	001260	024012	DH1	:	'TEST# ERR PC TCSR''
1133	001262	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1134	001264	000000	0		
1135					
1136	001266	020213	EM13	:	'CAN NOT CLEAR BIT0 OF TCSR''
1137	001270	024012	DH1	:	'TEST# ERR PC TCSR''
1138	001272	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1139	001274	000000	0		
1140					
1141	001276	020246	EM14	:	'RESET DID NOT CLEAR BIT0 OF TCSR''
1142	001300	024012	DH1	:	'TEST# ERR PC TCSR''
1143	001302	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1144	001304	000000	0		
1145					
1146	001306	020307	EM15	:	'BIT2 OF TCSR NOT CLEAR AFTER RESET''
1147	001310	024012	DH1	:	'TEST# ERR PC TCSR''
1148	001312	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1149	001314	000000	0		
1150					
1151	001316	020352	EM16	:	'CAN NOT SET BIT2 OF TCSR''
1152	001320	024012	DH1	:	'TEST# ERR PC TCSR''
1153	001322	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1154	001324	000000	0		
1155					
1156	001326	020403	EM17	:	'CAN NOT CLEAR BIT2 OF TCSR''
1157	001330	024012	DH1	:	'TEST# ERR PC TCSR''
1158	001332	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1159	001334	000000	0		
1160					
1161	001336	020436	EM20	:	'RESET DID NOT CLEAR BIT2 OF TCSR''
1162	001340	024012	DH1	:	'TEST# ERR PC TCSR''
1163	001342	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1164	001344	000000	0		
1165					
1166	001346	020477	EM21	:	'BIT6 OF TCSR NOT CLEAR AFTER RESET2
1167	001350	024012	DH1	:	'TEST# ERR PC TCSR''
1168	001352	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1169	001354	000000	0		
1170					
1171	001356	020542	EM22	:	'XMIT INTERRUPT WITH PRIORITY 7''
1172	001360	024012	DH1	:	'TEST# ERR PC TCSR''
1173	001362	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1174	001364	000000	0		
1175					
1176	001366	020577	EM23	:	'CAN NOT SET BIT6 OF TCSR''
1177	001370	024012	DH1	:	'TEST# ERR PC TCSR''
1178	001372	024572	DT1	:	\$TESTN,\$ERRPC,TCSR
1179	001374	000000	0		
1180					

1181	001376	020630	EM24	:'CAN NOT CLEAR BIT6 OF TCSR'
1182	001400	024012	DH1	:'TEST# ERR PC TCSR'
1183	001402	024572	DT1	:\$TESTN,\$ERRPC,TCSR
1184	001404	000000	0	
1185				
1186	001406	020665	EM25	:'RESET DID NOT CLEAR BIT6 OF TCSR'
1187	001410	024012	DH1	:'TEST# ERR PC TCSR'
1188	001412	024572	DT1	:\$TESTN,\$ERRPC,TCSR
1189	001414	000000	0	
1190				
1191	001416	020724	EM26	:'BIT6 OF RCSR NOT CLEAR AFTER RESET'
1192	001420	024064	DH6	:'TEST# ERR PC RCSR'
1193	001422	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1194	001424	000000	0	
1195				
1196	001426	020767	EM27	:'RCVR INTERRUPT WITH PRIORITY 7'
1197	001430	024064	DH6	:'TEST# ERR PC RCSR'
1198	001432	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1199	001434	000000	0	
1200				
1201	001436	021026	EM30	:'CAN NOT SET BIT6 OF RCSR'
1202	001440	024064	DH6	:'TEST# ERR PC RCSR'
1203	001442	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1204	001444	000000	0	
1205				
1206	001446	021057	EM31	:'CAN NOT CLEAR BIT6 OF RCSR'
1207	001450	024064	DH6	:'TEST# ERR PC RCSR'
1208	001452	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1209	001454	000000	0	
1210				
1211	001456	021112	EM32	:'CAN NOT CLEAR BIT6 OF RCSR WITH RESET2
1212	001460	024064	DH6	:'TEST# ERR PC RCSR'
1213	001462	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1214	001464	000000	0	
1215				
1216	001466	021160	EM33	:'BIT6 OF LKS NOT CLEAR AFTER RESET'
1217	001470	024136	DH10	:'TEST# ERR PC LKS'
1218	001472	024632	DT10	:\$TESTN,\$ERRPC,LKS
1219	001474	000000	0	
1220				
1221	001476	021222	EM34	:'LKS INTERRUPT WITH PRIORITY 7'
1222	001500	024136	DH10	:'TEST# ERR PC LKS'
1223	001502	024632	DT10	:\$TESTN,\$ERRPC,LKS
1224	001504	000000	0	
1225				
1226	001506	021260	EM35	:'CAN NOT SET BIT6 OF LKS'
1227	001510	024136	DH10	:'TEST# ERR PC LKS'
1228	001512	024632	DT10	:\$TESTN,\$ERRPC,LKS
1229	001514	000000	0	
1230				
1231	001516	021310	EM36	:'CAN NOT CLEAR BIT6 OF LKS'
1232	001520	024136	DH10	:'TEST# ERR PC LKS'
1233	001522	024632	DT10	:\$TESTN,\$ERRPC,LKS
1234	001524	000000	0	
1235				
1236	001526	021342	EM37	:'RESET DID NOT CLEAR BIT6 OF LKS'

1237	001530	024136	DH10	:'TEST# ERR PC LKS''
1238	001532	024632	DT10	:\$TESTN,\$ERRPC,LKS
1239	001534	000000	0	
1240				
1241	001536	021402	EM40	:'DUAL ADDRESSING ERROR''
1242	001540	024162	DH40	:'TEST# ERR PC GOOD BAD''
1243	001542	024642	DT40	:\$TESTN,\$ERRPC,\$GDADR,\$BDCSR
1244	001544	000000	0	
1245				
1246	001546	021430	EM41	:'BIT7 OF LKS NOT SET AFTER RESET2
1247	001550	024136	DH10	:'TEST# ERR PC LKS''
1248	001552	024632	DT10	:\$TESTN,\$ERRPC,LKS
1249	001554	000000	0	
1250				
1251	001556	021470	EM42	:'CAN NOT CLEAR BIT7 OF LKS''
1252	001560	024136	DH10	:'TEST# ERR PC LKS''
1253	001562	024632	DT10	:\$TESTN,\$ERRPC,LKS
1254	001564	000000	0	
1255				
1256	001566	021522	EM43	:'BIT7 OF LKS DOES NOT SET''
1257	001570	024136	DH10	:'TEST# ERR PC LKS''
1258	001572	024632	DT10	:\$TESTN,\$ERRPC,LKS
1259	001574	000000	0	
1260				
1261	001576	021553	EM44	:'RTC INTERRUPT AT PRIORITY 7''
1262	001600	024136	DH10	:'TEST# ERR PC LKS''
1263	001602	024632	DT10	:\$TESTN,\$ERRPC,LKS
1264	001604	000000	0	
1265				
1266	001606	021607	EM45	:'RTC INTERRUPTS WHEN DISABLED''
1267	001610	024136	DH10	:'TEST# ERR PC LKS''
1268	001612	024632	DT10	:\$TESTN,\$ERRPC,LKS
1269	001614	000000	0	
1270				
1271	001616	021644	EM46	:'RTC INTERRUPT DID NOT OCCUR''
1272	001620	024136	DH10	:'TEST# ERR PC LKS''
1273	001622	024632	DT10	:\$TESTN,\$ERRPC,LKS
1274	001624	000000	0	
1275				
1276	001626	021644	EM47	:'RTC INTERRUPT DID NOT OCCUR''
1277	001630	024136	DH10	:'TEST# ERR PC LKS''
1278	001632	024632	DT10	:\$TESTN,\$ERRPC,LKS
1279	001634	000000	0	
1280				
1281	001636	021700	EM50	:'RTC DOUBLE INTERRUPT''
1282	001640	024136	DH10	:'TEST# ERR PC LKS''
1283	001642	024632	DT10	:\$TESTN,\$ERRPC,LKS
1284	001644	000000	0	
1285				
1286	001646	021725	EM51	:'RESET DID NOT CLEAR RTC INTERRUPT''
1287	001650	024136	DH10	:'TEST# ERR PC LKS''
1288	001652	024632	DT10	:\$TESTN,\$ERRPC,LKS
1289	001654	000000	0	
1290				
1291	001656	021755	EM52	:'RTC INTERRUPT DID NOT CLEAR WITH BIT7 OF LKS''
1292	001660	024136	DH10	:'TEST# ERR PC LKS''

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 31
 CZDLDE.P11 10-JUN-80 15:20 ERROR POINTER TABLE

SEQ 0030

1293	001662	024632	DT10	;\$TESTN,\$ERRPC,LKS
1294	001664	000000	0	
1295				
1296	001666	022032	EM53	
1297	001670	024216	DH53	;'TEST# ERR PC LKS CNT1 CNT2'
1298	001672	024654	DT53	;\$TESTN,\$ERRPC,LKS,FIRST,SECND
1299	001674	000000	0	
1300				
1301	001676	022056	EM54	;'XMIT INTERRUPTS WHEN DISABLED'
1302	001700	024012	DH1	;'TEST# ERR PC TCSR'
1303	001702	024572	DT1	;\$TESTN,\$ERRPC,TCSR
1304	001704	000000	0	
1305				
1306	001706	022214	EM55	;'XMIT DID NOT INTERRUPT'
1307	001710	024012	DH1	;'TEST# ERR PC TCSR'
1308	001712	024572	DT1	;\$TESTN,\$ERRPC,TCSR
1309	001714	000000	0	
1310				
1311	001716	022114	EM56	;'XMIT INTERRUPT AT PRIORITY 7'
1312	001720	024012	DH1	;'TEST# ERR PC TCSR'
1313	001722	024572	DT1	;\$TESTN,\$ERRPC,TCSR
1314	001724	000000	0	
1315				
1316	001726	022152	EM57	;'XMIT INTERRUPTS WITH ENABLE CLEAR'
1317	001730	024012	DH1	;'TEST# ERR PC TCSR'
1318	001732	024572	DT1	;\$TESTN,\$ERRPC,TCSR
1319	001734	000000	0	
1320				
1321	001736	022214	EM60	;'XMIT DID NOT INTERRUPT'
1322	001740	024012	DH1	;'TEST# ERR PC TCSR'
1323	001742	024572	DT1	;\$TESTN,\$ERRPC,TCSR
1324	001744	000000	0	
1325				
1326	001746	022243	EM61	;'XMIT RE-INTERRUPTED'
1327	001750	024012	DH1	;'TEST# ERR PC TCSR'
1328	001752	024572	DT1	;\$TESTN,\$ERRPC,TCSR
1329	001754	000000	0	
1330				
1331	001756	022267	EM62	;'LOADING TBUF DID NOT CLEAR INTERRUPT'
1332	001760	024012	DH1	;'TEST# ERR PC TCSR'
1333	001762	024572	DT1	;\$TESTN,\$ERRPC,TCSR
1334	001764	000000	0	
1335				
1336	001766	022334	EM63	;'RCVR ACTIVE NOT SET'
1337	001770	024064	DH6	;'TEST# ERR PC RCSR'
1338	001772	024612	DT6	;\$TESTN,\$ERRPC,RCSR
1339	001774	000000	0	
1340				
1341	001776	022360	EM64	;'RECEIVER DONE NEVER SET'
1342	002000	024064	DH6	;'TEST# ERR PC RCSR'
1343	002002	024612	DT6	;\$TESTN,\$ERRPC,RCSR
1344	002004	000000	0	
1345				
1346	002006	022404	EM65	;'RCVR ACTIVE NOT CLEARED WITH DONE SET2'
1347	002010	024064	DH6	;'TEST# ERR PC RCSR'
1348	002012	024612	DT6	;\$TESTN,\$ERRPC,RCSR

1349	002014	000000	0	
1350				
1351	002016	022452	EM66	:'RESET DID NOT CLEAR RCVR DONE''
1352	002020	024064	DH6	:'TEST# ERR PC RCSR''
1353	002022	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1354	002024	000000	0	
1355				
1356	002026	022510	EM67	:'RDR ENABLE SET DID NOT CLEAR RCVR DONE''
1357	002030	024064	DH6	:'TEST# ERR PC RCSR''
1358	002032	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1359	002034	000000	0	
1360				
1361	002036	022553	EM70	:'READING RBUF DID NOT CLEAR RCVR DONE''
1362	002040	024064	DH6	:'TEST# ERR PC RCSR''
1363	002042	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1364	002044	000000	0	
1365				
1366	002046	022620	EM71	:'RCVR INTERRUPTS WITH ENABLE CLEAR''
1367	002050	024064	DH6	:'TEST# ERR PC RCSR''
1368	002052	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1369	002054	000000	0	
1370				
1371	002056	022767	EM72	:'RCVR DID NOT INTERRUPT''
1372	002060	024064	DH6	:'TEST# ERR PC RCSR''
1373	002062	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1374	002064	000000	0	
1375				
1376	002066	022662	EM73	:'RCVR INTERRUPTS AT PRIORITY 7''
1377	002070	024064	DH6	:'TEST# ERR PC RCSR''
1378	002072	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1379	002074	000000	0	
1380				
1381	002076	022720	EM74	:'RCVR INTERRUPT REQUEST PASSED WITH ENABLE CLEAR''
1382	002100	024064	DH6	:'TEST# ERR PC RCSR''
1383	002102	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1384	002104	000000	0	
1385				
1386	002106	022767	EM75	:'RCVR DID NOT INTERRUPT''
1387	002110	024064	DH6	:'TEST# ERR PC RCSR''
1388	002112	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1389	002114	000000	0	
1390	002116	023016	EM76	:'RECEIVER RE-INTERRUPTED''
1391	002120	024064	DH6	:'TEST# ERR PC RCSR''
1392	002122	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1393	002124	000000	0	
1394				
1395	002126	023042	EM77	:'READING RBUF DID NOT CLEAR INTERRUPT''
1396	002130	024064	DH6	:'TEST# ERR PC RCSR''
1397	002132	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1398	002134	000000	0	
1399				
1400	002136	023107	EM100	:'RESET DID NOT CLEAR RCVR INTERRUPT''
1401	002140	024064	DH6	:'TEST# ERR PC RCSR''
1402	002142	024612	DT6	:\$TESTN,\$ERRPC,RCSR
1403	002144	000000	0	
1404				

1405				
1406	002146	023152	EM101	;'OR' FLAG DID NOT SET'
1407	002150	024064	DH6	
1408	002152	024612	DT6	;\$TESTN,\$ERRPC,RCSR
1409	002154	000000	0	
1410				
1411	002156	023200	EM102	;'ERROR' NOT SET WITH 'OR' FLAG''
1412	002160	024064	DH6	;'TEST# ERR PC RCSR''
1413	002162	024612	DT6	;\$TESTN,\$ERRPC,RCSR
1414	002164	000000	0	
1415	002166	023237	EM103	;'BREAK DID NOT TRANSMIT ALL ZEROES''
1416	002170	024263	DH103	;'TEST# ERR PC RCSR DATA''
1417	002172	024670	DT103	;\$TESTN,\$ERRPC,RCSR,\$BDDAT
1418	002174	000000	0	
1419				
1420	002176	023275	EM104	;'BREAK DID NOT SET FRAMING EPROR''
1421	002200	024064	DH6	;'TEST# ERR PC RCSR''
1422	002202	024612	DT6	;\$TESTN,\$ERRPC,RCSR
1423	002204	000000	0	
1424				
1425	002206	023332	EM105	;'DATA COMPARE ERROR''
1426	002210	024320	DH105	;'TEST# ERR PC RCSR GOOD BAD''
1427	002212	024702	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1428	002214	000000	0	
1429				
1430	002216	023355	EM106	;'DATA COMPARE ERROR WITH CABLE''
1431	002220	024320	DH105	;'TEST# ERR PC RCSR GOOD BAD''
1432	002222	024702	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1433	002224	000000	0	
1434				
1435	002226	023413	EM107	;'TIMEOUT IN EXERCISER TEST''
1436	002230	024064	DH6	;'TEST# ERR PC RCSR''
1437	002232	024612	DT6	;\$TESTN,\$ERRPC,RCSR
1438	002234	000000	0	
1439				
1440	002236	023445	EM110	;'INCORRECT RECEIVE COUNT
1441	002240	024364	DH110	;'TEST# ERR PC RCSR TRANS RCV''
1442	002242	024716	DT110	;\$TESTN,\$ERRPC,RCSR,XMTCNT,RCVCNT
1443	002244	000000	0	
1444				
1445	002246	023475	EM111	;'DATA COMPARE ERROR IN EXERCISER''
1446	002250	024320	DH105	;'TEST# ERR PC RCSR GOOD BAD''
1447	002252	024702	DT105	;\$TESTN,\$ERRPC,RCSR,GD,BD
1448	002254	000000	0	
1449				
1450	002256	023535	EM112	;'TRAP CATCHER''
1451	002260	024430	DH112	;'TEST# ERR PC RCSR OLDPC TRAP ADP''
1452	002262	024732	DT112	;\$TESTN,\$ERRPC,RCSR,OLDPC,BDVECT
1453	002264	000000	0	
1454				
1455	002266	023552	EM113	;'NO CLK INTERRUPT IN EXERCISER''
1456	002270	024136	DH10	;'TEST# ERR PC LKS''
1457	002272	024632	DT10	;\$TESTN,\$ERRPC,LKS
1458	002274	000000	0	
1459				
1460	002276	023610	EM111'	;'ERROR' NOT SET WITH 'FR' FLAG''

1461	002300	024064				DH6	;'TEST# ERR PC RCSR'
1462	002302	024612				DT6	;\$TESTN,\$ERRPC,RCSR
1463	002304	000000				0	
1464							
1465	002306	023647				EM115	;RCV ACTVIE NOT CLEAR WITH INIT
1466	002310	024064				DH6	;'TEST# ERR PC RCSR'
1467	002312	024612				DT6	;\$TESTN,\$ERRPC,RCSR
1468	002314	000000				0	
1469							
1470	002316	023706				EM116	;RCV ACTIVE WITHOUT 'START' BIT
1471	002320	024064				DH6	;'TEST# ERR PC RCSR'
1472	002322	024612				DT6	;\$TESTN,\$ERRPC,RCSR
1473	002324	000000				0	
1474							
1475	002326	023745				EM117	;RDR ENABLE NOT CLEAR WITH RCV ACTIVE
1476	002330	024064				DH6	;'TEST# ERR PC RCSR'
1477	002332	024612				DT6	;\$TESTN,\$ERRPC,RCSR
1478	002334	000000				0	
1479							
1480							;STORAGE LOCATIONS
1481	002336	000000				FIRST: .WORD	0
1482	002340	000000				LOC1: .WORD	0
1483	002342	000000				LOC2: .WORD	0
1484	002344	000000				SAVE0: .WORD	0
1485	002346	000022				SAVLOC: .BLKW	22
1486	002412	000010				ENDSTK: .BLKW	10
1487	002432	000000				JIMSTK: .WORD	0
1488	002434	000000				SAVEPS: .WORD	0
1489	002436	000000				OLDSUM: .WORD	0
1490	002440	000000				RCVCNT: .WORD	0
1491	002442	000000				XMTCNT: .WORD	0
1492	002444	000000				CLKCNT: .WORD	0
1493	002446	000000				STPSW: .WORD	0
1494	002450	000000				SRPSW: .WORD	0
1495	002452	000000				SCPSW: .WORD	0
1496	002454	000044				BUF: .BLKW	44
1497	002564	000020				CNTLP: .ASCIZ	<20>
1498	002566	050136	006400	041412		PROMPT: .ASCII	/*P/<0><CR><LF>/CONSOLE/<CR><LF>/>>>/<377>
1499	002574	047117	047523	042514			
1500	002602	005015	037076	177476			
1501	002610	027524	006501	000012		TA: .ASCIZ	\$T/\$S<CR><LF>
1502							
1503						.EVEN	
1504	002616	000000				SECND: .WORD	0
1505	002620	000000				OLDPC: .WORD	0
1506	002622	000000				BDVECT: .WORD	0
1507							
1508							
1509							
1510							
1511							
1512	002624	000000				CTSTFL: .WORD	0
1513	002626	000000				TMP1: .WORD	0
1514	002630	000000				TMP2: .WORD	0
1515	002632	000000				TMP3: .WORD	0
1516							;REGISTER AND VECTOR ADDRESSES FOR THE DL-11W UNDER TEST

```
;'TEST# ERR PC RCSR'
;$TESTN,$ERRPC,RCSR

;RCV ACTVIE NOT CLEAR WITH INIT
;'TEST# ERR PC RCSR'
;$TESTN,$ERRPC,RCSR

;RCV ACTIVE WITHOUT 'START' BIT
;'TEST# ERR PC RCSR'
;$TESTN,$ERRPC,RCSR

;RDR ENABLE NOT CLEAR WITH RCV ACTIVE
;'TEST# ERR PC RCSR'
;$TESTN,$ERRPC,RCSR

;TIMER LOOP COUNTER
;TIMER LOOP COUNTER
;STORAGE FOR LOCATIONS
;THAT T/E USES
;JIMS SPECIAL STACK (WRAP AROUND)

;SAVE PSW AREA
;CHECKSUM STORAGE

;CONSLE UNDER TEST FLAG
;TEMP LOCATION FOR TABLE OFFSETS
;TEMP LOCATION FOR DEVICE COUNT
;LOCATION FOR DEVICE MAP BIT TEST MASK
```

```

1517
1518 002634 000000 RCSR: .WORD 0
1519 002636 000000 RBUF: .WORD 0
1520 002640 000000 TCSR: .WORD 0
1521 002642 000000 TBUF: .WORD 0
1522 002644 000000 RVECT: .WORD 0
1523 002646 000000 RPSW: .WORD 0
1524 002650 000000 TVECT: .WORD 0
1525 002657 000000 TPSW: .WORD 0
1526
1527 ;CONSOLE REGISTER AND VECTOR ADDRESSES FOR THE DL-11W
1528
1529 002654 177560 CRCSR: 177560 ;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
1530 002656 177562 CRBUF: 177562 ;ADDRESS OF RECEIVER BUFFER
1531 002660 177564 CTCRSR: 177564 ;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
1532 002662 177566 CTBUF: 177566 ;ADDRESS OF TRANSMITTER BUFFER
1533 002664 000060 CRVECT: 60 ;RECEIVER INTERRUPT VECTOR
1534 002666 000062 CRPSW: 62
1535 002670 000064 CTVECT: 64 ;TRANSMITTER INTERRUPT VECTOR
1536 002672 000066 CTPSW: 66
1537
1538 ;REAL TIME CLOCK REGISTER AND VECTOR ADDRESSES
1539 002674 177546 LKS: .WORD 177546
1540 002676 000100 RTCVT: .WORD 100
1541 002700 000102 RTCPSW: .WORD 102
1542
1543 002702 000020 ADRTBL: .BLKW 20
1544 002742 000020 VCTTBL: .BLKW 20
1545 003002 000000 FLAG44: .WORD 0
1546 003004 176500 TURCSR: .WORD 176500
1547 003006 176502 TURBUF: .WORD 176502
1548 003010 176504 TUTCSR: .WORD 176504
1549 003012 176506 TUTBUF: .WORD 176506
1550
1551
1552 ;SUBROUTINE TO GENERATE DEVICE ADDRESS TABLE
1553
1554 003014 012702 002702 DEVADR: MOV #ADRTBL,R2 ;POINT R2 TO THE DEVICE ADDRESS TABLE
1555 003020 013700 001142 MOV $BASE,R0 ;LOAD BASE DEVICE ADDRESS IN R0
1556 003024 010001 MOV R0,R1 ;
1557 003026 062701 000170 ADD #170,R1 ;POINT R1 TO LAST DEVICE ADDRESS
1558 003032 010022 1$: MOV R0,(R2)+ ;MOVE DEVICE ADDRESS TO TABLE
1559 003034 062700 000010 ADD #10,R0 ;POINT R0 TO NEXT DEVICE ADDRESS
1560 003040 020001 CMP R0,R1 ;FINISHED GENERATING TABLE?
1561 003042 003773 BLE 1$ ;BR, IF LAST DEVICE ADDRESS NOT LOADED
1562 003044 000207 RTS PC
1563
1564
1565
1566 003046 005037 001070 START: CLR $FATAL ;CLEAR ERROR NO.
1567 003052 005037 001066 CLR $MSGTYP ;CLEAR MESSAGE TYPE
1568 003056 005037 001072 CLR $TESTN ;CLEAR TEST NO.
1569 003062 005037 002624 CLR CTSTFL ;CLEAR CONSOLE UNDER TEST FLAG
1570 003066 005037 001076 CLR $DEVCT ;CLEAR DEVICE COUNT
1571 003072 005037 001100 CLR $UNIT ;CLEAR UNIT NUMBER
1572 003076 005037 003002 CLR FLAG44 ;** CLEAR 11/44 CPU FLAG
  
```

ERROR POINTER TABLE

```

1573 003102 005737 001112          TST    $USWR          ;IS $USWR LOADED?
1574 003106 001003                   BNE    1$            ;BR IF YES
1575 003110 012737 000400 001112    MOV    #400,$USWR    ;ELSE, DEFAULT TO $USWR-400
1576 003116 012737 000006 000004 1$:  MOV    #6,@#4        ;INITIALIZE TIMEOUT VECTORS TO TRAP
1577 003124 012737 000003 000006    MOV    #3,@#6        ; CATCHER ROUTINE
1578
1579
1580          .SBTTL  INITIALIZE THE COMMON TAGS
          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1581 003132 012706 001000          MOV    # $CMTAG,R6   ;;FIRST LOCATION TO BE CLEARED
1582 003136 005026                   CLR    (R6)+         ;;CLEAR MEMORY LOCATION
1583 003140 022706 001040          CMP    #SWR,R6      ;;DONE?
1584 003144 001374                   BNE    .-6           ;;LOOP BACK IF NO
1585 003146 012706 001000          MOV    #1000,SP     ;;SETUP THE STACK POINTER
1586          ;;INITIALIZE A FEW VECTORS
1587 003152 012737 015272 000020    MOV    # $$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1588 003160 012737 000340 000022    MOV    #340,@#IOTVEC+2 ;;LEVEL 7
1589 003166 012737 014576 000030    MOV    # $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1590 003174 012737 000340 000032    MOV    #340,@#EMTVEC+2 ;;LEVEL 7
1591 003202 012737 017246 000034    MOV    # $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1592 003210 012737 000340 000036    MOV    #340,@#TRAPVEC+2;LEVEL 7
1593 003216 012737 015114 000024    MOV    # $PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1594 003224 012737 000340 000026    MOV    #340,@#PWRVEC+2 ;;LEVEL 7
1595 003232 013737 014430 014422    MOV    $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
1596 003240 005037 001060          CLR    $ESCAPE      ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1597 003244 112737 000001 001015    MOV    #1,$ERMAX    ;;ALLOW ONE ERROR PER TEST
1598 003252 012737 003252 001006    MOV    #.,$SLPADR   ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1599 003260 012737 003260 001010    MOV    #.,$LPERR    ;;SETUP THE ERROR LOOP ADDRESS
1600          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1601          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1602 003266 013746 000004          MOV    @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
1603 003272 012737 003326 000004    MOV    #64$,@#ERRVEC ;;SET UP ERROR VECTOR
1604 003300 012737 177570 001040    MOV    #DSWR,SWR    ;;SETUP FOR A HARDWARE SWICH REGISTER
1605 003306 012737 177570 001042    MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1606 003314 022777 177777 175516    CMP    #-1,@SWR    ;;TRY TO REFERENCE HARDWARE SWR
1607 003322 001012                   BNE    66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1608          ;;AND THE HARDWARE SWR IS NOT = -1
1609 003324 000403                   BR     65$          ;;BRANCH IF NO TIMEOUT
1610 003326 012716 003334          64$: MOV    #65$,(SP)   ;;SET UP FOR TRAP RETURN
1611 003332 000002                   RTI
1612 003334 012737 000176 001040 65$: MOV    #SWREG,SWR   ;;POINT TO SOFTWARE SWR
1613 003342 012737 000174 001042    MOV    #DISPREG,DISPLAY
1614 003350 012637 000004          66$: MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
1615
1616 003354 005037 001074          CLR    $PASS        ;;CLEAR PASS COUNT
1617 003360 132737 000200 001107    BITB  #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
1618 003366 001403                   BEQ    67$          ;;YES,USE NON-APT SWITCH
1619 003370 012737 001110 001040    MOV    # $$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
1620 003376
1621
1622          ;SIZE FOR 11/44 CPU
1623
1624 003376 013746 000010          MOV    @#10,-(SP)   ;** SAVE VECTOR
1625 003402 012737 003430 000010    MOV    #10$,@#10   ;** SET UP FOR TRAP
1626 003410 000007                   MFPT              ;** WHAT CPU??
1627 003412 122700 000001          CMPB  #1,R0         ;** ARE WE A 11/44
1628 003416 001007                   BNE    11$         ;** NO GO RESET VECTOR

```

INITIALIZE THE COMMON TAGS

```

1629 003420 052737 000001 003002      BIS      #1,@#FLAG44      ;** YES SET FLAG
1630 003426 000403                    BR      11$            ;** SKIP RTI
1631 003430 012716 003436      10$:    MOV      #11$, (SP) ;** SET STACK RETURN
1632 003434 000002                    RTI                    ;** RETURN
1633 003436 012637 000010      11$:    MOV      (SP)+,@#10 ;** RESTORE VECTOR
1634 003442 032777 000020 175370    BIT      #BIT4,@SWR    ;TEST CLOCK ONLY?
1635 003450 001404                    BEQ     INIT          ;BR IF NOT
1636 003452 005237 002624                    INC     CTSTFL        ;ELSE, SET CONSOLE TEST FLAG TO ENABLE CLOCK TESTS
1637 003456 000137 004620                    JMP     ID             ; AND JUMP TO TYPE PROGRAM ID
1638 003462 132737 000001 001106  INIT:   BITB     #BIT0,$ENV    ;CHECK IF ON APT
1639 003470 001404                    BEQ     MANL          ;BR IF NOT APT
1640 003472 132737 000200 001107    BITB     #BIT7,$ENVM   ;DID APT SIZE
1641 003500 001056                    BNE     APTSZD        ;BR, IF APT SIZED
1642 003502 032777 000040 175330  MANL:   BIT      #BIT5,@SWR ;WAS '$DEVN' MANUALLY SET?
1643 003510 001052                    BNE     APTSZD        ;IF YES, SKIP SELF-SIZING
1644
1645 003512 004737 003014      SIZE:   JSR     PC,DEVADR ;GENERATE DEVICE ADDRESS TABLE
1646 003516 005037 002630                    CLR     TMP2          ;CLR TEMP LOCATION TO KEEP DEVICE COUNT
1647 003522 005037 001144                    CLR     $DEVN         ;CLEAR DEVICE MAP
1648 003526 013703 000004                    MOV     @#4,R3        ;SAVE TIMEOUT VECTOR
1649 003532 012737 003562 000004    MOV     #4$,@#4       ;SET TIMEOUT POINTER
1650 003540 013700 001142                    MOV     $BASE,R0      ;LOAD BASE ADDRESS
1651 003544 062700 000160                    ADD     #160,R0       ;POINT R0 TO UNIT #15 (UNIT#0=CONSOLE)
1652 003550 005710      3$:    TST      (R0)        ;CHECK FOR DEVICE EXISTANCE
1653 003552 005237 001144                    INC     $DEVN         ;INDICATE DEVICE EXISTANCE IN DEVICE MAP
1654 003556 005237 002630                    INC     TMP2          ;INCREMENT DEVICE COUNT
1655 003562 012706 001000      4$:    MOV     #1000,SP     ;RESET STACK POINTER
1656 003566 006337 001144                    ASL     $DEVN         ;ADJUST DEVICE MAP FOR NEXT UNIT CHECK
1657 003572 162700 000010                    SUB     #10,R0        ;POINT R0 TO NEXT DEVICE NUMBER
1658 003576 023700 001142                    CMP     $BASE,R0     ;FINISHED SIZING?
1659 003602 003762                    BLE     3$           ;BR, IF BASE ADDRESS HAS NOT BEEN CHECKED
1660 003604 013700 002654                    MOV     CRCSR,R0     ;LOAD CONSOLE DEVICE ADDRESS
1661 003610 012737 003630 000004    MOV     #5$,@#4       ;SET UP TIMEOUT POINTER
1662 003616 005710                    TST     (R0)         ;TEST FOR CONSOLE EXISTANCE
1663 003620 005237 001144                    INC     $DEVN         ;INDICATE CONSOLE EXISTANCE IN DEVICE MAP
1664 003624 005237 002630                    INC     TMP2          ;INCREMENT DEVICE COUNT
1665 003630 010337 000004      5$:    MOV     R3,@#4       ;RESTORE TIMEOUT VECTOR
1666
1667 003634 000415                    BR      VCTADR        ;BR TO GENERATE VECTOR ADDRESS TABLE
1668
1669 003636 005037 002630      APTSZD: CLR     TMP2          ;CLEAR TEMP LOCATION TO KEEP DEVICE CNT
1670 003642 013702 001144                    MOV     $DEVN,R2     ;MOVE DEVICE MAP TO R2
1671 003646 005702      TSTDVM: TST     R2        ;TEST MSB OF DEVICE MAP
1672 003650 100002                    BPL     1$           ;BR, IF MSB IS ZERO
1673 003652 005237 002630                    INC     TMP2          ;INCREMENT DEVICE COUNT, IF MSB=1
1674 003656 006302      1$:    ASL     R2           ;SHIFT NEXT BIT INTO MSB POSITION
1675 003660 001401                    BEQ     DVADT        ;BR, IF NO OTHER BITS ARE SET IN $DEVN
1676 003662 000771                    BR      TSTDVM       ;CONTINUE CHECKING $DEVN, IF MORE BITS SET
1677 003664 004737 003014  DVADT: JSR     PC,DEVADR ;GENERATE DEVICE ADDRESS TABLE
1678
1679      ;GENERATE VECTOR ADDRESS TABLE
1680
1681 003670 012702 002742      VCTADR: MOV     #VCTTBL,R2 ;GET LOCATION OF VECTOR TABLE
1682 003674 013700 001136                    MOV     @#$VECT1,R0  ;COPY BASE VECTOR      ;:++EO
1683 003700 042700 177000                    BIC     #177000,R0   ;CLEAR BYTE SIGN EXTENSION ;:++EO
1684 003704 010001                    MOV     R0,R1

```

INITIALIZE THE COMMON TAGS

```

1685 003706 062701 000170          ADD    #170,R1      ;POINT R1 TO LAST DEVICE VECTOR
1686 003712 010022          1$:  MOV    RO,(R2)+ ;PUT VECTOR ADDRESS IN TABLE
1687 003714 062700 000010          ADD    #10,RO      ;POINT RO TO NEXT VECTOR ADDRESS
1688 003720 020001          CMP    RO,R1       ;FINISHED GENERATING VECTOR TABLE?
1689 003722 003773          BLE    1$          ;BR, IF LAST VECTOR IS NOT LOADED
1690
1691          ;MOVE DEVICE COUNT INTO DEVICE COUNT MESSAGE
1692
1693 003724 013700 002630          MOV    TMP2,RO     ;COPY DEVICE COUNT INTO RO
1694 003730 005001          CLR    R1          ;CLEAR AUXILIARY REGISTER
1695 003732 000300          SWAB   RO          ;PUT DEVICE COUNT IN UPPER BYTE OF RO
1696 003734 006300          ASL    RO          ;MOVE MSB OF COUNT INTO
1697 003736 006300          ASL    RO          ;MSB OF RO
1698 003740 006300          SHIFT: ASL   RO      ;PUT MSB OF COUNT INTO CARRY
1699 003742 106101          ROLB   R1          ;MOVE MSB OF COUNT INTO R1
1700 003744 006300          ASL    RO          ;MOVE NEXT BIT TO CARRY
1701 003746 106101          ROLB   R1          ;MOVE INTO R1
1702 003750 006300          ASL    RO          ;MOVE LAST BIT OF DIGIT
1703 003752 106101          ROLB   R1          ;INTO R1
1704 003754 062701 000060          ADD    #60,R1     ;CONVERT DIGIT TO ASCII
1705 003760 000301          SWAB   R1          ;MOVE DIGIT TO UPPER BYTE
1706 003762 032701 000020          BIT    #BIT4,R1   ;HAVE BOTH DIGITS BEEN MOVED TO R1?
1707 003766 001764          BEQ    SHIFT      ;BR, IF NOT
1708 003770 010137 024542          MOV    R1,M2A     ;MOVE DEVICE COUNT TO OUTPUT MESSAGE
1709
1710
1711 003774 052737 000002 002632 BEGIN: BIS    #BIT1,TMP3   ;SET UP BIT MASK TO TEST $DEVN FOR DEVICES EXCEPT CONSUL
1712 004002 005037 002626          CLR    TMP1       ;CLEAR LOCATION TO STORE TABLE OFFSETS
1713 004006 032737 000001 001144          BIT    #BIT0,$DEVN ;IS CONSOLE TO BE TESTED?
1714 004014 001001          BNE    TCONS      ;BR, IF CONSOLE IS TO BE TESTED
1715 004016 000414          BR     TSTDEV     ;BR, TO TEST OTHER DEVICES
1716 004020 005237 002624          TCONS: INC   CTSTFL ;INDICATE CONSOLE UNDER TEST
1717 004024 012700 002654          MOV    #CRCSR,RO  ;SET UP CONSOLE DEVICE ADDRESSES
1718 004030 012701 002634          MOV    #RCSR,R1   ;POINT R1 TO UUT ADDRESS TABLE
1719 004034 012021          1$:  MOV    (RO)+,(R1)+ ;TRANSFER CONSOLE ADDRESSES
1720 004036 022701 002652          CMP    #TPSW,R1   ;FINISHED TRANSFER?
1721 004042 002374          BGE    1$         ;BR, IF NOT
1722 004044 000137 004172          JMP    TST1       ;GO TEST CONSOLE INTERFACE
1723
1724          ;PREPARE ADDRESSES AND VECTORS FOR UUT
1725 004050 033737 002632 001144          TSTDEV: BIT   TMP3,$DEVN ;CHECK TO SEE IF DEVICE IS TO BE TESTED
1726 004056 001010          BNE    SETADR     ;BR, IF YES
1727 004060 006337 002632          ASL    TMP3       ;SHIFT MASK TO CHECK NEXT $DEVN BIT
1728 004064 062737 000002 002626          ADD    #2,TMP1    ;INCREMENT TABLE INDEX
1729 004072 005237 001100          INC    $UNIT      ;INCREMENT UNIT NUMBER
1730 004076 000764          BR     TSTDEV     ;GO TEST NEXT BIT OF DEVICE MAP
1731
1732 004100 005237 001100          SETADR: INC   $UNIT  ;UPDATE UNIT NUMBER
1733 004104 006337 002632          ASL    TMP3       ;UPDATE DEVICE MAP TEST MASK
1734 004110 013702 002626          MOV    TMP1,R2    ;MOVE TABLE OFFSET TO R2
1735 004114 062737 000002 002626          ADD    #2,TMP1    ;UPDATE TABLE OFFSET FOR NEXT DEVICE
1736 004122 016200 002702          MOV    ADRTBL(R2),RO ;PUT UUT ADDRESS INTO RO
1737 004126 012701 002634          MOV    #RCSR,R1   ;POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
1738 004132 010021          ADR:  MOV    RO,(R1)+ ;TRANSFER UUT ADDRESS
1739 004134 062700 000002          ADD    #2,RO      ;POINT TO NEXT UUT REGISTER
1740 004140 030027 000006          BIT    RO,#6      ;FINISHED TRANSFER?

```

```

1741 004144 001372          BNE    ADR          ;BR, IF NOT
1742
1743 004146 016200 002742          MOV    VCTTBL(R2),R0 ;PUT UUT VECTOR INTO R0
1744 004152 010021          VECT:  MOV    R0,(R1)+ ;TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
1745 004154 062700 000002          ADD    #2,R0        ;POINT TO NEXT VECTOR
1746 004160 030027 000006          BIT    R0,#6        ;FINISHED TRANSFER?
1747 004164 001372          BNE    VECT         ;BR, IF NOT
1748 004166 000137 004172          JMP    TST1         ;GO TEST DEVICE
1749
1750

```

```
1751
1752
1753
1754      ;*****
1755      ;*TEST 1      TEST ABILITY TO REFERENCE TCSR
1756      ;*****
1756 004172 000004 TST1: SCOPE
1757 004174 013703 000004      MOV      @#4,R3      ;SAVE TIMEOUT VECTOR
1758 004200 012737 004214 000004      MOV      #1$,@#4      ;SET UP TIMEOUT VECTOR
1759 004206 005777 176426      TST      @TCSR      ;REFERENCE THE XMIT COMMAND/STATUS REG.
1760 004212 000412      BR      4$      ;GO TO END OF TEST
1761
1762 004214 022626      1$:      CMP      (SP)+,(SP)+      ;RESTORE SP AFTER TIMEOUT
1763 004216 005737 002624      TST      CTSTFL      ;CHECK IF DEVICE IS CONSOLE
1764 004222 001002      BNE      2$      ;IF YES, SKIP ERROR TYPEOUT
1765 004224 104001      ERROR+1      ;REPORT ERROR TO APT & TTY
1766 004226 000404      BR      4$      ;BR TO END OF TEST
1767 004230
1768 004230 004737 015462      2$:      JSR      PC,$ATY4      ;;ONLY REPORT A FATAL ERROR
1769 004234 000001      1      ;;THE ERROR NUMBER (FROM APT LIST)
1770 004236 000000      3$:      HALT
1771 004240 010337 000004      4$:      MOV      R3,@#4      ;RESTORE TIMEOUT VECTOR
1772
1773
1774
1775      ;*****
1776      ;*TEST 2      TEST ABILITY TO REFERENCE TBUF
1777      ;*****
1778 004244 000004 TST2: SCOPE
1779 004246 013703 000004      MOV      @#4,R3      ;SAVE TIMEOUT VECTOR
1780 004252 012737 004266 000004      MOV      #1$,@#4      ;SET UP TIMEOUT VECTOR
1781 004260 005777 176356      TST      @TBUF      ;REFERENCE THE XMIT BUFFER
1782 004264 000412      BR      4$      ;GO TO END OF TEST
1783
1784 004266 022626      1$:      CMP      (SP)+,(SP)+      ;RESTORE SP AFTER TIMEOUT
1785 004270 005737 002624      TST      CTSTFL      ;CHECK IF DEVICE IS CONSOLE
1786 004274 001002      BNE      2$      ;IF YES, SKIP ERROR TYPEOUT
1787 004276 104002      ERROR+2      ;REPORT ERROR TO APT & TTY
1788 004300 000404      BR      4$      ;BR TO END OF TEST
1789 004302
1790 004302 004737 015462      2$:      JSR      PC,$ATY4      ;;ONLY REPORT A FATAL ERROR
1791 004306 000002      2      ;;THE ERROR NUMBER (FROM APT LIST)
1792 004310 000000      3$:      HALT
1793 004312 010337 000004      4$:      MOV      R3,@#4      ;RESTORE TIMEOUT VECTOR
```



```
1794
1795
1796
1797
1798
1799 004316 000004
1800 004320 000005
1801 004322 032777 000004 176310
1802 004330 001411
1803
1804 004332 005737 002624
1805 004336 001002
1806 004340 104015
1807 004342 000404
1808
1809 004344
1810 004344 004737 015462
1811 004350 000015
1812 004352 000000
1813
1814 004354 052777 000004 176256
1815 004362 032777 000004 176250
1816 004370 001001
1817
1818 004372 104016
1819
1820 004374 042777 000004 176236
1821 004402 032777 000004 176230
1822 004410 001411
1823
1824 004412 005737 002624
1825 004416 001002
1826 004420 104017
1827 004422 000404
1828 004424
1829 004424 004737 015462
1830 004430 000017
1831 004432 000000
1832
1833 004434 052777 000004 176176
1834 004442 000005
1835 004444 032777 000004 176166
1836 004452 001404
1837
1838 004454 042777 000004 176156
1839 004462 104020
1840
1841
1842 004464 000240
1843
1844
1845
1846 004466 000004
1847 004470 052777 000004 176142
1848 004476 005077 176140
1849 004502 105777 176132
```

*TEST 3 TEST THAT BIT2(MAINT. BIT) CAN BE SET & RESET

TST3: 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 10\$;** IF CLEAR, GO TO NEXT TEST

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

10\$: NOP ;**

*TEST 4 TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED

TST4: SCOPE
BIS #BIT2,@TCSR ;** ENABLE MAINT. WRAP
CLR @TBUF ;LOAD XBUF
TSTB @TCSR ;CHECK DONE

```
1850 004506 100021          BPL      3$          ;BR IF CLEAR
1851                          ;FILL SECOND BUFFER, BECAUSE REFRESH COULD CAUSE
1852                          ; FIRST TEST TO FAIL
1853 004510 005077 176126    CLR      @TBUF      ;FILL DOUBLE BUFFER
1854 004514 105777 176120    TSTB    @TCSR      ;CHECK DONE
1855 004520 100014          BPL      3$          ;BR IF CLEAR
1856
1857 004522 005737 002624    TST     CTSTFL     ;CHECK IF DEVICE IS CONSOLE
1858 004526 001005          BNE     1$          ;IF YES, SKIP ERROR TYPEOUT
1859 004530 042777 000004 176122 BIC     #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
1860                          ;** CONSOLE TO ALLOW FOR COMMUNICATION
1861                          ;** WITH TERMINAL.
1862 004536 104003          ERROR+3          ;DONE NOT CLEARED WITH TBUF FULL
1863 004540 000404          BR       3$          ;BR TO END OF TEST
1864 004542          1$:
1865 004542 004737 015462    JSR     PC,$ATY4    ;: ONLY REPORT A FATAL ERROR
1866 004546 000003          3          ;: THE ERROR NUMBER (FROM APT LIST)
1867 004550 000000          2$: HALT          ;TCSR 'DONE' NOT CLEARED WITH TBUF FULL
1868 004552 005000          3$: CLR      R0          ;CLEAR TIMER
1869 004554 105777 176060    4$: TSTB    @TCSR      ;CHECK FOR XMIT DONE
1870 004560 100417          BMI     ID          ;IF DONE SETS, BR TO END OF TEST
1871 004562 005200          INC     R0          ;INCREMENT TIMER
1872 004564 001373          BNE     4$          ;BR IF TIMER NOT DONE
1873
1874 004566 005737 002624    TST     CTSTFL     ;CHECK IF DEVICE IS CONSOLE
1875 004572 001005          BNE     5$          ;** DISABLE MAINTENANCE MODE FOR
1876 004574 042777 000004 176056 BIC     #BIT2,@TCSR ;** CONSOLE TO ALLOW FOR COMMUNICATION
1877                          ;** WITH TERMINAL.
1878
1879 004602 104004          ERROR+4          ;TCSR 'DONE' DOES NOT SET
1880 004604 000405          BR       ID          ;BR TO END OF TEST
1881 004606          5$:
1882 004606 004737 015462    JSR     PC,$ATY4    ;: ONLY REPORT A FATAL ERROR
1883 004612 000004          4          ;: THE ERROR NUMBER (FROM APT LIST)
1884 004614 000000          HALT
1885 004616 000427          BR       ENDB7     ;** BR TO NEXT TEST,
1886                          ;** AND SKIP THE TYPEOUT THAT FOLLOWS
1887                          ;** BECAUSE OF THIS FAILURE
1888
1889 004620          ID:
1890 004620 042777 000004 176032 BIC     #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
1891                          ;** CONSOLE TO ALLOW FOR COMMUNICATION
1892                          ;** WITH TERMINAL.
1893 004626 023737 000042 000046 CMP     @#42,@#46   ;UNDER ACT11?
1894 004634 001412          BEQ     6$          ;IF YES, SKIP IDENT. TYPEOUT
1895 004636 005737 001074    TST     $PASS      ;IS THIS THE FIRST PASS?
1896 004642 001007          BNE     6$          ;IF NOT BR TO NEXT TEST & SKIP THE IDENTIFICATION TYPEOU
1897 004644 005737 001076    TST     $DEVCT     ;IS THIS THE FIRST SUBPASS?
1898 004650 001004          BNE     6$          ;IF NOT, BR TO NEXT TEST
1899 004652 104401          TYPE
1900 004654 024502          M1
1901 004656 104401          TYPE          ;TYPE NUMBER OF DEVICES UNDER TEST
1902 004660 024540          M2
1903 004662 032777 000020 174150 6$: BIT     #BIT4,@SWR  ;CLOCK TEST ONLY?
1904 004670 001402          BEQ     ENDB7     ;** BR IF NOT
1905 004672 000137 005536    JMP     TCLOCK     ;ELSE, JUMP TO TEST CLOCK
```

1906 004676
1907 004676 005000
1908 004700 012701 000002
1909 004704 005300
1910 004706 001376
1911 004710 005301
1912 004712 001374
1913
1914
1915
1916
1917
1918
1919 004714 000004
1920 004716 052777 000004 175714
1921 004724 005077 175712
1922 004730 105777 175704
1923 004734 100375
1924 004736 005077 175700
1925 004742 000240
1926 004744 000005
1927 004746 105777 175666
1928 004752 100401
1929
1930 004754 104005
1931 004756 000240
1932

```
ENDB7:
40$: CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
      MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
      DEC R0 ;** TRANSMITTED TO FINISH BEFORE MAINTENANCE
      BNE 40$ ;** WRAP FOR THE UART UNDER TEST IS DISABLED.
      DEC R1 ;** THIS WILL INHIBIT ANY COMMUNICATION
      BNE 40$ ;** TO HARDWARE MEDIA SUCH AS TUSB THAT MIGHT
           ;** BE ATTACHED TO JART UNDER TEST.
```

```
::*****
;*TEST 5 TEST THAT TCSR 'DONE' SETS WITH RESET
*****
```

```
TST5: SCOPE
      BIS #BIT2,@TCSR ;** ENABLE MAINT. WRAP
      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 10$ ;** BR TO NEXT TEST IF DONE SET
10$: ERROR+5 ;TCSR 'DONE' DOES NOT SET WITH RESET
      NOP ;**
```

1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964

004760 000004
004762 013703 000004
004766 012737 005002 000004
004774 005777 175634
005000 000402

005002 022626
005004 104006
005006 010337 000004

005012 000004
005014 013703 000004
005020 012737 005034 000004
005026 005777 175604
005032 000402

005034 022626
005036 104007
005040 010337 000004

: *TEST 6 TEST ABILITY TO ACCESS RCSR

TST6: 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

: *TEST 7 TEST ABILITY TO ACCESS RBUF

TST7: 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

```
1965
1966
1967
1968
1969 005044 000004
1970 005046 032777 000400 173764
1971 005054 001475
1972 005056 032737 000001 003002
1973 005064 001407
1974 005066 005737 002624
1975
1976 005072 001404
1977 005074 032777 000010 173736
1978
1979 005102 001462
1980 005104 000005
1981 005106 032777 000001 175524
1982 005114 001411
1983 005116 005737 002624
1984 005122 001002
1985 005124 104011
1986 005126 000404
1987 005130
1988 005130 004737 015462
1989 005134 000011
1990 005136 000000
1991
1992 005140 052777 000001 175472
1993 005146 032777 000001 175464
1994 005154 001001
1995
1996 005156 104012
1997 005160 042777 000001 175452
1998 005166 032777 000001 175444
1999 005174 001411
2000 005176 005737 002624
2001 005202 001002
2002 005204 104013
2003 005206 000404
2004 005210
2005 005210 004737 015462
2006 005214 000013
2007 005216 000000
2008
2009 005220 052777 000001 175412
2010 005226 000005
2011 005230 032777 000001 175402
2012 005236 001404
2013 005240 042777 000001 175372
2014 005246 104014
2015 005250 000240
2016
```

:TEST 10 TEST THAT BITO(BREAK BIT) CAN BE SET & CLEARED & RESET

TST10: SCOPE
BIT #BIT8,@SWR ;IS BREAK FUNCTION ENABLED?
BEQ 10\$;** BR TO NEXT TEST, IF NOT
BIT #BIT0,FLAG4 ;** IS THIS A 11/44
BEQ 9\$;** NO
TST CTSTFL ;** YES THIS IS 11/44. IS THIS THE CONSOLE
BEQ 9\$;** SLU
BIT #BIT03,@SWR ;** NO
;** THIS IS THE CONSOLE SLU.SHOULD THE BREAK
;** TEST BE PERFORMED
BEQ 10\$;** NO
9\$: RESET ;CLEAR EVERYTHING
BIT #BIT0,@TCSR ;CHECK BITO OF TCSR CLEAR
BEQ 3\$;BR IF CLEAR
TST CTSTFL
BNE 1\$;BITO WAS NOT CLEAR AFTER RESET
ERROR+11
BR 3\$
1\$: JSR PC,\$ATY4 ;:ONLY REPORT A FATAL ERROR
11 ;:THE ERROR NUMBER (FROM APT LIST)
2\$: HALT
3\$: BIS #BIT0,@TCSR ;SET BITO IN TCSR
BIT #BIT0,@TCSR ;TEST BITO OF TCSR
BNE 4\$;BR IF SET
4\$: ERROR+12 ;BITO OF TCSR WILL NOT SET
BIC #BIT0,@TCSR ;CLEAR BITO OF TCSR
BIT #BIT0,@TCSR ;TEST BITO OF TCSR
BEQ 7\$
TST CTSTFL
BNE 5\$
5\$: ERROR+13 ;BITO OF TCSR WILL NOT CLEAR
BR 7\$
6\$: JSR PC,\$ATY4 ;:ONLY REPORT A FATAL ERROR
13 ;:THE ERROR NUMBER (FROM APT LIST)
7\$: BIS #BIT0,@TCSR ;SET BITO IN TCSR
RESET ;CLEAR BITO WITH RESET
BIT #BIT0,@TCSR ;TEST BITO CLEAR
BEQ 10\$;** BR IF CLEAR
BIC #BIT0,@TCSR ;CLEAR BITO, TO PRINT ERROR
ERROR+14 ;RESET DID NOT CLEAR BITO OF TCSR
10\$: NOP ;**

```
2017
2018
2019
2020 .....
2021 *TEST 11 TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET
2022 .....
2023 TST11: SCOPE
2024 RESET ;CLEAR EVERYTHING
2025 MOV @TVECT,R3 ;SAVE XMIT VECTOR
2026 MOV #18,@TVECT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
2027 JSR PC,WRPSW ;SET PSW TO PRIORITY=7
2028 .WORD 340
2029 BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
2030 BEQ 2$ ;BR IF ZERO
2031 ERROR+21 ;BIT6 IN TCSR NOT CLEAR AFTER RESET
2032 BR 2$
2033
2034 1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
2035 ERROR+22 ;XMIT INTERRUPT OCCURRED PRIO=7
2036
2037
2038 2$: BIS #BIT6,@TCSR ;SET BIT6 OF TCSR
2039 BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
2040 BNE 3$ ;BR, IF SET
2041
2042 ERROR+23 ;CANNOT SET BIT6 OF TCSR
2043
2044
2045 3$: BIC #BIT6,@TCSR ;CLEAR BIT6 OF TCSR
2046 BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
2047 BEQ 4$ ;BR IF CLEAR
2048 ERROR+24 ;CANNOT CLEAR BIT6 OF TCSR
2049
2050
2051 4$: BIS #BIT6,@TCSR ;SET BIT6 OF TCSR
2052 RESET ;CLEAR BIT6 WITH RESET
2053 BIT #BIT6,@TCSR ;TEST BIT6 OF TCSR
2054 BEQ 5$ ;BR IF CLEAR
2055
2056 ERROR+25 ;CANNOT CLEAR BIT6 OF TCSR WITH RESET
2057
2058 5$: MOV R3,@TVECT ;RESTORE XMIT VECTOR
```

```
2059
2060
2061
2062
2063
2064 005404 000004
2065 005406 000005
2066 005410 017703 175230
2067 005414 012777 005444 175222
2068 005422 004737 014540
2069 005426 000340
2070 005430 032777 000100 175176
2071 005436 001404
2072 005440 104026
2073
2074 005442 000402
2075
2076 005444 022626 1$:
2077 005446 104027 ERROR+27
2078
2079
2080 005450 052777 000100 175156 2$:
2081 005456 032777 000100 175150 BIT #BIT6,@RCSR
2082 005464 001001 BNE 3$ ;SET BIT6 OF RCSR
2083 ;TEST BIT6 OF RCSR
2084 005466 104030 ERROR+30 ;BR, IF SET
2085 ;CANNOT SET BIT6 OF RCSR
2086
2087 005470 042777 000100 175136 3$:
2088 005476 032777 000100 175130 BIC #BIT6,@RCSR ;CLEAR BIT6 OF RCSR
2089 005504 001401 BIT #BIT6,@RCSR ;TEST BIT6 OF RCSR
2090 BEQ 4$ ;BR, IF CLEAR
2091 005506 104031 ERROR+31
2092 ;CANNOT CLEAR BIT6 OF RCSR
2093
2094 005510 052777 000100 175116 4$:
2095 005516 000005 RESET ;SET BIT6 OF RCSR
2096 005520 032777 000100 175106 BIT #BIT6,@RCSR ;CLEAR BIT6 OF RCSR WITH RESET
2097 005526 001401 BEQ 5$ ;TEST BIT6 OF RCSR
2098 ;BR, IF CLEAR
2099 005530 104032 ERROR+32
2100 ;CANNOT CLEAR BIT6 OF RCSR WITH RESET
2101 005532 010377 175106 5$:
2102 MOV R3,@RVECT ;RESTORE RECEIVE VECTOR
```

```
2103
2104 005536 012737 000012 001002 TCLOCK: MOV #12,STSTNM ;ADJUST TEST NUMBER TO (NEXT TEST - 1)
2105 :*****
2106 :*TEST 13 TEST ABILITY TO ACCESS LKS
2107 :*****
2108 005544 000004 TST13: SCOPE
2109 005546 005737 002624 TST CTSTFL ;IS CONSOLE UNDER TEST?
2110 005552 001420 BEQ TST14 ;IF NOT, SKIP THIS TEST
2111 005554 032777 000100 173256 BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
2112 005562 001014 BNE TST14 ;IF YES, SKIP THIS TEST
2113 005564 013703 000004 MOV @#4,R3 ;SAVE TIMEOUT VECTOR
2114 005570 012737 005604 000004 MOV #1$,@#4 ;SET UP TIMEOUT VECTOR
2115 005576 005777 175072 TST @LKS ;ACCESS LKS
2116 005602 000402 BR 2$ ;NO TIMEOUT - BR TO END OF TEST
2117
2118 005604 022626 1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER TIMEOUT
2119 005606 104010 ERROR+10 ;CAN NOT ACCESS LKS
2120
2121 005610 010337 000004 2$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
2122
2123 :*****
2124 :*TEST 14 TEST THAT BIT6 OF LKS CAN BE SET & RESET
2125 :*****
2126 005614 000004 TST14: SCOPE
2127 005616 005737 002624 TST CTSTFL ;IS CONSOLE UNDER TEST?
2128 005622 001460 BEQ TST15 ;IF NOT, SKIP THIS TEST
2129 005624 032777 000100 173206 BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?
2130 005632 001054 BNE TST15 ;IF YES, SKIP THIS TEST
2131 005634 000005 RESET
2132 005636 017703 175034 MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR
2133 005642 012777 005672 175026 MOV #1$,@RTCVT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
2134 005650 004737 014540 JSR PC,WRPSW ;SET PSW TO PRIORITY 7
2135 005654 000340 .WORD 340
2136 005656 032777 000100 175010 BIT #BIT6,@LKS ;TEST BIT6 OF LKS
2137 005664 001404 BEQ 2$
2138 005666 104033 ERROR+33
2139 ;BIT6 OF LKS NOT CLEAR AFTER RESET
2140 005670 000402 BR 2$
2141
2142 005672 022626 1$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
2143 005674 104034 ERROR+34
2144 ;LKS INTERRUPT WITH PRIORITY=7
2145
2146 005676 052777 000100 174770 2$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
2147 005704 032777 000100 174762 BIT #BIT6,@LKS ;TEST BIT6 OF LKS
2148 005712 001001 BNE 3$ ;BR IF SET
2149
2150 005714 104035 ERROR+35
2151 ;CANNOT SET BIT6 OF LKS
2152
2153 005716 042777 000100 174750 3$: BIC #BIT6,@LKS ;CLEAR BIT6 OF LKS
2154 005724 032777 000100 174742 BIT #BIT6,@LKS ;TEST BIT6 OF LK
2155 005732 001401 BEQ 4$
2156 005734 104036 ERROR+36
2157 ;CANNOT CLEAR BIT6 OF LKS
2158 005736 052777 000100 174730 4$: BIS #BIT6,@LKS ;SET BIT6 OF LKS
```


2159	005744	000005				RESET		:CLEAR BIT6 OF LKS WITH RESET
2160	005746	032777	000100	174720		BIT	#BIT6,@LKS	:TEST BIT6 OF LKS
2161	005754	001401				BEQ	SS	:BR IF CLEAR
2162								
2163	005756	104037				ERROR+57		
2164								
2165	005760	010377	174712		SS:	MOV	R3,@RTCVT	:CANNOT CLEAR BIT6 OF LKS WITH RESET :RESTORE LINE CLOCK VECTOR

```
2166
2167
2168
2169
2170
2171 005764 000004
2172 005766 013703 000004
2173 005772 013704 000006
2174 005776 012737 006130 000004
2175 006004 012737 000340 000006
2176 006012 000005
2177 006014 012700 000002
2178 006020 032777 000020 173012
2179 006026 001404
2180 006030 013737 002674 001020
2181 006036 000403
2182 006040 013737 002634 001020 1$:
2183 006046 013737 001020 001022 2$:
2184 006054 040037 001022
2185 006060 023737 001020 001022
2186 006066 001002
2187 006070 050037 001022
2188 006074 017737 172722 001024 3$:
2189 006102 052777 000100 172712
2190 006110 032777 000100 172702
2191 006116 001011
2192 006120 013777 001024 172674
2193 006126 000401
2194 006130 022626 4$:
2195 006132 006300 5$:
2196 006134 105700
2197 006136 100343
2198 006140 000401
2199
2200 006142 104040 6$:
2201
2202
2203
2204 006144 010337 000004 7$:
2205 006150 010437 000006
```

```
*****
: *TEST 15 TEST FOR DUAL ADDRESSING OF REGISTERS
*****
TST15: SCOPE
MOV @#4,R3 ;SAVE TIMEOUT VECTOR
MOV @#6,R4 ;SAVE TIMEOUT PSW
MOV #4$,@#4 ;SET UP TIMEOUT VECTOR
MOV #340,@#6 ;KEEP PRIO=7
RESET ;CLEAR EVERYTHING
MOV #BIT1,R0 ;SET UP BIT MASK
BIT #BIT4,@SWR ;CLOCK TEST ONLY?
BEQ 1$ ;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,$SGDADR ;CHECK TO SEE IF GOOD ADDRESS CONTAINS BIT6
BNE 6$ ;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

6$: ERROR+40 ;DUAL ADDRESSING ERROR
; $BDADR = DUAL ADDRESS
; $GDADR = GOOD ADDRESS

7$: MOV R3,@#4 ;RESTORE TIMEOUT VECTOR
MOV R4,@#6 ;RESTORE TIMEOUT PSW
```

```
2206
2207
2208
2209
2210
2211 006154 000004
2212 006156 005737 002624
2213 006162 001437
2214 006164 032777 000100 172646
2215 006172 001033
2216 006174 000005
2217 006176 105777 174472 1$:
2218 006202 100401
2219
2220 006204 104041
2221
2222 006206 042777 000200 174460 2$:
2223 006214 032777 000200 174452
2224 006222 001410
2225 006224 042777 000200 174442
2226 006232 032777 000200 174434
2227 006240 001401
2228
2229 006242 104042
2230
2231 006244 005000 3$:
2232 006246 105777 174422 CONT:
2233 006252 100403
2234 006254 005200
2235 006256 001373
2236
2237 006240 104043
2238
```

```
*****
: *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$: BIC #BIT7,@LKS ; CLEAR BIT7 OF LKS
BIT #BIT7,@LKS ; TEST BIT7 OF LKS
BEQ 3$
BIC #BIT7,@LKS ; TRY ONE MORE TIME BECAUSE THE CLOCK
BIT #BIT7,@LKS ; MAY HAVE SET IMMEDIATELY AFTER THE FIRST CLEAR
BEQ 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
```

2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292

006262 000004
006264 005737 002624
006270 001505
006272 032777 000100 172540
006300 001101
006302 004737 014540
006306 000340
006310 017703 174362
006314 017704 174360
006320 012777 006364 174350
006326 012777 000340 174344
006334 042777 000200 174332
006342 052777 000100 174324
006350 105777 174320
006354 100375
006356 000240
006360 000240
006362 000402
006364 022626
006366 104044
006370 005077 174300
006374 012777 006424 174274
006402 004737 014540
006406 000240
006410 105777 174260
006414 100375
006416 000240
006420 000240
006422 000402
006424 022626
006426 104045
006430 012777 006464 174240
006436 042777 000200 174230
006444 052777 000100 174222
006452 105777 174216
006456 100375
006460 000240
006462 104046
006464 022626
006466 042777 000100 174200
006474 010377 174176
006500 010477 174174

```
*****  
: *TEST 17 TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY  
: *****  
TST17: SCOPE  
TST C1STFL ;IS CONSOLE UNDER TEST?  
BEQ TST20 ;IF NOT, SKIP THIS TEST  
BIT #BIT6,@SWR ;ARE LINE CLOCK TESTS INHIBITED?  
BNE TST20 ;IF YES, SKIP THIS TEST  
JSR PC,WRPSW ;SET PSW TO PRIORITY 7  
 .WORD 340  
MOV @RTCVT,R3 ;SAVE LINE CLOCK VECTOR  
MOV @RTCP SW,R4 ;SAVE LINE CLOCK PSW VECTOR  
MOV #2,@RTCVT ;SET RTC INTERRUPT VECTOR TO ERROR REPORT  
MOV #340,@RTCP SW ;KEEP PRIORITY AT 7  
BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG  
BIS #BIT6,@LKS ;SET INTERRUPT ENABLE  
1$: TSTB @LKS ;WAIT FOR RTC DONE (INTERRUPT REQUEST)  
 BPL 1$  
 NOP ;GIVE TIME FOR ANY INTERRUPTS  
 NOP ;GIVE TIME FOR ANY INTERRUPTS  
 BR 3$ ;BR, IF NO INTERRUPT OCCURS  
3$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
 ERROR+44 ;RTC INTERRUPTS AT PRIORITY 7  
3$: CLR @LKS ;DISABLE RTC INTERRUPTS & CLEAR DONE  
 MOV #4,@RTCVT ;SET RTC INTERRUPT VECTOR FOR ERROR  
 JSR PC,WRPSW ;CHANGE PSW TO PRIORITY 5  
 .WORD 240  
20$: TSTB @LKS ;WAIT FOR DONE (INTERRUPT REQUEST)  
 BPL 20$  
 NOP ;GIVE TIME FOR ANY INTERRUPT  
 NOP ;GIVE TIME FOR ANY INTERRUPT  
 BR 5$ ;IF NO INTERRUPT - BR TO CONTINUE TEST  
4$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
 ERROR+45 ;RTC INTERRUPTS WITH INTERRUPTS DISABLED  
5$: MOV #7,@RTCVT ;POINT RTC VECTOR TO END OF TEST  
 BIC #BIT7,@LKS ;CLEAR CLOCK DONE FLAG  
 BIS #BIT6,@LKS ;ALLOW INTERRUPTS  
6$: TSTB @LKS ;WAIT FOR RTC DONE  
 BPL 6$  
 NOP ;GIVE TIME FOR INTERRUPT  
 ERROR+46 ;RTC INTERRUPT DID NOT OCCUR  
7$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT  
 BIC #BIT6,@LKS ;DISABLE INTERRUPTS  
 MOV R3,@RTCVT ;RESTORE LINE CLOCK VECTOR  
 MOV R4,@RTCP SW ;RESTORE LINE CLOCK PSW VECTOR
```

2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335

006504 000004
006506 005737 002624
006512 001462
006514 032777 000100 172316
006522 001056
006524 000005
006526 017703 174144
006532 017704 174142
006536 012777 006606 174132
006544 012777 000340 174126
006552 004737 014540
006556 000240
006560 042777 000200 174106
006566 052777 000100 174100
006574 105777 174074
006600 100375
006602 000240
006604 104047
006606 022626
006610 012777 006636 174060
006616 004737 014540
006622 000240
006624 000240
006626 000240
006630 000240
006632 000240
006634 000402
006636 022626
006640 104050
006642 042777 000100 174024
006650 010377 174022
006654 010477 174020

```
*****
*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      @RTCP SW,R4 ;SAVE LINE CLOCK PSW VECTOR
        MOV      #2,@RTCVT   ;SET UP RTC INTERRUPT VECTOR
        MOV      #340,@RTCP SW ;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$:     CMP      (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
        MOV      #3,@RTCVT  ;POINT RTC VECTOR TO ERROR REPORT
        JSR      PC,WRPSW    ;SET PSW TO PRIORITY 5
        .WORD    240
        NOP               ;GIVE SOME TIME FOR AN INTERRUPT
        NOP               ;GIVE SOME TIME FOR AN INTERRUPT
        NOP               ;GIVE SOME TIME FOR AN INTERRUPT
        NOP               ;GIVE SOME TIME FOR AN INTERRUPT
        BR      4$        ;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,@RTCP SW ;RESTORE LINE CLOCK PSW VECTOR
```

```
2336
2337
2338
2339
2340
2341 006660 000004
2342 006662 005737 002624
2343 006666 001445
2344 006670 032777 000100 172142
2345 006676 001041
2346 006700 004737 014540
2347 006704 000340
2348 006706 017703 173764
2349 006712 012777 006772 173756
2350 006720 042777 000200 173746
2351 006726 052777 000100 173740
2352 006734 105777 173734
2353 006740 100375
2354 006742 000005
2355 006744 004737 014540
2356 006750 000240
2357 006752 000240
2358 006754 000240
2359 006756 000240
2360 006760 000240
2361 006762 042777 000100 173704
2362 006770 000402
2363
2364 006772 022626
2365 006774 104051
2366
2367 006776 010377 173674
```

```
*****
: *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
NOP ;GIVE TIME FOR ANY INTERRUPT
NOP ;GIVE TIME FOR ANY INTERRUPT
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
```

```

2368
2369
2370
2371
2372
2373 007002 000004
2374 007004 005737 002624
2375 007010 001452
2376 007012 032777 000100 172020
2377 007020 001046
2378 007022 004737 014540
2379 007026 000340
2380 007030 017703 173642
2381 007034 012777 007120 173634
2382 007042 042777 000200 173624
2383 007050 052777 000100 173616
2384 007056 105777 173612 1$:
2385 007062 100375
2386 007064 042777 000200 173602
2387 007072 004737 014540
2388 007076 000240
2389 007100 000240
2390 007102 000240
2391 007104 000240
2392 007106 000240
2393 007110 042777 000100 173556
2394 007116 000402
2395
2396
2397 007120 022626 2$:
2398 007122 104052
2399
2400 007124 010377 173546 3$:
2401 007130 004737 014540
2402 007134 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)
BPL 1$
BIC #BIT7,@LKS ;CLEAR DONE & INTERRUPT
JSR PC,WRPSW ;ALLOW INTERRUPTS
.WORD 240
NOP ;GIVE TIME FOR ANY INTERRUPT
NOP ;GIVE TIME FOR ANY INTERRUPT
NOP ;GIVE TIME FOR ANY INTERRUPT
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

```

```

2403
2404
2405
2406
2407
2408 007136 000004
2409 007140 005737 002624
2410 007144 001467
2411 007146 032777 000100 171664
2412 007154 001063
2413 007156 042777 000100 173510
2414
2415 007164 005000
2416 007166 012701 177777
2417 007172 005002 1$: CLR R2 ;CLEAR CLOCK COUNTER
2418 007174 005077 173474 CLR @LKS ;CLEAR DONE
2419 007200 105777 173470 TSTB @LKS ;SYNC ON DONE
2420 007204 100375 BPL ?$
2421 007206 005077 173462 CLR @LKS ;CLEAR DONE
2422 007212 105777 173456 3$: TSTB @LKS ;IS CLOCK DONE?
2423 007216 1000C3 BPL 4$ ;BR IF NOT , TO INCREMENT TIMER
2424 007220 005202 INC R2 ;IF DONE, INCREMENT CLOCK COUNT
2425 007222 005077 173446 CLR @LKS ;CLEAR DONE
2426 007226 005200 4$: INC R0 ;INCREMENT TIMER
2427 007230 001370 BNE 3$ ;BR IF TIME REMAINS
2428 007232 005201 INC R1 ;INCREMENT LOOP PASS FLAG
2429 007234 001003 BNE CMPARE ;BR IF TWO PASSES HAVE BEEN MADE
2430 007236 010237 002336 MOV R2,FIRST ;IF NOT, STORE FIRST CLOCK COUNT
2431 007242 000753 BR 1$ ;DO LOOP AGAIN
2432 007244 013701 002336 CMPARE: MOV FIRST,R1 ;RECALL FIRST CLOCK COUNT
2433 007250 160201 SUB R2,R1 ;CALCULATE DIFFERENCE OF TWO COUNTS
2434 007252 100001 BPL TOLER ;IF POSITIVE,SKIP NEGATION OF DIFFERENCE
2435 007254 005401 NEG R1 ;MAKE DIFFERENCE A POSITIVE NUMBER
2436 007256 032737 000001 003002 TOLER: BIT #BIT0,FLAG44 ;** IS THIS A 11/44
2437 007264 001403 BEQ 6$ ;** NO
2438 007266 020127 000002 CMP R1,#2 ;** YES; COMPARE DIFFERENCE WITH DESIRED
2439 BR 7$ ;** TOLERANCE OF 2
2440 007272 000402
2441 007274 020127 000001 6$: CMP R1,#1 ;COMPARE DIFFERENCE WITH DESIRED TOLERANCE
2442 007300 003403 7$: BLE 5$ ;BR, IF LOWER/EQUAL TO TOLERANCE
2443
2444 007302 010237 002616 MOV R2,SECND ;STORE SECOND COUNT
2445 007306 104053 ERROR+53 ;CLOCK REPEATABILITY ERROR
2446
2447 007310 032777 000020 171522 5$: BIT #BIT4,@SWR ;CLOCK TESTS ONLY?
2448 007316 001402 BEQ TST24 ;BR IF NOT
2449 007320 000137 014400 JMP $EOP ;ELSE, JUMP TO END OF PASS ROUTINE
2450

```



```

2451
2452
2453
2454
2455
2456 007324 000004
2457 007326 042777 000100 173304
2458 007334 017703 173310
2459 007340 012777 007364 173302
2460 007346 105777 173266
2461 007352 100375
2462 007354 004737 014540
2463 007360 000140
2464 007362 000402
2465
2466 007364 022626
2467 007366 104054
2468
2469 007370 012777 007416 173252
2470 007376 052777 000100 173234
2471 007404 000240
2472 007406 000240
2473 007410 000240
2474 007412 000240
2475
2476 007414 104055
2477
2478 007416 042777 000100 173214
2479 007424 022626
2480 007426 010377 173216
2481

```

```

:*****
:*TEST 24 TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
:*****
TST24: SCOPE
BIC #BIT6,@TCSR ;CLEAR TRANSMIT INTERRUPT ENABLE
MOV @TVECT,R3 ;SAVE XMIT VECTOR
MOV #2$,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT
1$: TSTB @TCSR ;WAIT FOR DONE
BPL 1$
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
BR 3$

2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR+54

3$: MOV #4$,@TVECT ;XMIT INTERRUPTS WITH INTERRUPT ENABLE CLEAR
BIS #BIT6,@TCSR ;SET XMIT VECTOR TO END OF TEST
NOP ;ENABLE INTERRUPTS
NOP ;**
NOP ;**
NOP ;**

ERROR+55 ;XMIT DID NOT INTERRUPT

4$: BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
MOV R3,@TVECT ;RESTORE XMIT VECTOR

```

```

2482
2483
2484
2485
2486 007432 000004
2487 007434 042777 000100 173176
2488 007442 004737 014540
2489 007446 000340
2490 007450 017703 173174
2491 007454 012777 007510 173166
2492 007462 105777 173152
2493 007466 100375
2494 007470 052777 000100 173142
2495 007476 000240
2496 007500 000240
2497 007502 000240
2498 007504 000240
2499 007506 000402
2500
2501 007510 022626
2502 007512 104056
2503
2504 007514 042777 000100 173116
2505 007522 012777 007550 173120
2506 007530 004737 014540
2507 007534 000140
2508 007536 000240
2509 007540 000240
2510 007542 000240
2511 007544 000240
2512 007546 000402
2513
2514 007550 022626
2515 007552 104057
2516
2517 007554 010377 173070

```

```

*****
*TEST 25      TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED
*****
TST25:  SCOPE
        BIC      #BIT6,@TCSR      ;DISABLE INTERRUPTS
        JSR      PC,WRPSW        ;SET PSW TO PRIORITY 7
        .WORD   340
        MOV      @TVECT,R3       ;SAVE XMIT VECTOR
        MOV      #2$,@TVECT     ;POINT XMIT VECTOR TO ERROR REPORT
1$:     TSTB     @TCSR           ;WAIT FOR DONE
        BPL      1$
        BIS      #BIT6,@TCSR     ;ENABLE INTERRUPT
        NOP
        NOP
        NOP
        NOP
        BR      3$              ;CONTINUE TEST
2$:     CMP      (SP)+,(SP)+     ;RESTORE SP AFTER INTERRUPT
        ERROR+56
3$:     BIC      #BIT6,@TCSR     ;XMIT INTERRUPTS AT PRIORITY=7
        MOV      #4$,@TVECT     ;CLEAR INTERRUPT ENABLE
        JSR      PC,WRPSW        ;POINT XMIT VECTOR TO ERROR REPORT
        .WORD   140            ;SET PSW TO PRIORITY 3
        NOP
        NOP
        NOP
        NOP
        BR      5$              ;BR TO END OF TEST-NO INTERRUPT
4$:     CMP      (SP)+,(SP)+     ;RESTORE SP AFTER INTERRUPT
        ERROR+57
5$:     MOV      R3,@TVECT      ;XMIT INTERRUPT OCCURES WITH BIT6 CLEAR
        ;RESTORE XMIT VECTOR

```

```

2518
2519
2520
2521
2522
2523 007560 000004
2524 007562 042777 000100 173050
2525 007570 017703 173054
2526 007574 017704 173052
2527 007600 012777 007650 173042
2528 007606 012777 000340 173036
2529 007614 004737 014540
2530 007620 000140
2531 007622 105777 173012
2532 007626 100375
2533 007630 052777 000100 173002
2534 007636 000240
2535 007640 000240
2536 007642 000240
2537 007644 000240
2538
2539 007646 104060
2540
2541 007650 022626
2542 007652 012777 007706 172770
2543 007660 004737 014540
2544 007664 000140
2545 007666 000240
2546 007670 000240
2547 007672 000240
2548 007674 000240
2549 007676 042777 000100 172734
2550 007704 000402
2551
2552 007706 022626
2553 007710 104061
2554
2555 007712 010377 172732
2556 007716 010477 172730
2557
2558
2559
2560
2561 007722 000004
2562 007724 042777 000100 172706
2563 007732 004737 014540
2564 007736 000340
2565 007740 017703 172704
2566 007744 012777 010032 172676
2567 007752 052777 000004 172660
2568 007760 052777 000100 172652
2569 007766 005077 172650
2570 007772 105777 172642
2571 007776 100375
2572 010000 005077 172636
2573 010004 004737 014540

```

```

*****
*TEST 26 TEST TRANSMITTER FOR DOUBLE INTERRUPTS
*****
TST26: SCOPE
BIC #BIT6,@TCSR ;CLEAR INTERRUPT ENABLE
MOV @TVECT,R3 ;SAVE XMIT VECTOR
MOV @TPSW,R4 ;SAVE XMIT PSW VECTOR
MOV #2,@TVECT ;SET UP XMIT VECTOR
MOV #340,@TPSW ;SET PIO 7 AFTER INTERRUPT
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
        .WORD 140
1$: TSTB @TCSR ;WAIT FOR DONE
    BPL 1$
    BIS #BIT6,@TCSR ;ENABLE INTERRUPTS
    NOP ;**
    NOP ;**
    NOP ;**
    NOP ;**
        ERROR+60
        ;XMIT INTERRUPT DID NOT OCCUR
2$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
    MOV #4,@TVECT ;POINT XMIT VECTOR TO ERROR
    JSR PC,WRPSW ;SET PSW TO PRIORITY 3
        .WORD 140
        ;GIVE TIME FOR ANY INTERRUPTS
    NOP ;GIVE TIME FOR ANY INTERRUPTS
    NOP ;GIVE TIME FOR ANY INTERRUPTS
    NOP ;GIVE TIME FOR ANY INTERRUPTS
    NOP ;GIVE TIME FOR ANY INTERRUPTS
    BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
    BR 5$ ;BR TO END OF TEST
4$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
    ERROR+61
        ;XMIT RE-INTERRUPTED
5$: MOV R3,@TVECT ;RESTORE XMIT VECTOR
    MOV R4,@TPSW ;RESTORE XMIT PSW VECTOR
*****
*TEST 27 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF
*****
TST27: SCOPE
BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
JSR PC,WRPSW ;SET PSW TO PRIORITY 7
        .WORD 340
    MOV @TVECT,R3 ;SAVE XMIT VECTOR
    MOV #2,@TVECT ;POINT XMIT VECTOR TO ERROR
    BIS #BIT2,@TCSR ;** ENABLE MAINT. WRAP
    BIS #BIT6,@TCSR ;ENABLE INTERRUPTS
    CLR @TBUF ;LOAD TBUF
1$: TSTB @TCSR ;WAIT FOR DONE (INTERRUPT)
    BPL 1$
    CLR @TBUF ;FILL SECOND BUFFER TO RESET INT.
    JSR PC,WRPSW ;ALLOW INTERRUPTS

```



```

2597
2598
2599
2600
2601 010066 000004
2602 010070 032737 000001 003002
2603 010076 001073
2604 010100 000005
2605 010102 052777 000004 172530
2606 010110 005000
2607 010112 012701 000002
2608
2609
2610 010116 105777 172512 42$: TSTB @RCSR
2611 010122 100405 BMI 43$
2612 010124 005300 DEC R0
2613 010126 001373 BNE 42$
2614 010130 005301 DEC R1
2615 010132 001371 BNE 42$
2616 010134 000402 BR .+6
2617 010136 005777 172474 43$: TST @RBUF
2618 010142 005000 CLR R0
2619 010144 005077 172472 CLR @TBUF
2620 010150 032777 004000 172456 WACTV: BIT #BIT11,@RCSR
2621 010156 001006 BNE 2$
2622 010160 005200 INC R0
2623 010162 001372 BNE WACTV
2624 010164 042777 000004 172466 BIC #BIT2,@CTCSR
2625
2626
2627
2628 010172 104063 ERROR+63 ;RCVR ACTIVE DID NOT SET WHILE RECEIVING
2629
2630 010174 2$:
2631 010174 005000 CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
2632 010176 012701 000002 MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
2633 010202 005300 40$: DEC R0 ;** TRANSMITTED TO FINISH BEFORE MAINTENANCE
2634 010204 001376 BNE 40$ ;** WRAP FOR THE UART UNDER TEST IS DISABLED.
2635 010206 005301 DEC R1 ;** THIS WILL INHIBIT ANY COMMUNICATION
2636 010210 001374 BNE 40$ ;** TO HARDWARE MEDIA SUCH AS TUSB THAT MIGHT
2637 ;** BE ATTACHED TO UART UNDER TEST.
2638 010212 000005 RESET
2639 010214 032777 004000 172412 BIT #BIT11,@RCSR ;VERIFY "INIT" CLFARS RCV ACTIVE
2640 010222 001401 BEQ 3$
2641
2642 010224 104115 ERROR+115 ;INIT DID NOT CLEAR RCV ACTIVE
2643
2644 010226 005000 3$: CLR R0 ;CLEAR A TIMER
2645 010230 052777 000004 172402 BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
2646 010236 062700 000000 WT: ADD #0,R0 ;WAIT AT LEAST ONE BIT TIME
2647 010242 005200 INC R0
2648 010244 001374 BNE WT
2649 010246 033777 004000 172360 BIT BIT11,@RCSR ;VERIFY RCV ACTIVE STILL CLEAR
2650 010254 001404 BEQ RCVDON ;BR IF CLEAR
2651
2652 010256 042777 000004 172374 BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR

```



```

2694
2695
2696      :*****
2697      :*TEST 31      TEST THAT READING RBUF CLEARS RECEIVER DONE
2698      :*****
2698 010402 000004      TST31: SCOPE
2699 010404 000005      RESET
2700 010406 052777 000004 172224      BIS      #BIT2,@TCSR      ;CLEAR EVERYTHING
2701 010414 005000      CLR      R0      ;SET MAINTENANCE WRAP
2702 010416 012701 000002      MOV      #2,R1      ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
2703      ;** THAT MIGHT BE IN THE PROCESS OF BEING
2704      ;** RECEIVED TO FINISH AFTER MAINTENANCE
2705 010422 105777 172206      42$: TSTB   @RCSR      ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
2706 010426 100405      BMI     43$      ;** CHECK FOR RECEIVER DONE
2707 010430 005300      DEC     R0      ;**
2708 010432 001373      BNE     42$      ;**
2709 010434 005301      DEC     R1      ;**
2710 010436 001371      BNE     42$      ;**
2711 010440 000402      BR      .+6      ;**
2712 010442 005777 172170      43$: TST   @RBUF      ;** READ TO CLEAR DONE
2713 010446 005077 172170      CLR   @TBUF      ;LOAD TRANSMITTER
2714 010452 105777 172156      1$: TSTB  @RCSR      ;WAIT FOR RECEIVER DONE
2715 010456 100375      BPL   1$
2716 010460 017700 172152      MOV   @RBUF,R0      ;READ RECEIVE BUFFER
2717 010464 042777 000004 172166      BIC   #BIT2,@TCSR      ;** DISABLE MAINTENANCE MODE FOR
2718      ;** CONSOLE TO ALLOW FOR COMMUNICATION
2719      ;** WITH TERMINAL.
2720 010472 105777 172136      TSTB  @RCSR      ;CHECK FOR RECEIVE DONE CLEAR
2721 010476 001401      BEQ   10$      ;** BR, IF CLEAR TO NEXT TEST
2722 010500 104070      ERROR+70
2723      ;READING RBUF DID NOT CLEAR RCVR DONE
2724 010502 000240      10$: NOP
2725
2726      :*****
2727      :*TEST 32      TEST THAT RDR ENABLE CLEARS RECEIVER DONE FLAG
2728      :*****
2729 010504 000004      TST32: SCOPE
2730 010506 032737 000001 003002      BIT     #BIT0,FLAG44      ;** 11/44 ??
2731 010514 001050      BNE     10$      ;** YES DO NOT EXECUTE THIS TEST
2732 010516 000005      RESET      ;CLEAR EVERYTHING
2733 010520 052777 000001 172106      BIS     #BIT0,@RCSR      ;SET RDR ENABLE
2734 010526 052777 000004 172104      BIS     #BIT2,@TCSR      ;SET MAINTENANCE WRAP
2735 010534 005000      CLR     R0      ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
2736 010536 012701 000002      MOV     #2,R1      ;** THAT MIGHT BE IN THE PROCESS OF BEING
2737      ;** RECEIVED TO FINISH AFTER MAINTENANCE
2738      ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
2739 010542 105777 172066      42$: TSTB  @RCSR      ;** CHECK FOR RECEIVER DONE
2740 010546 100405      BMI     43$      ;**
2741 010550 005300      DEC     R0      ;**
2742 010552 001373      BNE     42$      ;**
2743 010554 005301      DEC     R1      ;**
2744 010556 001371      BNE     42$      ;**
2745 010560 000402      BR      .+6      ;**
2746 010562 005777 172050      43$: TST   @RBUF      ;** READ TO CLEAR DONE
2747 010566 005077 172050      CLR   @TBUF      ;LOAD TRANSMITTER
2748 010572 105777 172036      1$: TSTB  @RCSR      ;WAIT FOR RECEIVER DONE
2749 010576 100375      BPL   1$

```

```

2750 010600 032777 000001 172026      BIT    #BIT0,@RCSR      ;VERIFY RCV ACTIVE CLEARED RDR ENABLE
2751 010606 001401                      BEQ    2$              ;BR IF CLEAR
2752
2753 010610 104117                      ERROR+117             ;RDR ENABLE NOT CLEARED WITH RCV ACTIVE
2754
2755 010612 052777 000001 172014 2$:    BIS    #BIT0,@RCSR      ;CLEAR DONE BY SETTING RDR ENABLE
2756 010620 105777 172010                TSTB   @RCSR           ;CHECK FOR DONE CLEAR
2757 010624 001404                      BEQ    10$             ;** BR, IF CLEAR TO NEXT TEST
2758 010626 042777 000004 172024      BIC    #BIT2,@CTCSR    ;** DISABLE MAINTENANCE MODE FOR
2759                                     ;** CONSOLE TO ALLOW FOR COMMUNICATION
2760                                     ;** WITH TERMINAL.
2761 010634 104067                      ERROR+67
2762
2763 010636 000240                      10$:   NOP             ;SETTING RDR ENABLE DID NOT CLEAR RCVR DONE
2764                                     ;**
2765

```



```

2766
2767
2768      ;*****
2769      ;*TEST 33      TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
2770      ;*****
2770 010640 000C04      TST33: SCOPE
2771 010642 042777 000100 171770      BIC      #BIT6,@TCSR      ;DISABLE TRANSMIT INTERRUPTS
2772 010650 042777 000100 171756      BIC      #BIT6,@RCSR      ;DISABLE RECEIVER INTERRUPTS
2773 010656 052777 000004 171754      BIS      #BIT2,@TCSR      ;SET MAINTENANCE WRAP
2774 010664 005000      CLR      R0                ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
2775 010666 012701 000002      MOV      #2,R1            ;** THAT MIGHT BE IN THE PROCESS OF BEING
2776                                     ;** RECEIVED TO FINISH AFTER MAINTENANCE
2777                                     ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
2778 010672 105777 171736      42$:  TSTB      @RCSR      ;** CHECK FOR RECEIVER DONE
2779 010676 100405      BMI      43$
2780 010700 005300      DEC      R0                ;**
2781 010702 001373      BNE      42$
2782 010704 005301      DEC      R1                ;**
2783 010706 001371      BNE      42$
2784 010710 000402      BR       .+6
2785 010712 005777 171720      43$:  TST      @RBUF      ;** READ TO CLEAR DONE
2786 010716 017703 171722      MOV      @RVECT,R3        ;SAVE RECEIVE VECTOR
2787 010722 012777 010760 171714      MOV      #2$,@RVECT      ;POINT RCV VECTOR TO ERROR REPORT
2788 010730 004737 014540      JSR      PC,WRPSW        ;SET PSW TO PRIORITY 3
2789 010734 000140      .WORD   140
2790 010736 005077 171700      CLR      @TBUF            ;SEND A CHARACTER
2791 010742 105777 171666      1$:  TSTB      @RCSR      ;WAIT FOR RECEIVER DONE
2792 010746 100375      BPL      1$
2793 010750 042777 000004 171702      BIC      #BIT2,@CTCSR    ;** DISABLE MAINTENANCE MODE FOR
2794                                     ;** CONSOLE TO ALLOW FOR COMMUNICATION
2795                                     ;** WITH TERMINAL.
2796 010756 000405      BR       3$
2797 010760      2$:  BR       3$
2798 010760 042777 000004 171672      BIC      #BIT2,@CTCSR    ;** DISABLE MAINTENANCE MODE FOR
2799                                     ;** CONSOLE TO ALLOW FOR COMMUNICATION
2800                                     ;** WITH TERMINAL.
2801 010766 022626      CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
2802 010770 104071      ERROR+71
2803                                     ;RECEIVER INTERRUPTS WITH INT. ENABLE CLEAR
2804 010772 012777 011026 171644      3$:  MOV      #4$,@RVECT      ;POINT RCV VECTOR TO END OF TEST
2805 011000 052777 000100 171626      BIS      #BIT6,@RCSR      ;ENABLE RCV INTERRUPTS
2806 011006 000240      NOP
2807 011010 000240      NOP
2808 011012 000240      NOP
2809 011014 000240      NOP
2810 011016 042777 000004 171634      BIC      #BIT2,@CTCSR    ;** DISABLE MAINTENANCE MODE FOR
2811                                     ;** CONSOLE TO ALLOW FOR COMMUNICATION
2812                                     ;** WITH TERMINAL.
2813 011024 104072      ERROR+72
2814                                     ;RCVR DID NOT INTERRUPT
2815 011026 042777 000100 171600      4$:  BIC      #BIT6,@RCSR      ;DISABLE INTERRUPTS
2816 011034 042777 000004 171616      BIC      #BIT2,@CTCSR    ;** DISABLE MAINTENANCE MODE FOR
2817                                     ;** CONSOLE TO ALLOW FOR COMMUNICATION
2818                                     ;** WITH TERMINAL.
2819 011042 022626      CMP      (SP)+,(SP)+      ;RESTORE SP AFTER INTERRUPT
2820 011044 010377 171574      MOV      R3,@RVECT      ;RESTORE RECEIVE VECTOR

```

```

2821
2822
2823      ;*****
2824      ;*TEST 34      TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED
2825      ;*****
2825      TST34:  SCOPE
2826      RESET      ;CLEAR EVERYTHING
2827      JSR      PC,WRPSW      ;SET PSW TO PRIORITY 7
2828      .WORD      340
2829      MOV      @RVECT,R3      ;SAVE RECEIVE VECTOR
2830      MOV      #2,@RVECT      ;POINT RCVR VECTOR TO ERROR REPORT
2831      BIS      #BIT2,@TCSR      ;SET MAINTENANCE WRAP
2832      CLR      R0      ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
2833      MOV      #2,R1      ;** THAT MIGHT BE IN THE PROCESS OF BEING
2834      ;** RECEIVED TO FINISH AFTER MAINTENANCE
2835      ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
2836      42$:  TSTB      @RCSR      ;** CHECK FOR RECEIVER DONE
2837      BMI      43$
2838      DEC      R0
2839      BNE      42$
2840      DEC      R1
2841      BNE      42$
2842      BR      .+6
2843      43$:  TST      @RBUF      ;** READ TO CLEAR DONE
2844      6$:  TSTB      @S1PS      ;TEST FOR XMIT READY
2845      BPL      6$
2846      CLR      @TBUF      ;SEND A CHARACTER
2847      1$:  TSTB      @RCSR      ;WAIT FOR RECEIVER DONE
2848      BPL      1$
2849      BIS      #BIT6,@RCSR      ;ENABLE INTERRUPTS
2850      NOP
2851      NOP      ;** GIVE TIME FOR INTERRUPT
2852      NOP      ;** GIVE TIME FOR INTERRUPT
2853      NOP      ;** GIVE TIME FOR INTERRUPT
2854      NOP      ;** GIVE TIME FOR INTERRUPT
2855      BR      3$
2856
2857      2$:  BIC      #BIT2,@TCSR      ;** DISABLE MAINTENANCE MODE FOR
2858      ;** CONSOLE TO ALLOW FOR COMMUNICATION
2859      ;** WITH TERMINAL.
2860      LMP      (SP)+,(SP)+
2861      ERROR+73      ;RESTORE SP AFTER INTERRUPT
2862
2863      ;RCVR INTERRUPTS AT PRIORITY 7
2864
2865      3$:  BIC      #BIT6,@RCSR      ;CLEAR INTERRUPT ENABLE
2866      MOV      #4,@RVECT      ;POINT PCVR VECTOR TO ERROR REPORT
2867      JSR      PC,WRPSW      ;SET PSW TO PRIORITY 3
2868      .WORD      140
2869      NOP
2870      NOP      ;GIVE TIME FOR ANY INTERRUPT
2871      BIC      #BIT2,@TCSR      ;GIVE TIME FOR ANY INTERRUPT
2872      ;** DISABLE MAINTENANCE MODE FOR
2873      ;** CONSOLE TO ALLOW FOR COMMUNICATION
2874      ;** WITH TERMINAL.
2875      BR      5$
2876      ;BR TO END OF TEST, IF NO INTERRUPT

```



```
2885
2886
2887
2888
2889 011262 000004
2890 011264 000005
2891 011266 017703 171352
2892 011277 017704 171350
2893 011276 012777 011420 171340
2894 011304 012777 000340 171334
2895 011312 004737 014540
2896 011316 000140
2897 011320 052777 000004 171312
2898 011326 005000
2899 011330 012701 000002
2900
2901
2902 011334 105777 171274 42$: TSTB @RCSR
2903 011340 100405 BMI 43$
2904 011342 005300 DEC R0
2905 011344 001373 BNE 42$
2906 011346 005301 DEC R1
2907 011350 001371 BNE 42$
2908 011352 000402 BR .+6
2909 011354 005777 171256 43$: TST @RBUF
2910 011360 005077 171256 CLR @TBUF
2911 011364 105777 171244 1$: TSTB @RCSR
2912 011370 100375 BPL 1$
2913 011372 042777 000004 171260 BIC #BIT2,@TCSR
2914
2915
2916 011400 052777 000100 171226 BIS #BIT6,@RCSR
2917 011406 000240 NOP
2918 011410 000240 NOP
2919 011412 000240 NOP
2920 011414 000240 NOP
2921
2922 011416 104075 ERROR+75
2923
2924 011420 022626 2$: CMP (SP)+,(SP)+
2925 011422 012777 011466 171214 MOV #3$,@RVECT
2926 011430 004737 014540 JSR PC,WRPSW
2927 011434 000140 .WORD 140
2928 011436 000240 NOP
2929 011440 000240 NOP
2930 011442 000240 NOP
2931 011444 000240 NOP
2932 011446 042777 000100 171160 BIC #BIT6,@RCSR
2933 011454 010377 171164 MOV R3,@RVECT
2934 011460 010477 171162 MOV R4,@RPSW
2935 011464 000402 BR 4$
2936 011466 022626 3$: CMP (SP)+,(SP)+
2937 011470 104076 ERROR+76
2938
2939 011472 010377 171146 4$ MOV R3,@RVECT
```

```
*****
*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 R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
;** RECEIVED TO FINISH AFTER MAINTENANCE
;** WRAP FOR THE UART UNDER TEST IS ENABLED.
;** CHECK FOR RECEIVER DONE
42$: TSTB @RCSR
BMI 43$
DEC R0
BNE 42$
DEC R1
BNE 42$
BR .+6
43$: TST @RBUF
CLR @TBUF ;SEND A CHARACTER
1$: TSTB @RCSR ;WAIT FOR RCVR DONE
BPL 1$
BIC #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
BIS #BIT6,@RCSR ;ENABLE RCV INTERRUPTS
NOP ;** GIVE SOME TIME
NOP ;** GIVE SOME TIME
NOP ;** GIVE SOME TIME
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
; ** GIVE SOME TIME
; ** GIVE SOME TIME
; ** GIVE SOME TIME
; ** 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
```

2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
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

011476 000004
011500 000005
011502 004737 014540
011506 000340
011510 017703 171130
011514 012777 011642 171122
011522 052777 000100 171104
011530 052777 000004 171102
011536 005000
011540 012701 000002

011544 105777 171064
011550 100405
011552 005300
011554 001373
011556 005301
011560 001371
011562 000402
011564 005777 171046
011570 005077 171046
011574 105777 171034
011600 100375
011602 042777 000004 171050

011610 005077 171022
011614 004737 014540
011620 000140
011622 000240
011624 000240
011626 000240
011630 000240
011632 042777 000100 170774
011640 000402

011642 022626
011644 104077

011646 010377 170772

:TEST 36 TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF

TST36: SCOPE
RESET ;CLEAR EVERYTHING
JSR PC,WRPSW ;SET PSW PRIORITY TO 7
.WORD 340
MOV @RVECT,R3 ;SAVE RECEIVE VECTOR
MOV #2,@RVECT ;POINT RCV VECTOR TO ERROR REPORT
BIS #BIT6,@RCSR ;SET RCVR INTERRUPT ENABLE
BIS #B. 2,@RCSR ;SET MAINTENANCE WRAP
CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
;** RECEIVED TO FINISH AFTER MAINTENANCE
;** WRAP FOR THE UART UNDER TEST IS ENABLED.
;** CHECK FOR RECEIVER DONE
42\$: TSTB @RCSR
BMI 43\$
DEC R0
BNE 42\$
DEC R1
BNE 42\$
BR .+6
43\$: TST @RBUF ;** READ TO CLEAR DONE
CLR @TBUF ;SEND A CHARACTER
1\$: TSTB @RCSR ;WAIT FOR DONE (INTERRUPT)
BPL 1\$
BIC #BIT2,@RCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
CLR @RBUF ;READ RBUF TO CLEAR PENDING INTERRUPT
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
;** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
;** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
;** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
;** ALLOW TIME FOR ANY ERRONEOUS INTERRUPT
BIC #BIT6,@RCSR ;NO INTERRUPT-CLEAR INT. ENABLE
BR 3\$
2\$: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
ERROR+77 ;READING RBUF DID NOT CLEAR INTERRUPT
3\$: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR

```

2986
2987
2988
2989
2990
2991 011652 000004
2992 011654 000005
2993 011656 004737 014540
2994 011662 000340
2995 011664 017703 170754
2996 011670 012777 012010 170746
2997 011676 052777 000100 170730
2998 011704 052777 000004 170726
2999 011712 005000
3000 011714 012701 000002
3001
3002
3003 011720 105777 170710 42$: TSTB @RCSR
3004 011724 100405 BMI 43$
3005 011726 005300 DEC R0
3006 011730 001373 BNE 42$
3007 011732 005301 DEC R1
3008 011734 001371 BNE 42$
3009 011736 000402 BR .+6
3010 011740 005777 170672 43$: TST @RBUF
3011 011744 012777 000377 170670 MOV #377,@TBUF
3012 011752 105777 170656 1$: TSTB @RCSR
3013 011756 100375 BPL 1$
3014 011760 000005 RESET
3015 011762 004737 014540 JSR PC,WRPSW
3016 011766 000140 .WORD 140
3017 011770 000240 NOP
3018 011772 000240 NOP
3019 011774 000240 NOP
3020 011776 000240 NOP
3021 012000 042777 000100 170626 BIC #BIT6,@RCSR
3022 012006 000402 BR 3$
3023
3024
3025 012010 022626 2$: CMP (SP)+,(SP)+
3026 012012 104100 ERROR+100
3027
3028 012014 010377 170624 3$: MOV R3,@RVECT

```

```

:*****
: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
CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
;** RECEIVED TO FINISH AFTER MAINTENANCE
;** WRAP FOR THE UART UNDER TEST IS ENABLED.
;** CHECK FOR RECEIVER DONE
42$: TSTB @RCSR
BMI 43$
DEC R0
BNE 42$
DEC R1
BNE 42$
BR .+6
43$: TST @RBUF
MOV #377,@TBUF ;SEND AN ALL 1'S CHARACTER
1$: TSTB @RCSR ;WAIT FOR RCV DONE
BPL 1$
RESET ;CLEAR RCV INTERRUPT & RBUF
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
; ** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
; ** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
; ** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
; ** ALLOW TIME FOR AN ERRONEOUS INTERRUPT
;NO INTERRUPT-CLEAR INT. ENABLE
;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

```

```
3029
3030
3031
3032
3033
3034 012020 000004
3035 012022 032777 002000 167010
3036 012030 001471
3037 012032 032737 000001 003002
3038 012040 001407
3039 012042 005737 002624
3040
3041 012046 001404
3042 012050 032777 000010 166762
3043
3044 012056 001456
3045 012060 000005
3046 012062 052777 000004 170550
3047 012070 005000
3048 012072 012701 000002
3049
3050
3051 012076 105777 170532
3052 012102 100405
3053 012104 005300
3054 012106 001373
3055 012110 005301
3056 012112 001371
3057 012114 000402
3058 012116 005777 170514
3059 012122 012700 000003
3060 012126 005077 170510
3061 012132 105777 170502
3062 012136 100375
3063 012140 005300
3064 012142 001371
3065 012144 042777 000004 170506
3066
3067
3068 012152 032777 040000 170456
3069 012160 001001
3070 012162 104101
3071
3072
3073 012164 032777 100000 170444
3074 012172 001001
3075 012174 104102
3076
3077 012176
3078 012176 005000
3079 012200 012701 000002
3080 012204 005300
3081 012206 001376
3082 012210 005301
3083 012212 001374
3084
```

```
*****
:TEST 40 TEST THAT THE 'OR' ERROR & 'ERROR' CAN BE SET
*****
TST40: SCOPE
BIT #BIT10,@SWR ;IS THIS TEST ENABLED
BEQ TST41 ;IF NOT ENABLED, BR TO NEXT TEST
BIT #BIT0,FLAG44 ;** IS THIS A 11/44
BEQ 9$ ;** NO
TST CTSTFL ;** YES THIS IS 11/44. IS THIS THE CONSOLE
;** SLU
BEQ 9$ ;** NO
BIT #BIT03,@SWR ;** THIS IS THE CONSOLE SLU.SHOULD THE OVERRUN
;** ERROR TEST BE PERFORMED
BEQ TST41 ;** NO
9$: RESET ;CLEAR EVERYTHING
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
;** RECEIVED TO FINISH AFTER MAINTENANCE
;** WRAP FOR THE UART UNDER TEST IS ENABLED.
;** CHECK FOR RECEIVER DONE
42$: TSTB @RCSR
BMI 43$
DEC R0
BNE 42$
DEC R1
BNE 42$
BR .+6
43$: TST @RBUF
MOV #3,R0 ;** READ TO CLEAR DONE
CLR @TBUF ;SET CHARACTER COUNT TO SEND 3 CHAR.
2$: TSTB @TCSR ;LOAD TRANSMIT BUFFER
BPL 2$ ;WAIT FOR TRANSMIT DONE
DEC R0 ;DECREMENT CHARACTER COUNT
BNE 1$ ;BR IF ALL CHARACTERS NOT TRANSMITTED
BIC #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
BIT #BIT14,@RBUF ;TEST FOR 'OR' ERROR FLAG
BNE 3$ ;BR, IF SET
ERROR+101 ;'OR' ERROR FLAG DID NOT SET
3$: BIT #BIT15,@RBUF ;TEST 'ERROR' FLAG
BNE 4$ ;BR, IF SET
ERROR+102 ;'ERROR' FLAG DID NOT SET WITH 'OR' FLAG
4$: CLR R0
MOV #2,R1
40$: DEC R0
BNE 40$
DEC R1
BNE 40$
;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
;** THAT MIGHT BE IN THE PROCESS OF BEING
;** TRANSMITTED TO FINISH BEFORE MAINTENANCE
;** WRAP FOR THE UART UNDER TEST IS DISABLED.
;** THIS WILL INHIBIT ANY COMMUNICATION
;** TO HARDWARE MEDIA SUCH AS TUSB THAT MIGHT
;** BE ATTACHED TO UART UNDER TEST.
```

```

3085
3086
3087
3088
3089
3090 012214 000004
3091 012216 032777 000400 166614
3092 012224 001505
3093 012226 032737 000001 003002
3094 012234 001407
3095 012236 005737 002624
3096
3097 012242 001404
3098 012244 032777 000010 166566
3099
3100 012252 001472
3101 012254 000005
3102 012256 052777 000004 170354
3103 012264 005000
3104 012266 012701 000002
3105
3106
3107 012272 105777 170336
3108 012276 100405
3109 012300 005300
3110 012302 001373
3111 012304 005301
3112 012306 001371
3113 012310 000402
3114 012312 005777 170320
3115 012316 012777 177777 170316
3116 012324 105777 170304
3117 012330 100375
3118 012332 005077 170300
3119 012336 052777 000001 170274
3120 012344 005000
3121 012346 117737 170262 001026
3122 012354 100411
3123 012356 005200
3124 012360 001372
3125 012362 042777 000001 170250
3126 012370 042777 000004 170262
3127
3128
3129 012376 104103
3130 012400 105777 170232
3131 012404 001407
3132 012406 042777 000001 170224
3133 012414 042777 000004 170236
3134
3135
3136 012422 104103
3137 012424 042777 000001 170206
3138 012432 042777 000004 170220
3139
3140

```

```

*****
*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 NOT ENABLED
BIT #BIT0,FLAG44 ;** IS THIS A 11/44
BEQ 9$ ;** NO
TST CTSTFL ;** YES THIS IS 11/44. IS THIS THE CONSOLE
;** SLU
BEQ 9$ ;** NO
BIT #BIT03,@SWR ;** THIS IS THE CONSOLE SLU.SHOULD THE BREAK
;** TEST BE PERFORMED
BEQ TST42 ;** NO
9$: RESET ;CLEAR EVERYTHING
BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
;** RECEIVED TO FINISH AFTER MAINTENANCE
;** WRAP FOR THE UARI UNDER TEST IS ENABLED.
;** CHECK FOR RECEIVER DONE
42$: TSTB @RCR
BMI 43$
DEC R0
BNE 42$
DEC R1
BNE 42$
BR .+6
43$: TST @RBUF ;** READ TO CLEAR DONE
MOV #-1,@TBUF ;TRANSMIT ALL ONES TO RCVR
1$: TSTB @RCR ;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
2$: MOVB @RCR,$BDDAT ;WAIT FOR RCVR DONE
BMI CONT41 ;BR IF DONE
INC R0 ;IF NOT, INCREMENT TIMER
BNE 2$ ;BR IF TIME REMAINS
BIC #BIT0,@TCSR ;CLEAR BREAK BIT
BIC #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
;BREAK DID NOT TRANSMIT ANYTHING
CONT41: TSTB @RBUF ;CHECK RECEIVE BUFFER FOR ZERO
BEQ 3$ ;BR, IF ZERO
BIC #BIT0,@TCSR ;CLEAR BREAK BIT
BIC #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
;BREAK DID NOT TRANSMIT ALL ZEROES
3$: BIC #BIT0,@TCSR ;CLEAR BREAK BIT
BIC #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.

```



```
3141
3142
3143
3144
3145
3146 012440 000004
3147 012442 032777 002000 166370
3148 012450 001470
3149 012452 032777 000400 166360
3150 012460 001464
3151 012462 032737 000001 003002
3152 012470 001407
3153 012472 005737 002624
3154
3155 012476 001404
3156 012500 032777 000010 166332
3157
3158 012506 001451
3159 012510 000005
3160 012512 052777 000004 170120
3161 012520 005000
3162 012522 012701 000002
3163
3164
3165 012526 105777 170102
3166 012532 100405
3167 012534 005300
3168 012536 001373
3169 012540 005301
3170 012542 001371
3171 012544 000402
3172 012546 005777 170064
3173 012552 052777 000001 170060
3174 012560 005077 170056
3175 012564 105777 170044
3176 012570 100375
3177 012572 042777 000001 170040
3178 012600 042777 000004 170052
3179
3180
3181 012606 032777 020000 170022
3182 012614 001001
3183
3184 012616 104104
3185
3186 012620 032777 100000 170010
3187 012626 001001
3188
3189 012630 104114
3190
3191 012632
```

*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
BIT #BIT0,FLAG44 ;** IS THIS A 11/44
BEQ 9\$;** NO
TST CTSTFL ;** YES THIS IS 11/44. IS THIS THE CONSOLE
BEQ 9\$;** SLU
BIT #BIT03,@SWR ;** NO
BEQ TST43 ;** THIS IS THE CONSOLE SLU.SHOULD THE FRAME
BIT #BIT2,@TCSR ;** ERROR TEST BE PERFORMED
CLR R0 ;** NO
MOV #2,R1 ;CLEAR EVERYTHING
TSTB @RCSR ;SET MAINTENANCE WRAP
BMI 43\$;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
DEC R0 ;** THAT MIGHT BE IN THE PROCESS OF BEING
BNE 42\$;** RECEIVED TO FINISH AFTER MAINTENANCE
DEC R1 ;** WRAP FOR THE UART UNDER TEST IS ENABLED.
BNE 42\$;** CHECK FOR RECEIVER DONE
BR .+6
TST @RBUF ;** READ TO CLEAR DONE
BIS #BIT0,@TCSR ;SEND BREAK
CLR @TBUF ;TRANSMIT A CHARACTER TO TIME BREAK
TSTB @RCSR ;WAIT FOR RCVR DONE
BPL 1\$
BIC #BIT0,@TCSR ;CLEAR BREAK BIT
BIC #BIT2,@TCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
BIT #BIT13,@RBUF ;CHECK FOR FRAMING ERROR FLAG
BNE 2\$;BR, IF SET
ERROR+104 ;BREAK DID NOT SET FRAMING ERROR
BIT #BIT15,@RBUF ;TEST 'ERROR' FLAG
BNE 3\$;BR, IF SET
ERROR+114 ;'ERROR' FLAG DID NOT SET WITH 'OR' FLAG
3\$:

3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247

012632 000004
012634 000005
012636 052777 000004 167774
012644 005000
012646 012701 000002
012652 105777 167756
012656 100405
012660 005300
012662 001373
012664 005301
012666 001371
012670 000402
012672 005777 167740
012676 005001
012700 105201
012702 032737 000001 003002
012710 001406
012712 122701 000020
012716 001770
012720 122701 000220
012724 001765
012726 010177 167710
012732 105777 167676
012736 100375
012740 017702 167672
012744 043701 001112
012750 020102
012752 001003
012754 105701
012756 001411
012760 000747
012762 010137 001024
012766 010237 001026
012772 042777 000004 167660
013000 104105
013002
013002 042777 000004 167650

:TEST 43 TEST DATA PATH FROM XMIT TO REC USING MAINT WRAP

TST43: SCOPE

RESET #BIT2,@TCSR ;CLEAR EVERYTHING
BIS ;SET MAINTENANCE WRAP
;TRANSMIT A BINARY COUNT PATTERN - UP
;TO THE BIT POSITION INDICATED BY THE
;CONTENTS OF LOCATION '\$USWR'
CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
;** RECEIVED TO FINISH AFTER MAINTENANCE
;** WRAP FOR THE UART UNDER TEST IS ENABLED.
;** CHECK FOR RECEIVER DONE
42\$: TSTB @RCSR ;**
BMI 43\$;**
DEC R0 ;**
BNE 42\$;**
DEC R1 ;**
BNE 42\$;**
BR .+6 ;**
43\$: TST @RBUF ;** READ TO CLEAR DONE
CLR R1 ;CLEAR REGISTER FOR TEST DATA
1\$: INCB R1 ;INCREMENT THE TEST DATA
BIT #BIT0,FLAG44 ;** 11/44 CPU
BEQ 5\$;** TEST ALL DATA
CMPB #20,R1 ;** CHECK FOR CONT-P IN TEST DATA
BEQ 1\$;** DO NOT XMIT ^P
CMPB #220,R1 ;**
BEQ 1\$;** DO NOT XMIT 220
5\$: MOV R1,@TBUF ;XMIT A CHARACTER
2\$: TSTB @RCSR ;WAIT FOR RECEIVER DONE
BPL 2\$;**
MOV @RBUF,R2 ;GET RECEIVED CHARACTER
BIC @#\$USWR,R1 ;CLEAR LOWEST UNUSED DATA BIT POSITION IN TEST DATA
CMP R1,R2 ;COMPARE DATA
BNE 3\$;BR, IF NON-COMPARE
TSTB R1 ;TEST XMIT DATA FOR ZERO
BEQ 4\$;BR, IF FINISHED
BR 1\$;CONTINUE IF NOT
3\$: MOV R1,\$GDDAT ;STORE THE EXPECTED DATA
MOV R2,\$BDDAT ;STORE RECEIVED DATA
BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.
;DATA COMPARE DATA
ERROR+105
4\$: BIC #BIT2,@CTCSR ;** DISABLE MAINTENANCE MODE FOR
;** CONSOLE TO ALLOW FOR COMMUNICATION
;** WITH TERMINAL.

```

3248
3249
3250
3251 013010 000004
3252 013012 032777 000200 166020
3253 013020 001444
3254 013022 000005
3255 013024 005001
3256
3257
3258
3259 013026 105201
3260 013030 032737 000001 003002
3261 013036 001406
3262 013040 122701 000020
3263 013044 001770
3264 013046 122701 000220
3265 013052 001765
3266 013054 010177 167562
3267 013060 005000
3268 013062 105777 167546
3269 013066 100403
3270 013070 005200
3271 013072 001373
3272
3273 013074 104064
3274
3275 013076 017702 167534
3276 013102 043701 001112
3277 013106 020102
3278 013110 001003
3279 013112 105701
3280 013114 001406
3281 013116 000743
3282 013120 010137 001024
3283 013124 010237 001026
3284
3285 013130 104106

```

```

*****
: *TEST 44 TEST DATA PATHS USING WRAP CABLE
*****
TST44: SCOPE
BIT #BIT7,@SWR ;IS THIS TEST ENABLED
BEQ TST45 ;BR, IF NOT
RESET ;CLEAR EVERYTHING
CLR R1 ;CLEAR REGISTER FOR TEST DATA
;TRANSMIT A BINARY COUNT PATTERN - UP
;TO THE BIT POSITION INDICATED BY THE
;CONTENTS OF LOCATION '$USWR'
1$: INCB R1 ;INCREMENT THE TEST DATA
BIT #BIT0,FLAG44 ;11/44 CPU
BEQ 5$ ;TEST ALL DATA
CMPB #20,R1 ;CHECK TEST DATA FOR ^P
BEQ 1$ ;DO NOT XMIT ^P
CMPB #220,R1
BEQ 1$
5$: MOV R1,@TBUF ;XMIT A CHARACTER
CLR R0 ;CLEAR A TIMER
2$: TSTB @RCSR ;WAIT FOR RECEIVER DONE
BMI 3$ ;BR IF DONE
INC R0 ;INCREMENT TIMER IF NOT
BNE 2$ ;BR IF TIME REMAINS
ERROR+64 ;RECEIVER DONE NOT SET
3$: MOV @RBUF,R2 ;GET RECEIVED CHARACTER
BIC @#$USWR,R1 ;CLEAR LOWEST UNUSED DATA BIT POSITON IN TEST DATA
CMP R1,R2 ;COMPARE DATA
BNE 4$ ;BR, IF NON-COMPARE
TSTB R1 ;TEST XMIT DATA FOR ZERO
BEQ TST45 ;BR, IF FINISHED
BR 1$ ;CONTINUE IF NOT
4$: MOV R1,$GDDAT ;STORE EXPECTED DATA
MOV R2,$BDDAT ;STORE RECEIVED DATA
ERROR+106 ;DATA COMPARE ERROR WITH WRAP CABLE

```

```
3286
3287
3288
3289
3290
3291 013132 000004
3292 013134 000005
3293 013136 004737 014540
3294 013142 000340
3295 013144 017703 167500
3296 013150 017704 167470
3297 013154 017705 167516
3298 013160 017737 167466 002446
3299 013166 017737 167454 002450
3300 013174 017737 167500 002452
3301 013202 012777 013614 167440
3302 013210 012777 000200 167434
3303 013216 012777 013666 167420
3304 013224 012777 000200 167414
3305 013232 005737 002624
3306 013236 001415
3307 013240 032777 000100 165572
3308 013246 001011
3309 013250 012777 013702 167420
3310 013256 012777 000300 167414
3311 013264 052777 000100 167402
3312 013272 052777 000004 167340 1$:
3313 013300 005000
3314 013302 012701 000002
3315
3316
3317 013306 105777 167322 42$:
3318 013312 100405
3319 013314 005300
3320 013316 001373
3321 013320 005301
3322 013322 001371
3323 013324 000402
3324 013326 005777 167304 43$:
3325 013332 052777 000100 167300
3326 013340 052777 000100 167266
3327 013346 005037 002442
3328 013352 005037 002440
3329 013356 005001
3330 013360 005000
3331 013362 012702 002454
3332 013366 005077 167250
3333 013372 004737 014540
3334 013376 000140
3335
3336 013400 000240 2$:
3337 013402 000240
3338 013404 062700 000000
3339 013410 062700 000001
3340 013414 001371
3341 013416 032777 000100 167214

*****
: *TEST 45 TEST DL11-W LOGIC BY EXERCISING THE XMIT, REC, & CLOCK
*****
TST45: SCOPE
RESET ;CLEAR EVERYTHING
JSR PC,WRPSW ;SET PRIORITY TO 7
.WORD 340
MOV @TVECT,R3 ;SAVE XMIT VECTOR
MOV @RVECT,R4 ;SAVE RECEIVE VECTOR
MOV @RTCVT,R5 ;SAVE CLOCK VECTOR
MOV @TPSW,STPSW ;SAVE XMIT PSW VECTOR
MOV @RPSW,SRPSW ;SAVE RECEIVE PSW VECTOR
MOV @RTCPSW,SCPSW ;SAVE CLOCK PSW VECTOR
MOV #XMIT,@TVECT ;POINT TRANSMIT VECTOR TO TRANSMIT ROUTINE
MOV #200,@TPSW ;NO MULTIPLE INTERRUPTS ALLOWED
MOV #RCV,@RVECT ;POINT RECEIVE VECTOR TO RECEIVE ROUTINE
MOV #200,@RPSW ;NO MULT INTERRUPTS
TST CTSTFL ;IS CONSOLE UNDER TEST?
BEQ 1$ ;IF NOT SKIP CLOCK SET UP
BIT #BIT6,@SWR ;IF YES, ARE CLOCK TEST DISABLED?
BNE 1$ ;IF YES, SKIP CLOCK SET UP
MOV #CLK,@RTCVT ;POINT VECTOR TO CLOCK INTERRUPT ROUTINE
MOV #300,@RTCPSW ;NO MULT INTERRUPTS
BIS #BIT6,@LKS ;ENABLE CLOCK INTERRUPTS
1$: BIS #BIT2,@TCSR ;SET MAINTENANCE WRAP
CLR R0 ;** DELAY ENOUGH TIME TO ALLOW ANY CHARACTERS
MOV #2,R1 ;** THAT MIGHT BE IN THE PROCESS OF BEING
; ** RECEIVED TO FINISH AFTER MAINTENANCE
; ** WRAP FOR THE UART UNDER TEST IS ENABLED.
; ** CHECK FOR RECEIVER DONE
42$: TSTB @RCSR
BMI 43$
DEC R0
BNE 42$
DEC R1
BNE 42$
BR .+6
43$: TST @RBUF
BIS #BIT6,@TCSR ;ENABLE TRANSMIT INTERRUPTS
BIS #BIT6,@RCSR ;ENABLE RECEIVE INTERRUPTS
CLR XMTCNT ;CLEAR XMIT INTERRUPT COUNTER
CLR RCVCNT ;CLEAR RCV INTERRUPT COUNTER
CLR R1 ;CLEAR A REGISTER FOR TEST DATA USE
CLR R0 ;CLEAR TIMER
MOV #BUF,R2 ;POINT R2 TO RECEIVE DATA STORAGE
CLR @TBUF ;SEND FIRST CHARACTER
JSR PC,WRPSW ;SET PSW TO PRIORITY 3
.WORD 140
;WAIT FOR INTERRUPTS
;STALL
;STALL
ADD #0,R0 ;ADD INSTRUCTIONS ARE USED TO LENGTHEN LOOP TIME
ADD #1,R0 ; TO COVER THE SLOWEST BAUD RATE ON THE FASTEST CPU
BNE 2$
BIT #BIT6,@TCSR ;FINISHED ENTIRE TRANSMISSION
```

```

3342 013424 001402          BEQ      3$          ;BR, IF INTERRUPTS ARE DISABLED (FINISHED)
3343 013426 000005          RESET          ;CLEAR EVERYTHING
3344
3345 013430 104107          ERROR+107        ;TRANSMIT INTERRUPT TIMEOUT IN MAIN. DATA TEST
3346
3347 013432 023737 002442 002440 3$:  CMP      XMITCNT,RCVCNT ;COMPARE THE NUMBER OF INTERRUPTS
3348 013440 001402          BEQ      4$          ;BR, IF EQUAL
3349 013442 000005          RESET          ;CLEAR EVERYTHING
3350
3351 013444 104110          ERROR+110        ;RECEIVER DID NOT GET FULL TRANSMISSION
3352                          ; IF RCVCNT=0, NO DATA RECEIVED
3353                          ; IF RCVCNT=?, THEN (XMITCNT-RCVCNT)
3354                          ; EQUALS THE NO, OF INTERRUPTS LOST.
3355

```

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 78
 CZDLDE.P11 10-JUN-80 15:20 T45 TEST DL11-W LOGIC BY EXERCISING THE XMIT,REC, & CLOCK

SEQ 0077

```

3356
3357
3358 013446 005737 002624 4$: TST CTSTFL ;IS CONSOLE UNDER TEST?
3359 013452 001410 BEQ 5$ ;IF NOT, SKIP CLOCK COUNT CHECK
3360 013454 032777 000100 165356 BIT #BIT6,@SWR ;IF YES, ARE CLOCK TESTS DISABLED?
3361 013462 001004 BNE 5$ ;IF YES, SKIP CLOCK COUNT CHECK
3362 013464 005737 002444 TST CLKCNT ;CHECK FOR AT LESST ONE CLOCK INTERRUPT
3363 013470 001001 BNE 5$ ;BR IF INTERRUPTS OCCURRED
3364
3365 013472 104113 ERROR+113 ;NO CLOCK INTERRUPTS IN EXERCISER
3366
3367 013474 000005 5$: RESET ;CLEAR EVERYTHING
3368 013476 012700 002454 MOV #BUF,R0 ;LOAD RECEIVED DATA POINTER TO R0
3369 013502 005001 CLR R1 ;SET UP REGISTER FOR COMPARISON
3370 013504 022001 COMP: CMP (R0)+,R1 ;COMPARE XMIT & RCV DATA
3371 013506 001014 BNE 6$ ;BR, IF NOT EQUAL
3372 013510 105201 9$: INCB R1 ;INCREMENT COMPARE DATA
3373 013512 032737 000001 003002 BIT #BIT0,FLAG44 ;11/44 CPU
3374 013520 001403 BEQ 8$ ;YES SKIP
3375 013522 122701 000020 CMPB #20,R1 ;SKIP 20 DATA IF 11/44
3376 013526 001770 BEQ 9$ ;SKIP 20
3377 013530 032701 000040 8$: BIT #BIT5,R1 ;FINISHED CHECKING RECEIVED DATA?
3378 013534 001763 BEQ COMP ;BR, IF NOT FINISHED
3379 013536 000405 BR 7$ ;BR TO END OF TEST
3380 013540 014037 001026 6$: MOV -(R0),%BDDAT ;STORE BAD DATA FOR ERROR REPORT
3381 013544 010137 001024 MOV R1,%GDDAT ;STORE GOOD DATA FOR ERROR REPORT
3382 013550 104111 ERROR+111 ;DATA COMPARE ERROR IN EXERCISER
3383
3384 013552 010377 167072 7$: MOV R3,@TVECT ;RESTORE XMIT VECTOR
3385 013556 010477 167062 MOV R4,@RVECT ;RESTORE RECEIVE VECTOR
3386 013562 010577 167110 MOV R5,@RTCVT ;RESTORE CLOCK VECTOR
3387 013566 013777 002446 167056 MOV STPSW,@TPSW ;RESTORE XMIT PSW VECTOR
3388 013574 013777 002450 167044 MOV SRPSW,@RPSW ;RESTORE RECEIVE PSW VECTOR
3389 013602 013777 002452 167070 MOV SCPSW,@RTCPSW ;RESTORE CLOCK PSW VECTOR
3390 013610 000137 014344 JMP ENDEV ;GOTO NEXT TEST
3391 013614 005237 002442 XMIT: INC XMTCNT ;INCREMENT XMIT INTERRUPT COUNTER
3392 013620 105201 1$: INCB R1 ;INCREMENT TEST DATA
3393 013622 032737 000001 003002 BIT #BIT0,FLAG44 ;11/44 CPU
3394 013630 001403 BEQ 2$ ;TEST ALL DATA
3395 013632 122701 000020 CMPB #20,R1 ;CHECK DATA FOR ^P
3396 013636 001770 BEQ 1$ ;DO NOT XMIT ^P
3397 013640 032701 000040 2$: BIT #BIT5,R1 ;SEND DATA PATTERN FROM 00 --> 37
3398 013644 001404 BEQ XCONT ;BR, IF MORE DATA TO BE SENT
3399 013646 042777 000100 166764 BIC #BIT6,@TCSR ;CLEAR XMIT INTERRUPT ENABLE
3400 013654 000402 BR XRET ;RETURN, WITHOUT SENDING ANY MORE DATA
3401 013656 110177 166760 XCONT: MOVB R1,@TBUF ;SEND NEW CHARACTER
3402 013662 005000 XRET: CLR R0 ;CLEAR TIMER
3403 013664 000002 RTI ;RETURN
3404 013666 017722 166744 RCV: MOV @RBUF,(R2)+ ;STORE RECEIVED DATA
3405 013672 005237 002440 INC RCVCNT ;INCREMENT RCV INTERRUPT COUNTER
3406 013676 005000 CLR R0 ;CLEAR TIMER
3407 013700 000002 RTI ;RETURN
3408 013702 005237 002444 CLK: INC CLKCNT ;INCREMENT CLOCK INTERRUPT COUNT
3409 013706 000002 RTI ;RETURN
3410

```

```

3411
3412
3413
3414
3415
3416      176500
3417      176502
3418      176504
3419      176506
3420      003014
3421      024746
3422
3423
3424
3425
3426  013710  000004
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446  013712  032777  000004  165120  WRAP:
3447  013720  001451
3448
3449  013722  013737  177776  002434
3450  013730  012737  000340  177776
3451  013736  012706  002432
3452
3453  013742  005037  002340
3454  013746  012737  000100  002342
3455
3456  013754  004537  014254
3457
3458  013760  004537  014062
3459  013764  010437  002436
3460
3461  013770  012700  002564
3462  013774  004537  014102
3463
3464  014000  012700  002566
3465  014004  004537  014134
3466

```

```

:*****
: WRAPAROUND TESTING- AUTO INITIATION OF T/A CONSOLE TEST
:*****

```

```

RCSR58 = 176500
RBUF58 = 176502
TCSR58 = 176504
TBUF58 = 176506
BGNADD = DEVADR
ENDADD = ENDADR

```

```

:*****
:*TEST 46 TEST CONSOLE WITH WRAP AROUND
:*****

```

```

TST46: SCOPE
:*****

```

```

: MAIN LINE TEST

```

```

: THIS TEST PUTS COMMANDS OUT OVER THE SERIAL LINE WHICH IS WRAPPED
: AROUND TO THE CONSOLE AND RECEIVES THE RESPONSES

```

```

: CONTROL P IS SEND TO GET THE CONSOLE'S ATTENTION AND THEN
: T/A(A FOR APT) IS SENT. SINCE THE CPU IS HALTED DURING THE TESTING
: THE PROGRAM CAN NOT SEE THE CHARS RETURNING, THUS THE PROGRAM WILL
: SIT IN A LOOP WAITING FOR A 'B' TO BE PRINTED BY THE CONSOLE AS A
: SIGNAL TO APT THAT THE TESTING IS DONE. ALSO SINCE THE TESTING
: INVOLVES WRITTING TO THE CPU'S MEMORY A CHECKSUM IS CALCULATED AND THE
: MEMORY TO BE USED INSIDE THIS PROGRAMS SPACE IS SAVE BEFORE TESTING
: AND RESTORED AFTER TEST WITH ANOTHER CHECKSUM CALCULATION
:*****

```

```

BIT      #BIT2,@SWR      ;RUN THIS TEST ONLY IF
BEQ      10$             ;SW BIT 2 IS ON

MOV      PSW,SAVEPS     ;SAVE OLD PSW
MOV      #340,PSW       ;PUT IN NOW PSW
MOV      #JIMSTK,SP     ;USE MY STACK (SO NO CLOBER)

CLR      LOC1
MOV      #100,LOC2     ;TIMING LOOP COUNTERS

JSR      R5,SAVEETE     ;SAVE LOCATIONS T/A WRITES

JSR      R5,CHKSUM     ;CALCULATE CHECKSUM
MOV      R4,OLDSUM     ;SAVE OLD CHECKSUM

MOV      #CNTLP,R0
JSR      R5,PUTLIN     ;SEND OUT CONTROL P

MOV      #PROMPT,R0
JSR      R5,GETLIN     ;GET CRLF CONSOLE>>>

```

```

3467 014010 012700 002610      MOV    #TA,R0
3468 014014 004537 014102      JSR    R5,PUTLIN          ;SEND OUT T/A<CRLF>
3469                               JSR    R5,GETB           ;GET THE A FROM '-TESTB'
3470 014020 004537 014200      JSR    R5,RESTTE        ;RESTORE LOCATIONS T/A WRITES
3471                               JSR    R5,CHKSUM        ;RECALCULATE CHECKSUM
3472 014024 004537 014310      JSR    R5,CHKSUM
3473                               CMP    R4,OLDSUM
3474 014030 004537 014062      BEQ    10$
3475 014034 020437 002436      BEQ    10$
3476 014040 001401
3477 014042 000000      HALT
3478 014044 012706 001000      MOV    #1000,SP          ;RETURN THEIR SP
3479 014050 013737 002434      MOV    SAVEPS,PSW        ;RETURN PSW
3480 014056 000137 014400      JMP    $EOP              ;BR TO END OF PASS ROUTINE

```

```

3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498

```

```

:*****
: ROUTINE TO CALCULATE CHECKSUM ON PROGRAM
: INPUT CONDITIONS
:   BGNADD = ADDRESS TO START CHECK SUM (INCLUSIVE)
:   ENDADD = ADDRESS TO END CHECK SUM (EXCLUSIVE)
: OUTPUT CONDITIONS
:   REG4 = CHECK SUM
:*****

```

```

3499 014062 012700 003014      CHKSUM:  MOV    #BGNADD,R0      ;GET STARTING ADDRESS
3500 014066 005004              CLR    R4                  ;RESET SUM
3501 014070 062004      1$:    ADD    (R0)+,R4          ;ADD WORD TO SUM
3502 014072 022700 024746      CMP    #ENDADD,R0        ;CHECK FOR END
3503 014076 001374      BNE    1$                  ;IF NOT DONE LOOP
3504 014100 000205      RTS    R5                  ;DONE RETURN

```

```

3505
3506
3507
3508
3509
3510
3511
3512
3513
3514

```

```

:*****
: THIS ROUTINE OUTPUTS TO THE SERIAL LINE CHARS STARTING AT
: THE ADDRESS IN REG0 UNTIL IT HITS A <377>
:*****

```

```

3515 014102 112001      PUTLIN:  MOVB   (R0)+,R1        ;GET DATA
3516 014104 001412      BEQ    10$                ;END OF DATA
3517 014106 004537 014234      1$:    JSR    R5,TIMER          ;PROVIDE FOR TIMEOUT
3518 014112 032737 000200 176504  BIT    #BIT7,TC5R58      ;TEST FOR XMIT READY
3519 014120 001772      BEQ    1$                  ;WAIT FOR XMIT READY
3520 014122 110137 176506      MOVB   R1,TBUF58         ;OUTPUT CHAR
3521 014126 000137 014102      JMP    PUTLIN            ;REPETE
3522 014132 000205      RTS    R5                  ;RETURN

```


3523
 3524
 3525
 3526
 3527
 3528
 3529
 3530
 3531
 3532 014134 112001
 3533 014136 105201
 3534 014140 001416
 3535 014142 105301
 3536 014144 004537 014234
 3537 014150 032737 000200 176500
 3538 014156 001772
 3539 014160 113702 176502
 3540 014164 042702 000200
 3541 014170 120102
 3542 014172 001760
 3543 014174 000000
 3544 014176 000205
 3545
 3546
 3547
 3548
 3549
 3550
 3551
 3552
 3553
 3554
 3555 014200 004537 014234
 3556 014204 032737 000200 176500
 3557 014212 001772
 3558 014214 113702 176502
 3559 014220 042702 000200
 3560 014224 122702 000102
 3561 014230 001363
 3562 014232 000205
 3563
 3564
 3565
 3566
 3567
 3568
 3569
 3570
 3571
 3572
 3573 014234 005337 002340
 3574 014240 001004
 3575 014242 005337 002342
 3576 014246 001001
 3577 014250 000000
 3578 014252 000205

```

:*****
: THIS ROUTINE INPUTS A CHARS FROM THE SERIAL LINE AND COMPARES
: IT WITH THE EXPECTED VALUES POINTED TO BY REGO UNTIL NULL<00>
:*****

```

```

GETLIN:      MOVB      (R0)+,R1      ;GET EXPECTED CHAR
             INCB      R1
             BEQ       10$          ;IF NULL EXIT
             DECB      R1
1$:          JSR       R5,TIMER      ;PROVIDE FOR TIMEOUT
             BIT       #BIT7,RCSR58 ;TEST FOR REC READY
             BEQ       1$           ;WAIT FOR REC READY
             MOVB      RBUF58,R2
             BIC       #BIT7,R2     ;STRIP PARITY
             CMPB      R1,R2       ;ARE THEY THE SAME
             BEQ       GETLIN      ;SAME GET MORE
             HALT      ;NOT SAME HALT
10$:         RTS        R5          ;RETURN

```

```

:*****
: THIS ROUTINE INPUTS CHARS UNTIL IT GETS THE CHAR 'B'
: AND THEN RETURNS
:*****

```

```

GETB:        JSR       R5,TIMER      ;OUT TIMING LOOP OUT
             BIT       #BIT7,RCSR58 ;TEST OFR REC READY
             BEQ       GETB         ;NOT DONE YET
             MOVB      RBUF58,R2
             BIC       #BIT7,R2     ;STRIP PARITY
             CMPB      #102,R2     ;CHECK FOR 'B'
             BNE       GETB         ;NOT 'B' REPETE
             RTS        R5          ;WAS 'B' RETURN

```

```

:*****
: THIS ROUTINE IS USED AS A TIME OUT FEATURE
: WHEN THE TIMING LOOPS BOTH REACH 0 THIS ROUTINE WILL
: CAUSE A HALT
:*****

```

```

TIMER:       DEC       LOC1         ;DECREPNT TIMING LOOPS
             BNE       10$
             DEC       LOC2
             BNE       10$
10$:         HALT      ;IF ZERO THIS ROUTINE
             RTS        R5          ;EXECUTED R3 TIMES

```

```

3579
3580
3581
3582
3583
3584
3585
3586
3587
3588 014254 013737 000000 002344 SAVETE:      MOV      0,SAVED      ;SAVE LOCATION 0
3589 014262 012702 000002          MOV      #2,R2       ;SET UP INDIRECT PNTER
3590 014266 012701 002346          MOV      #SAVLOC,R1  ;SET UP STORAGE LOC. PNTER
3591 014272 011221          1$:      MOV      (R2),(R1)+  ;GET WORD AND SAVE
3592 014274 000241          CLC                         ;DONT ROT IN ANY BITS
3593 014276 006102          ROL      R2               ;SET NEXT ADDRESS
3594 014300 020227 024746          CMP      R2,#ENDADD     ;SEE IF AT END
3595 014304 100772          BMI      1$              ;NO REPETE
3596 014306 000205          RTS      R5
3597
3598
3599
3600
3601
3602
3603
3604
3605 014310 013737 002344 000000 RESTTE:    MOV      SAVED,0
3606 014316 012702 000002          MOV      #2,R2       ;SET UP INDIRECT PNTER
3607 014322 012701 002346          MOV      #SAVLOC,R1  ;SET UP STORAGE LOC. PNTER
3608 014326 012112          1$:      MOV      (R1)+,(R2)  ;GET WORD AND RESTORE
3609 014330 000241          CLC                         ;DONT ROT IN ANY BITS
3610 014332 006102          ROL      R2               ;SET NEXT ADDRESS
3611 014334 020227 024746          CMP      R2,#ENDADD     ;SEE IF AT END
3612 014340 100772          BMI      1$              ;NO REPETE
3613 014342 000205          RTS      R5
3614
3615
    
```

```

3616
3617 ;END OF DEVICE PASS ROUTINE
3618 .EVEN
3619 014344 005037 001002 ENDEV: CLR $STNM ;CLEAR TEST NO. COUNT FOR SCOPE ROUTINE
3620 014350 005237 001076 INC $DEVCT ;INCREMENT DEVICE COUNTER
3621 014354 023737 002630 001076 CMP TMP2,$DEVCT ;ALL DEVICES TESTED
3622 014362 001002 BNE NOEOP ;BR, IF NO
3623 014364 000137 013712 JMP WRAP ;EXECUTE WRAP AROUND AFTER ALL DEVICES
3624 014370 005037 002624 NOEOP: CLR CTSTFL ;CLEAR CONSOLE UNDER TEST FLAG
3625 014374 000137 004050 .MP TSTDEV ;GO TEST NEXT DEVICE
3626
3627
3628
3629 .SBTTL END OF PASS ROUTINE
3630
3631 ::*****
3632 ;+INCREMENT THE PASS NUMBER ($PASS)
3633 ;+IF THERES A MONITOR GO TO IT
3634 ;+IF THERE ISN'T JUMP TO GOAGIN
3635
3636 014400 $EOP:
3637 014400 000004 SCOPE
3638 014402 005037 001002 CLR $STNM ;;ZERO THE TEST NUMBER
3639 014406 005237 001074 INC $PASS ;;INCREMENT THE PASS NUMBER
3640 014412 042737 100000 001074 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
3641 014420 005327 DEC (PC)+ ;.LOOP?
3642 014422 000001 $EOPCT: .WORD 1
3643 014424 003015 BGT $DOAGN ;;YES
3644 014426 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
3645 014430 000001 $ENDCT: .WORD 1
3646 014432 014422 $EOPCT
3647 014434 104401 014470 TYPE ,ENDMG ;;TYPE 'END PASS'
3648 014440 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
3649 014444 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
3650 014446 000005 RESET ;;CLEAR THE WORLD
3651 014450 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
3652 014452 000240 NOP ;;SAVE ROOM
3653 014454 000240 NOP ;;FOR
3654 014456 000240 NOP ;;ACT11
3655 014460 $DOAGN:
3656 014460 000137 JMP @(PC)+ ;;RETURN
3657 014462 014504 $RTNAD: .WORD GOAGIN
3658 014464 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
3659 014470 014470 .EVEN
3660 014470 005015 047105 020104 ENDMG: .ASCIZ <CR><LF>/END PASS /
3661 014476 040520 051523 000040

```

```

3662
3663 014504 005037 001076 GOAGIN: CLR $DEVCT ;CLEAR DEVICE COUNT
3664 014510 022737 000001 002630 CMP #1,TMP2 ;IS THERE ONLY ONE DEVICE UNDER TEST?
3665 014516 001004 BNE RSTRT ;BR, IF NOT
3666 014520 012706 001000 MOV #1000,SP ;RESET STACK POINTER
3667 014524 000137 004172 JMP TST1 ;GO DO ANOTHER PASS
3668
3669 014530 005037 001100 RSTRT: CLR $UNIT ;CLEAR UNIT NUMBER
3670 014534 000137 003774 JMP BEGIN
3671
3672 014540 011646 WRPSW: MOV(SP),-(SP) ;COPY RETURN PC
3673 014542 013616 MOV @2(SP)+,(SP) ;MOVE NEW PSW TO STACK
3674 014544 062746 000002 ADD #2,-(SP) ;ADJUST JSR RETURN
3675 014550 000002 RTI ;POP RETURN PC & NEW PSW
3676
3677 ;SUBROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
3678
3679 014552 012600 CATCH: MOV (SP)+,R0 ;GET ADDRESS OF TRAP VECTOR + 4
3680 014554 162700 000004 SUB #4,R0 ;ADJUST TO POINT TO TRAP ADDRESS
3681 014560 010037 002622 MOV R0,BDVECT ;STORE TRAP OR INTERRUPT ADDRESS
3682 014564 016637 000002 002620 MOV 2(SP),OLDPC ;GET PC WHERE TRAP OR INTERRUPT OCCURRED
3683 014572 104112 ERROR+112 ;REPORT ERROR
3684
3685 014574 000000 HALT ;PROGRAM MUST BE RESTARTED AT THIS POINT
3686

```

```

3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700 014576
3701 014576 105237 001003
3702 014602 001775
3703 014604 013777 001002 164230
3704 014612 005237 001012
3705 014616 011637 001016
3706 014622 162737 000002 001016
3707 014630 117737 164162 001014
3708 014636 032777 020000 164174
3709 014644 001004
3710 014646 004737 014760
3711 014652 104401 001063
3712 014656
3713 014656 122737 000001 001106
3714 014664 001007
3715 014666 113737 001014 014700
3716 014674 004737 015462
3717 014700 000
3718 014701 000
3719 014702 000777
3720 014704 005777 164130
3721 014710 100001
3722 014712 000000
3723 014714 104406
3724 014716 032777 001000 164114
3725 014724 001402
3726 014726 013716 001010
3727 014732 005737 001060
3728 014736 001402
3729 014740 013716 001060
3730 014744
3731 014744 022737 014450 000042
3732 014752 001001
3733 014754 000000
3734 014756
3735 014756 000002
3736

```

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND ADDRESS OF THE ERROR CALL
*AND GO TO $ERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW09=1      LOOP IN ERROR
*CALL
*          ERROR+N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
*****

```

```

$ERROR:
7$:      INCB      $ERFLG      ;SET THE ERROR FLAG
        BEQ       7$          ;DON'T LET FLAG GO TO ZERO
        MOV      $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
        INC      $ERTTL      ;INCREMENT ERROR COUNT
        MOV      (SP),$ERRPC   ;GET ADDRESS OF ERROR INSTRUCTION
        SUB      #2,$ERRPC
        MOV      @ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
        BIT      #BIT13,@SWR   ;SKIP TYPEOUT IF SET
        BNE     20$          ;SKIP TYPEOUTS
        JSR     PC,$ERRTYP    ;GO TO USER ERROR ROUTINE
        TYPE    ,$CRLF

20$:
        CMPB    #APTENV,$ENV   ;RUNNING IN APT MODE
        BNE     2$            ;NO, SKIP APT ERROR REPORT
        MOV      $ITEMB,21$    ;SET ITEM NUMBER AS ERROR NUMBER
        JSR     PC,$ATY4      ;REPORT FATAL ERROR TO APT

21$:
        .BYTE   0
        .BYTE   0

22$:
        BR      22$          ;APT ERROR LOOP

2$:      TST      @SWR        ;HALT ON ERROR
        BPL     3$          ;SKIP IF CONTINUE
        HALT    ;HALT ON ERROR!

3$:      CKSWR
        BIT     #BIT09,@SWR   ;TEST FOR CHANGE IN SOFT-SWR
        BEQ    4$          ;LOOP ON ERROR SWITCH SET?
        BR     IF NO
        MOV    $LPERR,(SP)   ;FUDGE RETURN FOR LOOPING
        TST    $ESCAPE       ;CHECK FOR AN ESCAPE ADDRESS
        BEQ    5$          ;BR IF NONE
        MOV    $ESCAPE,(SP) ;FUDGE RETURN ADDRESS FOR ESCAPE

5$:
        CMP     #SENDAD,@#42  ;ACT-11 AUTO-ACCEPT?
        BNE    6$          ;BR IF NO
        HALT   ;YES

6$:      RTI                ;RETURN

```

3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785

014760
014760 104401 001063
014764 010046
014766 005000
014770 153700 001014
014774 001004
014776 013746 001016
015002 104402
015004 000426
015006 005300
015010 006300
015012 006300
015014 006300
015016 062700 001146
015022 012037 015032
015026 001404
015030 104401
015032 000000
015034 104401 001063
015040 012037 015050
015044 001404
015046 104401
015050 000000
015052 104401 001063
015056 011000
015060 001004
015062 012600
015064 104401 001063
015070 000207
015072
015072 013046
015074 104402
015076 005710
015100 001770
015102 104401 015110
015106 000771
015110 020040 000
015114

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

3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831

015114 012737 015254 000024
015122 012737 000340 000026
015130 010046
015132 010146
015134 010246
015136 010346
015140 010446
015142 010546
015144 017746 163670
015150 010637 015260
015154 012737 015166 000024
015162 000000
015164 000776

015166 012737 015254 000024
015174 013706 015260
015200 012677 163634
015204 012605
015206 012604
015210 012603
015212 012602
015214 012601
015216 012600
015220 012737 015114 000024
015226 012737 000340 000026
015234 005037 015260
015240 005237 015260
015244 001375
015246 104401
015250 015262
015252 000002
015254 000000
015256 000776
015260 000000
015262 005015 047520 042527
015270 000122

```
.SBTTL POWER DOWN AND UP ROUTINES
:;*****
;*POWER DOWN ROUTINE
:;*****
$PWRDN: MOV    $ILLUP,@#PWRVEC ;SET FOR FAST UP
        MOV    #340,@#PWRVEC+2 ;PRIO:7
        MOV    R0,-(SP)        ;PUSH R0 ON STACK
        MOV    R1,-(SP)        ;PUSH R1 ON STACK
        MOV    R2,-(SP)        ;PUSH R2 ON STACK
        MOV    R3,-(SP)        ;PUSH R3 ON STACK
        MOV    R4,-(SP)        ;PUSH R4 ON STACK
        MOV    R5,-(SP)        ;PUSH R5 ON STACK
        MOV    @SWR,-(SP)       ;PUSH @SWR ON STACK
        MOV    SP,$SAVR6       ;SAVE SP
        MOV    #$PWRUP,@#PWRVEC ;SET UP VECTOR
        HALT
        BR     -2              ;HANG UP

:;*****
;*POWER UP ROUTINE
:;*****
$PWRUP: MOV    $ILLUP,@#PWRVEC ;SET FOR FAST DOWN
        MOV    $SAVR6,SP       ;GET SP
        MOV    (SP)+,@SWR      ;POP STACK INTO @SWR
        MOV    (SP)+,R5
        MOV    (SP)+,R4        ;POP STACK INTO R4
        MOV    (SP)+,R3        ;POP STACK INTO R3
        MOV    (SP)+,R2        ;POP STACK INTO R2
        MOV    (SP)+,R1        ;POP STACK INTO R1
        MOV    (SP)+,R0        ;POP STACK INTO R0
        MOV    #$PWRDN,@#PWRVEC ;SET UP THE POWER DOWN VECTOR
        MOV    #340,@#PWRVEC+2 ;PRIO:7
        CLR    $SAVR6         ;WAIT LOOP FOR THE TTY
1$:      INC    $SAVR6         ;WAIT FOR THE INC
        BNE    1$             ;OF WORD
        TYPE   $SAVR6         ;REPORT THE POWER FAILURE
        $PWRMG: .WORD $POWER ;POWER FAIL MESSAGE POINTER
        RTI
$ILLUP: HALT                  ;THE POWER UP SEQUENCE WAS STARTED
        BR     -2              ;BEFORE THE POWER DOWN WAS COMPLETE
        $SAVR6: 0              ;PUT THE SP HERE
$POWER: .ASCIZ <15><12>'POWER'
```

3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877

015272
015272 104406
015274 032777 040000 163536
015302 001052
015304 000416
015306 013746 000004
015312 012737 015332 000004
015320 005737 177060
015324 012637 000004
015330 000421
015332 022626
015334 012637 000004
015340 000407
015342
015342 105737 001003
015346 001412
015350 032777 001000 163462
015356 001404
015360 013737 001010 001006
015366 000420
015370 105037 001003
015374 105237 001002
015400 113737 001002 001072
015406 011637 001006
015412 011637 001010
015416 005037 001060
015422 112737 000001 001015
015430 013777 001002 163404
015436 013716 001006
015442 000002

```
.SBTTL SCOPE HANDLER ROUTINE
*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW09=1 LOOP ON ERROR
*CALL
* SCOPE ;:SCOPE=IOT

$SCOPE:
CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
1$: BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?
BNE $OVER ;:YES IF SW14=1
;#####START OF CODE FOR THE XOR TESTER#####
$X*STR: BR 6$ ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
;THIS INSTRUCTION TO A 'NOP' (NOP=240)
MOV @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5$,@#ERRVEC ;:SET FOR TIMEOUT
TST @#177060 ;:TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
BR $SVLAD ;:GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
BR 7$ ;:LOOP ON THE PRESENT TEST
6$;#####END OF CODE FOR THE XOR TESTER#####
2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
BEQ $SVLAD ;:BR IF NO
BIT #BIT09,@SWR ;:LOOP ON ERROR?
BEQ 4$ ;:BR IF NO
7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
$SVLAD: INCB $TSTNM ;:COUNT TEST NUMBERS
MOVB $TSTNM,$TESTN ;:SET TEST NUMBER IN APT MAILBOX
MOV (SP),$LPADR ;:SAVE SCOPE LOOP ADDRESS
MOV (SP),$LPERR ;:SAVE ERROR LOOP ADDRESS
CLR $ESCAPE ;:CLEAR THE ESCAPE FROM ERROR ADDRESS
MOVB #1,$ERMAX ;:ONLY ALLOW ONE(1) ERROR ON NEXT TEST
$OVER: MOV $TSTNM,@DISPLAY ;:DISPLAY TEST NUMBER
MOV $LPADR,(SP) ;:FUDGE RETURN ADDRESS
RTI ;:FIXES PS
```


3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933

015444 112737 000001 015710
015452 112737 000001 015706
015460 000403
015462 112737 000001 015710
015470
015470 010046
015472 010146
015474 105737 015706
015500 001450
015502 122737 000001 001106
015510 001031
015512 132737 000100 001107
015520 001425
015522 017600 000004
015526 062766 000002 000004
015534 005737 001066
015540 001375
015542 010037 001102
015546 105720
015550 001376
015552 163700 001102
015556 006200
015560 010037 001104
015564 012737 000004 001066
015572 000413
015574 017637 000004 015620
015602 062766 000002 000004
015610 013746 177776
015614 004737 015712
015620 000000
015622
015622 105737 015710
015626 001413
015630 005737 001106
015634 001410
015636 005737 001066
015642 001375
015644 017637 000004 001070
015652 005237 001066
015655 062766 000002 000004
015664 105037 015710
015670 105037 015707
015674 105037 015706
015700 012601
015707 012600
015704 000207
015706 000
015707 000
015710 000
015712

:SBTTL APT COMMUNICATIONS ROUTINE

\$ATY1: MOVB #1,\$FFLG ;TO REPORT FATAL ERROR
\$ATY3: MOVB #1,\$MFLG ;TO TYPE A MESSAGE
BR \$ATYC
\$ATY4: MOVB #1,\$FFLG ;TO ONLY REPORT FATAL ERROR
\$ATYC:
MOV RO,-(SP) ;PUSH RO ON STACK
MOV R1,-(SP) ;PUSH R1 ON STACK
TSTB \$MFLG ;SHOULD TYPE A MESSAGE?
BEQ 5\$;IF NOT: BR
CMPB #APTENV,\$ENV ;OPERATING UNDER APT?
BNE 3\$;IF NOT: BR
BITB #APTSPOOL,\$ENVM ;SHOULD SPOOL MESSAGE?
BEQ 3\$;IF NOT: BR
MOV @4(SP),RO ;GET MESSAGE ADDRESS
ADD #2,4(SP) ;BUMP RETURN ADDRESS
1\$: TST \$MSGTYPE ;SEE IF DONE W/ LAST XMISSION?
BNE 1\$;IF NOT: WAIT
MOV RO,\$MSGAD ;PUT ADDRESS IN MAILBOX
2\$: TSTB (RO)+ ;FIND END OF MESSAGE
BNE 2\$
SUB \$MSGAD,RO ;SUB START OF MESSAGE
ASR RO ;GET MESSAGE LENGTH IN WORDS
MOV RO,\$MSGGLT ;PUT LENGTH IN MAILBOX
MOV #4,\$MSGTYPE ;TELL APT TO TAKE MESSAGE
BR 5\$
3\$: MOV @4(SP),4\$;PUT MSG ADDR IN JSR LINKAGE
ADD #2,4(SP) ;BUMP RETURN ADDRESS
MOV 177776,-(SP) ;PUSH 177776 ON STACK
JSR PC,\$TYPE ;CALL TYPE MACRO
4\$: .WORD 0
5\$:
10\$: TSTB \$FFLG ;SHOULD REPORT FATAL ERROR?
BEQ 12\$;IF NOT: BR
TST \$ENV ;RUNNING UNDER APT?
BEQ 12\$;IF NOT: BR
11\$: TST \$MSGTYPE ;FINISHED LAST MESSAGE?
BNE 11\$;IF NOT: WAIT
MOV @4(SP),\$FATAL ;GET ERROR #
INC \$MSGTYPE ;TELL APT TO TAKE ERROR
12\$: ADD #2,4(SP) ;BUMP RETURN ADDRESS
CLRB \$FFLG ;CLEAR FATAL FLAG
CLRB \$LFLG ;CLEAR LOG FLAG
CLRB \$MFLG ;CLEAR MESSAGE FLAG
MOV (SP)+,R1 ;POP STACK INTO R1
MOV (SP)+,RO ;POP STACK INTO R1
RTS PC ;RETURN
\$MFLG: .BYTE 0
\$LFLG: .BYTE 0 ;LOG FLAG
\$FFLG: .BYTE 0 ;FATAL FLAG

.EVEN

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 90
CZDLDE.P11 10-JUN-80 15:20 APT COMMUNICATIONS ROUTINE

SEQ 0089

3934	000200	APTSIZE=200
3935	000001	APTENV=001
3936	000100	APTSPOOL=100
3937	000040	APTCSUP=040
3938		
3939		

```

3940 .SBTTL TYPE ROUTINE
3941
3942 *****
3943 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3944 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3945 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3946 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3947 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3948
3949 *CALL:
3950 *1) USING A TRAP INSTRUCTION
3951 * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3952
3953 *OR
3954 * TYPE
3955 * MESADR
3956
3957 015712 105737 001057 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
3958 015716 100002 BPL 1$ ;;BR IF YES
3959 015720 000000 HALT ;;HALT HERE IF NO TERMINAL
3960 015722 000430 BR 3$ ;;LEAVE
3961 015724 010046 1$: MOV RO,-(SP) ;;SAVE RO
3962 015726 017600 000002 MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
3963 015732 122737 000001 001106 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
3964 015740 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
3965 015742 132737 000100 001107 BITB #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
3966 015750 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
3967 015752 010037 015762 MOV RO,61$ ;;SETUP MESSAGE ADDRESS FOR APT
3968 015756 004737 015452 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
3969 015762 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
3970 015764 132737 000040 001107 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
3971 015772 001003 BNE 60$ ;;YES,SKIP TYPE OUT
3972 015774 112046 2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
3973 015776 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
3974 016000 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
3975 016002 012600 60$: MOV (SP)+,RO ;;RESTORE RO
3976 016004 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
3977 016010 000002 RTI ;;RETURN
3978 016012 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
3979 016016 001430 BEQ 8$
3980 016020 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
3981 016024 001006 BNE 5$
3982 016026 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
3983 016030 104401 TYPE ;;TYPE A CR AND LF
3984 016032 001063 $CRLF
3985 016034 105037 016240 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
3986 016040 000755 BR 2$ ;;GET NEXT CHARACTER
3987 016042 004737 016124 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
3988 016046 123726 001056 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
3989 016052 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
3990 016054 013746 001054 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
3991 ;;AND THE NULL CHAR.
3992 016060 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
3993 016064 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
3994 016066 004737 016124 JSR PC,$TYPEC ;;GO TYPE A NULL
3995 016072 105337 016240 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT

```

```

3996 016076 000770          BR      7$          ;;LOOP
3997
3998          ;HORIZONTAL TAB PROCESSOR
3999
4000 016100 112716 000040    8$:     MOVB   #' (SP)          ;;REPLACE TAB WITH SPACE
4001 016104 004737 016124    9$:     JSR    PC,$TYPEC          ;;TYPE A SPACE
4002 016110 132737 000007 016240    BITB   #7,$CHARCNT          ;;BRANCH IF NOT AT
4003 016116 001372          BNE    9$          ;;TAB STOP
4004 016120 003726          TST    (SP)+          ;;POP SPACE OFF STACK
4005 016122 000724          BR     2$          ;;GET NEXT CHARACTER
4006 016124 105777 162720    $TYPEC: TSTB   @STPS          ;;WAIT UNTIL PRINTER IS READY
4007 016130 100375          BPL    $TYPEC
4008 016132 116677 000002 162712    MOVB   2(SP),@$(PB          ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4009 016140 105777 162700    TSTB   @$(TKS          ;;SEE IF KEYBOARD IS TALKING.
4010 016144 100021          BPL    2$          ;;BRANCH IF IT ISN'T.
4011 016146 017746 162674    MOV    @$(TKB,-(SP)          ;;PUSH CHARACTER ONTO STACK.
4012 016152 042716 177600    BIC    #177600,(SP)          ;;BIT CLEAR TOP BYTE AND PARITY BIT.
4013 016156 022726 000C23    CMP    #23,(SP)+          ;;SEE IF THIS IS A ^S.
4014 016162 001012          BNE    2$          ;;BRANCH TO CONTINUE IF IT ISN'T.
4015 016164 105777 162654    3$:    TSTB   @$(TKS          ;;WAIT FOR ANOTHER INPUT.
4016 016170 100375          BPL    3$          ;;BRANCH BACK IF NOT READY.
4017 016172 017746 162650    MOV    @$(TKB,-(SP)          ;;PUSH NEXT CHARACTER ON STACK.
4018 016176 042716 177600    BIC    #177600,(SP)          ;;BIT CLEAR TOP BYTE AND PARITY BIT.
4019 016202 022726 000021    CMP    #21,(SP)+          ;;SEE IF THIS IS A ^Q.
4020 016206 001366          BNE    3$          ;;BRANCH BACK FOR MORE WAIT IF NOT.
4021 016210 122766 000015 000002 2$:    CMPB   #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
4022 016216 001003          BNE    1$          ;;BRANCH IF NO
4023 016220 105037 016240    CLRB   $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
4024 016224 000406          BR     $TYPEX          ;;EXIT
4025 016226 122766 000012 000002 1$:    CMPB   #LF,2(SP)          ;;IS CHARACTER A LINE FEED?
4026 016234 001402          BEQ    $TYPEX          ;;BRANCH IF YES
4027 016236 105227          INCB   (PC)+          ;;COUNT THE CHARACTER
4028 016240 000000    $CHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
4029 016242 000207    $TYPEX: RTS      PC

```

```

4030
4031          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
4032
4033          ;*****
4034          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4035          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
4036          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4037          ;*CALL:
4038          ;*      MOV     NUM,-(SP)          ;;NUMBER TO BE TYPED
4039          ;*      TYPOS          ;;CALL FOR TYPEOUT
4040          ;*      .BYTE   N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4041          ;*      .BYTE   M          ;;M=1 OR 0
4042          ;*                                     ;;1=TYPE LEADING ZEROS
4043          ;*                                     ;;0=SUPPRESS LEADING ZEROS
4044
4045          ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
4046          ;*$TYPOS OR $TYPOC
4047          ;*CALL:
4048          ;*      MOV     NUM,-(SP)          ;;NUMBER TO BE TYPED
4049          ;*      TYPON          ;;CALL FOR TYPEOUT
4050
4051          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

4052          ;*CALL:
4053          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
4054          ;*      TYPOC      ;;CALL FOR TYPEOUT
4055
4056 016244 017646 000000          $TYPOS: MOV      @(SP),-(SP)      ;;PICKUP THE MODE
4057 016250 116637 000001 016467      MOV      1(SP),%OFILL      ;;LOAD ZERO FILL SWITCH
4058 016256 112637 016471          MOV      (SP)+,%SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
4059 016262 062716 000002          ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
4060 016266 000406          BR      $TYPON
4061 016270 112737 000001 016467      $TYPOC: MOV      #1,%OFILL      ;;SET THE ZERO FILL SWITCH
4062 016276 112737 000006 016471      MOV      #6,%SOMODE+1      ;;SET FOR SIX(6) DIGITS
4063 016304 112737 000005 016466      $TYPON: MOV      #5,%OCNT      ;;SET THE ITERATION COUNT
4064 016312 010346          MOV      R3,-(SP)      ;;SAVE R3
4065 016314 010446          MOV      R4,-(SP)      ;;SAVE R4
4066 016316 010546          MOV      R5,-(SP)      ;;SAVE R5
4067 016320 113704 016471          MOV      %SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
4068 016324 005404          NEG      R4
4069 016326 062704 000006          ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
4070 016332 110437 016470          MOV      R4,%SOMODE      ;;SAVE IT FOR USE
4071 016336 113704 016467          MOV      %OFILL,R4      ;;GET THE ZERO FILL SWITCH
4072 016342 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
4073 016346 005003          CLR      R3      ;;CLEAR THE OUTPUT WORD
4074 016350 006105          1$: ROL      R5      ;;ROTATE MSB INTO 'C'
4075 016352 000404          BR      3$
4076 016354 006105          2$: ROL      R5      ;;FORM THIS DIGIT
4077 016356 006105          ROL      R5
4078 016360 006105          ROL      R5
4079 016362 010503          MOV      R5,R3
4080 016364 006103          3$: ROL      R3      ;;GET LSB OF THIS DIGIT
4081 016366 105337 016470          DECB      %SOMODE      ;;TYPE THIS DIGIT?
4082 016372 100016          BPL      7$      ;;BR IF NO
4083 016374 042703 177770          BIC      #177770,R3      ;;GET RID OF JUNK
4084 016400 001002          BNE      4$      ;;TEST FOR 0
4085 016402 005704          TST      R4      ;;SUPPRESS THIS 0?
4086 016404 001403          BEQ      5$      ;;BR IF YES
4087 016406 005204          4$: INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S
4088 016410 052703 000060          BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
4089 016414 052703 000040          5$: BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
4090 016420 110337 016464          MOV      R3,8$      ;;SAVE FOR TYPING
4091 016424 104401 016464          TYPE      ,8$      ;;GO TYPE THIS DIGIT
4092 016430 105337 016466          7$: DECB      %OCNT      ;;COUNT BY 1
4093 016434 003347          BGT      2$      ;;BR IF MORE TO DO
4094 016436 002402          BLT      6$      ;;BR IF DONE
4095 016440 005204          INC      R4      ;;INSURE LAST DIGIT ISN'T A BLANK
4096 016442 000744          BR      2$      ;;GO DO THE LAST DIGIT
4097 016444 012605          6$: MOV      (SP)+,R5      ;;RESTORE R5
4098 016446 012604          MOV      (SP)+,R4      ;;RESTORE R4
4099 016450 012603          MOV      (SP)+,R3      ;;RESTORE R3
4100 016452 016666 000002 000004          MOV      2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
4101 016460 012616          MOV      (SP)+,(SP)
4102 016462 000002          RTI      ;;RETURN
4103 016464 000          8$: .BYTE 0      ;;STORAGE FOR ASCII DIGIT
4104 016465 000          .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
4105 016466 000          %OCNT: .BYTE 0      ;;OCTAL DIGIT COUNTER
4106 016467 000          %OFILL: .BYTE 0      ;;ZERO FILL SWITCH
4107 016470 000000          %SOMODE: .WORD 0      ;;NUMBER OF DIGITS TO TYPE

```

4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163

016472 022737 000176 001040
016500 001074
016502 105777 162336
016506 100071
016510 117746 162332
016514 042716 177600
016520 022726 000007
016524 001062
016526 123727 001034 000001
016534 001456

016536 104401 017217
016542 104401 017224
016546 013746 000176
016552 104402
016554 104401 017235
016560 005046
016562 005046
016564 105777 162254
016570 100375

016572 117746 162250
016576 042716 177600

016602 021627 000025
016606 001005
016610 104401 017212
016614 062706 000006
016620 000757

016622 021627 000015
016626 001022
016630 005766 000004
016634 001403
016636 016677 000002 162174
016644 062706 000006
016650 104401 001063
016654 123727 001035 000001
016662 001003
016664 012777 000100 162152
016672 000002

```
.SBTTL TTY INPUT ROUTINE
:*****
:ENABL LSB
:*****
:SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
:ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
:SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
:WHEN OPERATING IN TTY FLAG MODE.
$CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
        BNE 15$ ;;BRANCH IF NO
        TSTB @STKS ;;CHAR THERE?
        BPL 15$ ;;IF NO, DON'T WAIT AROUND
        MOVB @STKB,-(SP) ;;SAVE THE CHAR
        BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
        CMP #7,(SP)+ ;;IS IT A CONTROL G?
        BNE 15$ ;;NO, RETURN TO USER
        CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
        BEQ 15$ ;;BRANCH IF YES

$GTSWR: TYPE ,SCNTLG ;;ECHO THE CONTROL-G (^G)
        TYPE ,SMSWR ;;TYPE CURREN. CONTENTS
        MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
        TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
        TYPE ,SMNEW ;;PROMPT FOR NEW SWR
19$: CLR -(SP) ;;CLEAR COUNTER
      CLR -(SP) ;;THE NEW SWR
7$: TSTB @STKS ;;CHAR THERE?
    BPL 7$ ;;IF NOT TRY AGAIN

        MOVB @STKB,-(SP) ;;PICK UP CHAR
        BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII

9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
    BNE 10$ ;;BRANCH IF NOT
    TYPE ,SCNTLU ;;YES, ECHO CONTROL-U (^U)
20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
    BR 19$ ;;LET'S TRY IT AGAIN

10$: CMP (SP),#15 ;;IS IT A <CR>?
     BNE 16$ ;;BRANCH IF NO
     TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
     BEQ 11$ ;;BRANCH IF YES
     MOV 2(SP),@SWR ;;SAVE NEW SWR
11$: ADD #6,SP ;;CLEAR UP STACK
14$: TYPE ,SCRLF ;;ECHO <CR> AND <LF>
     CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
     BNE 15$ ;;BRANCH IF NOT
     MOV #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
15$: RTI ;;RETURN
```

```

4164 016674 004737 016124      16$: JSR    PC,$TYPEC      ;;ECHO CHAR
4165 016700 021627 000060      CMP    (SP),#60          ;;CHAR < 0?
4166 016704 002420              BLT    18$              ;;BRANCH IF YES
4167 016706 021627 000067      CMP    (SP),#67          ;;CHAR > 7?
4168 016712 003015              BGT    18$              ;;BRANCH IF YES
4169 016714 042726 000060      BIC    #60,(SP)+        ;;STRIP-OFF ASCII
4170 016720 005766 000002      TST    2(SP)            ;;IS THIS THE FIRST CHAR
4171 016724 001403              BEQ    17$              ;;BRANCH IF YES
4172 016726 006316              ASL    (SP)              ;;NO, SHIFT PRESENT
4173 016730 006316              ASL    (SP)              ;;CHAR OVER TO MAKE
4174 016732 006316              ASL    (SP)              ;;ROOM FOR NEW ONE.
4175 016734 005766 000002      17$: INC    2(SP)          ;;KEEP COUNT OF CHAR
4176 016740 056616 177776      BIS    -2(SP),(SP)      ;;SET IN NEW CHAR
4177 016744 000707              BR     7$                ;;GET THE NEXT ONE
4178 016746 104401 001062      18$: TYPE  $QUES         ;;TYPE ?<CR><LF>
4179 016752 000720              BR     20$              ;;SIMULATE CONTROL-U
4180 .DSABL  LSB
4181
4182
4183 .....
4184 *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4185 *CALL:
4186 * RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
4187 * RETURN HERE   ;;CHARACTER IS ON THE STACK
4188 *              ;;WITH PARITY BIT STRIPPED OFF
4189 *
4190
4191 016754 011646      $RDCHR: MOV    (SP),-(SP)  ;;PUSH DOWN THE PC
4192 016756 016666 000004 000002 MOV    4(SP),2(SP)      ;;SAVE THE PS
4193 016764 105777 162054      1$: TSTB   @TKS         ;;WAIT FOR
4194 016770 100375              BPL    1$              ;;A CHARACTER
4195 016772 117766 162050 000004 MOVB   @TKB,4(SP)      ;;READ THE TTY
4196 017000 042766 177600 000004 BIC    #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
4197 017006 026627 000004 000023 CMP    4(SP),#23       ;;IS IT A CONTROL-S?
4198 017014 001013              BNE    3$              ;;BRANCH IF NO
4199 017016 105777 162022      2$: TSTB   @TKS         ;;WAIT FOR A CHARACTER
4200 017022 100375              BPL    2$              ;;LOOP UNTIL ITS THERE
4201 017024 117746 162016      MOVB   @TKB,-(SP)      ;;GET CHARACTER
4202 017030 042716 177600      BIC    #^C177,(SP)    ;;MAKE IT 7-BIT ASCII
4203 017034 022627 000021      CMP    (SP)+,#21      ;;IS IT A CONTROL-Q?
4204 017040 001366              BNE    2$              ;;IF NOT DISCARD IT
4205 017042 000750              BR     1$              ;;YES, RESUME
4206 017044 026627 000004 000140 3$: CMP    4(SP),#140     ;;IS IT UPPER CASE?
4207 017052 002407              BLT    4$              ;;BRANCH IF YES
4208 017054 026627 000004 000175      CMP    4(SP),#175     ;;IS IT A SPECIAL CHAR?
4209 017062 003003              BGT    4$              ;;BRANCH IF YES
4210 017064 042766 000040 000004      BIC    #40,4(SP)      ;;MAKE IT UPPER CASE
4211 017072 000002      4$: RTI                ;;GO BACK TO USER
4212 .....
4213 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4214 *CALL:
4215 * RDLIN         ;;INPUT A STRING FROM THE TTY
4216 * RETURN HERE  ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4217 *              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4218 *
4219 017074 010346      $RDLIN: MOV    R3,-(SP) ;;SAVE R3

```

```

4220 017076 012703 017202      1$:  MOV  #STTYIN,R3      ;;GET ADDRESS
4221 017102 022703 017212      2$:  CMP  #STTYIN+8.,R3  ;;BUFFER FULL?
4222 017106 101405              BLOS  4$              ;;BR IF YES
4223 017110 104407              RDCHR              ;;GO READ ONE CHARACTER FROM THE TTY
4224 017112 112613              MOVB  (SP)+,(R3)     ;;GET CHARACTER
4225 017114 122713 000177      10$: CMPB  #177,(R3)     ;;IS IT A RUBOUT
4226 017120 001003              BNE   3$             ;;SKIP IF NOT
4227 017122 104401 001062      4$:  TYPE ,SQUES      ;;TYPE A '?'
4228 017126 000763              BR    1$             ;;CLEAR THE BUFFER AND LOOP
4229 017130 111337 017200      3$:  MOVB  (R3),9$      ;;ECHO THE CHARACTER
4230 017134 104401 017200      TYPE  ,9$
4231 017140 122723 000015      CMPB  #15,(R3)+     ;;CHECK FOR RETURN
4232 017144 001356              BNE   2$             ;;LOOP IF NOT RETURN
4233 017146 105063 177777      CLRB  -1(R3)        ;;CLEAR RETURN (THE 15)
4234 017152 104401 001064      TYPE  ,LF           ;;TYPE A LINE FEED
4235 017156 012603              MOV  (SP)+,R3       ;;RESTORE R3
4236 017160 011646              MOV  (SP),-(SP)     ;;ADJUST THE STACK AND PUT ADDRESS OF THE
4237 017162 016666 000004 000002  MOV  4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
4238 017170 012766 017202 000004  MOV  #STTYIN,4(SP)
4239 017176 000002              RTI                 ;;RETURN
4240 017200      000          9$:  .BYTE  0           ;;STORAGE FOR ASCII CHAR. TO TYPE
4241 017201      000          .BYTE  0           ;;TERMINATOR
4242 017202 000010      $TTYIN: .BLKB 8.    ;;RESERVE 8 BYTES FOR TTY INPUT
4243 017212 052536 005015      $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
4244 017217      136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
4245 017224 005015 053523 020122 $MSWR:  .ASCIZ <15><12>/SWR - /
4246 017232 020075      000
4247 017235      040 047040 053505 $MNEW:  .ASCIZ / NEW - /
4248 017242 036440 000040
4249

```


4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293

017246 010046
017250 016600 000002
017254 005740
017256 111000
017260 006300
017262 016000 017302
017266 000200

017270 011646
017272 016666 000004 000002
017300 000002

017302 017270
017304 015712
017306 016270
017310 016244
017312 016304

017314 016542
017316 016472
017320 016754
017322 017074

.SBTTL TRAP DECODER

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

```
$TRAP: MOV    RO,-(SP)      ;;SAVE RO
        MOV    2(SP),RO   ;;GET TRAP ADDRESS
        TST   -(RO)      ;;BACKUP BY 2
        MOVB  (RO),RO     ;;GET RIGHT BYTE OF TRAP
        ASL   RO          ;;POSITION FOR INDEXING
        MOV   $TRPAD(RO),RO ;;INDEX TO TABLE
        RTS   RO          ;;GO TO ROUTINE
```

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

```
$TRAP2: MOV   (SP),-(SP)  ;;MOVE THE PC DOWN
        MOV   4(SP),2(SP) ;;MOVE THE PSW DOWN
        RTI   ;;RESTORE THE PSW
```

.SBTTL TRAP TABLE

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

ROUTINE	STARTING ADDRESS

\$TRPAD: .WORD	\$TRAP2
\$TYPE	::CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
\$TYPOC	::CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS	::CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON	::CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$GTSWR	::CALL=GTSWR TRAP+5(104405) GET SOFT-SWR SETTING
\$CKSWR	::CALL=CKSWR TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
\$RDCHR	::CALL=RDCHR TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
\$RDLIN	::CALL=RDLIN TRAP+10(104410) TTY TYPEIN STRING ROUTINE

```

4294
4295
4296
4297
4298
4299
4300
4301 017324
4302 017324 000005
4303 017326 112777 000052 163326
4304 017334 105777 163314
4305 017340 100375
4306 017342 117777 163310 163312
4307 017350 017700 163302
4308 017354 100023
4309 017356 052701 010000
4310 017362 030100
4311 017364 001403
4312 017366 004737 017450
4313 017372 017476
4314 017374 006301
4315 017376 030100
4316 017400 001403
4317 017402 004737 017450
4318 017406 017507
4319 017410 006301
4320 017412 030100
4321 017414 001403
4322 017416 004737 017450
4323 017422 017521
4324 017424 042700 000200
4325 017430 022700 000003
4326 017434 001337
4327 017436 004737 017450
4328 017442 017534
4329 017444 000000
4330 017446 000726
4331
4332
4333 017450 013600
4334 017452 062746 000002
4335 017456 105777 163176
4336 017462 100375
4337 017464 112077 163172
4338 017470 105710
4339 017472 001371
4340 017474 000207
4341
4342 017476 005015 040520 044522
4343 017504 054524 000
4344 017507 015 043012 040522
4345 017514 044515 043516 000
4346 017521 015 047412 042526
4347 017526 043122 047514 000127
4348 017534 005015 052123 050117
4349 017542 000

```

```

.SBTTL ECHO TEST
*****
; *THIS ROUTINE WILL ECHO ANY CHARACTER TYPED
; *AT THE CONSOLE
; *THE TEST IS HALTED BY TYPING A CONTROL-C
; *TEST CAN BE RESTARTED AFTER HALTING BY JUST CONTINUING
*****
ECHO:
RESET ;CLEAR EVERYTHING
MOVW #*,@CTBUF ;TRANSMIT PROMPT '*'
TSTB @CRCSR ;WAIT FOR INPUT
BPL 2$
MOVW @CRBUF,@CTBUF ;ECHO INPUT
MOV @CRBUF,R0 ;STORE INPUT
BPL 5$ ;BR IF 'ERROR' NOT SET
BIS #BIT12,R1 ;SET PARITY ERROR TEST MASK
BIT R1,R0 ;CHECK FOR PARITY ERROR FLAG
BEQ 3$ ;BR IF NOT SET
JSR PC,MSG ;REPORT PARITY ERROR
MPAR
3$: ASL R1 ;SHIFT MASK TO TEST 'FR' FLAG
BIT R1,R0 ;TEST FOR FRAMING ERROR FLAG
BEQ 4$ ;BR IF NOT SET
JSR PC,MSG ;REPORT FRAMING ERROR
MFR
4$: ASL R1 ;SHIFT MASK TO TEST 'OR' FLAG
BIT R1,R0 ;TEST FOR OVERFLOW ERROR
BEQ 5$ ;BR IF NOT SET
JSR PC,MSG ;REPORT OVERFLOW ERROR
MOR
5$: BIC #BIT7,R0 ;CLEAR ANY PARITY BIT
CMP #3,R0 ;WAS INPUT CONTROL-C
BNE 2$ ;BR IF NOT
JSR PC,MSG ;REPORT PROGRAM STOP
MSTOP
HALT ;END OF TEST HALT
BR ECHO ;AFTER END OF TEST HALT
; PRESS CONTINUE TO RESTART ECHO TEST

MSG: MOV @(SP)+,R0 ;PICK UP MESSAGE POINTER
ADD #2,-(SP) ;ADJUST RETURN PC
WAIT: TSTB @CTCSR ;WAIT FOR XMIT DONE
BPL WAIT
MOVW (R0)+,@CTBUF ;SEND CHARACTER
TSTB (R0) ;IS THIS END OF MESSAGE?
BNE WAIT ;BR IF NOT
RTS PC ;RETURN

MPAR: .ASCIZ <CR><LF>/PARITY/
MFR: .ASCIZ <CR><LF>/FRAMING/
MOR: .ASCIZ <CR><LF>/OVERFLOW/
MSTOP: .ASCIZ <CR><LF>/STOP/

```

4350
4351
4352 017544
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364 017544 012701 000040
4365 017550 012700 000040
4366 017554 105777 163100
4367 017560 100375
4368 017562 010177 163074
4369 017566 105201
4370 017570 005300
4371 017572 001370
4372 017574 004737 017450
4373
4374 017600 001063
4375 017602 105777 163046
4376 017606 100404
4377 017610 032701 000200
4378 017614 001353
4379 017616 000754
4380
4381 017620 005077 163032
4382 017624 000000
4383 017626 000746

```

.EVEN
.SBTTL  TERMINAL OUTPUT TEST
;*****
; *THIS ROUTINE WILL OUTPUT ALL WRITABLE CHARACTERS FOR THE
; *THE OCTAL CODE 040 --> 377
; *32 CHARACTERS ARE PRINTED ON EACH LINE
; *THE PATTERN IS REPEATED EVERY THREE LINES
; *
;*****
OUTTST: MOV    #40,R1          ;LOAD FIRST WRITABLE CHARACTER
1$:    MOV    #40,R0          ;LOAD CHAR COUNT PER LINE
2$:    TSTB   @CTCSR          ;WAIT FOR DONE
        BPL    2$
        MOV    R1,@CTBUF      ;TRANSMIT A CHARACTER
        INCB   R1             ;INCREMENT CHARACTER CODE
        DEC    R0             ;DECREMENT CHAR COUNT
        BNE    2$             ;BR IF LINE NOT COMPLETE
        JSR    PC,MSG         ;SSUE CR,LINE FEED

        $CRLF
        TSTB   @CRCSR          ;ANY CHARACTER RECEIVED?
        BMI    3$             ;BR IF YES
        BIT    #BIT7,R1       ;FINISHED ONE PASS OF WRITABLE CHARACTERS?
        BNE    OUTTST         ;BR IF YES
        BR     1$             ;IF NOT WRITE NEXT LINE

3$:    CLR    @CRBUF          ;CLEAR RECEIVER
        HALT
        BR     OUTTST         ;RESTART TEST IF CONTINUED

```

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 100
 CZDLDE.P11 10-JUN-80 15:20 TERMINAL OUTPUT TEST

SEQ 0099

4384						
4385						
4386	017630	040503	020116	047516	EM1:	.ASCIZ /CAN NOT ACCESS TCSR/
4387	017636	020124	041501	042503		
4388	017644	051523	052040	051503		
4389	017652	000122				
4390	017654	040503	020116	047516	EM2:	.ASCIZ /CAN NOT ACCESS TBUF/
4391	017662	020124	041501	042503		
4392	017670	051523	052040	052502		
4393	017676	000106				
4394	017700	041524	051123	042040	EM3:	.ASCIZ /TCSR DONE NOT CLEARED WITH TBUF FULL/
4395	017706	047117	020105	047516		
4396	017714	020124	046103	040505		
4397	017722	042522	020104	044527		
4398	017730	044124	052040	052502		
4399	017736	020106	052506	046114		
4400	017744	000				
4401	017745	124	051503	020122	EM4:	.ASCIZ /TCSR DONE NOT SET/
4402	017752	047504	042516	047040		
4403	017760	052117	051440	052105		
4404	017766	000				
4405	017767	124	051503	020122	EM5:	.ASCIZ /TCSR DONE NOT SET WITH RESET/
4406	017774	047504	042516	047040		
4407	020002	052117	051440	052105		
4408	020010	053440	052111	020110		
4409	020016	042522	042523	000124		
4410	020024	040503	020116	047516	EM6:	.ASCIZ /CAN NOT ACCESS RCSR/
4411	020032	020124	041501	042503		
4412	020040	051523	051040	051503		
4413	020046	000122				
4414	020050	040503	020116	047516	EM7:	.ASCIZ /CAN NOT ACCESS RBUF/
4415	020056	020124	041501	042503		
4416	020064	051523	051040	052502		
4417	020072	000106				
4418	020074	040503	020116	047516	EM10:	.ASCIZ /CAN NOT ACCESS LKS/
4419	020102	020124	041501	042503		
4420	020110	051523	046040	051513		
4421	020116	000				
4422	020117	102	052111	020060	EM11:	.ASCIZ /BITO OF TCSR NOT CLEAR AFTER RESET/
4423	020124	043117	052040	051503		
4424	020132	020122	047516	020124		
4425	020140	046103	040505	020122		
4426	020146	043101	042524	020122		
4427	020154	042522	042523	000124		
4428	020162	040503	020116	047516	EM12:	.ASCIZ /CAN NOT SET BITO OF TCSR/
4429	020170	020124	042523	020124		
4430	020176	044502	030124	047440		
4431	020204	020106	041524	051123		
4432	020212	000				
4433	020213	103	047101	047040	EM13:	.ASCIZ /CAN NOT CLEAR BITO OF TCSR/
4434	020220	052117	041440	042514		
4435	020226	051101	041040	052111		
4436	020234	020060	043117	052040		
4437	020242	051503	000122			
4438	020246	042522	042523	020124	EM14:	.ASCIZ /RESET DID NOT CLEAR BITO OF TCSR/
4439	020254	044504	020104	047516		

4440	020262	020124	046103	040505	
4441	020270	020122	044502	030124	
4442	020276	047440	020106	041524	
4443	020304	051123	000		
4444	020307	102	052111	020062	EM15: .ASCIIZ /BIT2 OF TCSR NOT CLEAR AFTER RESET/
4445	020314	043117	052040	051503	
4446	020322	020122	047516	020124	
4447	020330	046103	040505	020122	
4448	020336	043101	042524	020122	
4449	020344	042522	042523	000124	
4450	020352	040503	020116	047516	EM16: .ASCIIZ /CAN NOT SET BIT2 OF TCSR/
4451	020360	020124	042523	020124	
4452	020366	044502	031124	047440	
4453	020374	020106	041524	051123	
4454	020402	000			
4455	020403	103	047101	047040	EM17: .ASCIIZ /CAN NOT CLEAR BIT2 OF TCSR/
4456	020410	052117	041440	042514	
4457	020416	051101	041040	052111	
4458	020424	020062	043117	052040	
4459	020432	051503	000122		
4460	020436	042522	042523	020124	EM20: .ASCIIZ /RESET DID NOT CLEAR BIT2 OF TCSR/
4461	020444	044504	020104	047516	
4462	020452	020124	046103	040505	
4463	020460	020122	044502	031124	
4464	020466	047440	020106	041524	
4465	020474	051123	000		
4466	020477	102	052111	020066	EM21: .ASCIIZ /BIT6 OF TCSR NOT CLEAR AFTER RESET/
4467	020504	043117	052040	051503	
4468	020512	020122	047516	020124	
4469	020520	046103	040505	020122	
4470	020526	043101	042524	020122	
4471	020534	042522	042523	000124	
4472	020542	046530	052111	044440	EM22: .ASCIIZ /XMIT INTERRUPT AT PRIORITY 7/
4473	020550	052116	051105	052522	
4474	020556	052120	040440	020124	
4475	020564	051120	047511	0445??	
4476	020572	054524	033440	000	
4477	020577	103	047101	047040	EM23: .ASCIIZ /CAN NOT SET BIT6 OF TCSR/
4478	020604	052117	051440	052105	
4479	020612	041040	052111	020066	
4480	020620	043117	052040	051503	
4481	020626	000122			
4482	020630	040503	020116	047516	EM24: .ASCIIZ /CAN NOT CLEAR BIT6 OF TCSR/
4483	020636	020124	046103	040505	
4484	020644	020122	044502	033124	
4485	020652	047440	020106	041524	
4486	020660	051123	000		
4487	020663	122	051505	052105	EM25: .ASCIIZ /RESET DID NOT CLEAR BIT6 OF TCSR/
4488	020670	042040	042111	047040	
4489	020676	052117	041440	042514	
4490	020704	051101	041040	052111	
4491	020712	020066	043117	052040	
4492	020720	051503	000122		
4493	020724	044502	033124	047440	EM26: .ASCIIZ /BIT6 OF RCSR NOT CLEAR AFTER RESET/
4494	020732	020106	041522	051123	
4495	020740	047040	052117	041440	

.MAIN. MACY11 3GA(1052) 10-JUN-80 15:21 PAGE 102
 CZDLDE.P11 10-JUN-80 15:20 TERMINAL OUTPUT TEST

SEQ 0101

4496	020746	042514	051101	040440	
4497	020754	052106	051105	051040	
4498	020762	051505	052105	000	
4499	020767	122	053103	020122	EM27: .ASCIZ /RCVR INTERRUPT WITH PRIORITY 7/
4500	020774	047111	042524	051122	
4501	021002	050125	020124	044527	
4502	021010	044124	050040	044522	
4503	021016	051117	052111	020131	
4504	021024	000067			
4505	021026	040503	020116	047516	EM30: .ASCIZ /CAN NOT SET BIT6 OF RCSR/
4506	021034	020124	042523	020124	
4507	021042	044502	033124	047440	
4508	021050	020106	041522	051123	
4509	021056	000			
4510	021057	103	047101	047040	EM31: .ASCIZ /CAN NOT CLEAR BIT6 OF RCSR/
4511	021064	052117	041440	042514	
4512	021072	051101	041040	052111	
4513	021100	020066	043117	051040	
4514	021106	051503	000122		
4515	021112	040503	020116	047516	EM32: .ASCIZ /CAN NOT CLEAR BIT6 OF RCSR WITH RESET/
4516	021120	020124	046103	040505	
4517	021126	020122	044502	033124	
4518	021134	047440	020106	041522	
4519	021142	051123	053440	052111	
4520	021150	020110	042522	042523	
4521	021156	000124			
4522	021160	044502	033124	047440	EM33: .ASCIZ /BIT6 OF LKS NOT CLEAR AFTER RESET/
4523	021166	020106	045514	020123	
4524	021174	047516	020124	046103	
4525	021202	040505	020122	043101	
4526	021210	042524	020122	042522	
4527	021216	042523	000124		
4528	021222	045514	020123	047111	EM34: .ASCIZ /LKS INTERRUPT WITH PRIORITY 7/
4529	021230	042524	051122	050125	
4530	021236	020124	044527	044124	
4531	021244	050040	044522	051117	
4532	021252	052111	020131	000067	
4533	021260	040503	020116	047516	EM35: .ASCIZ /CAN NOT SET BIT6 OF LKS/
4534	021266	020124	042523	020124	
4535	021274	044502	033124	047440	
4536	021302	020106	045514	000123	
4537	021310	040503	020116	047516	EM36: .ASCIZ /CAN NOT CLEAR BIT6 OF LKS/
4538	021316	020124	046103	040505	
4539	021324	020122	044502	033124	
4540	021332	047440	020106	045514	
4541	021340	000123			
4542	021342	042522	042523	020124	EM37: .ASCIZ /RESET DID NOT CLEAR BIT6 OF LKS/
4543	021350	044504	020104	047516	
4544	021356	020124	046103	040505	
4545	021364	020122	044502	033124	
4546	021372	047440	020106	045514	
4547	021400	000123			
4548	021402	052504	046101	040440	EM40: .ASCIZ /DUAL ADDRESSING ERROR/
4549	021410	042104	042522	051523	
4550	021416	047111	020107	051105	
4551	021424	047522	000122		

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 103
 CZDLDE.P11 10-JUN-80 15:20 TERMINAL OUTPUT TEST

SEQ 0102

4552	021430	044502	033124	047440	EM41:	.ASCIZ	/BIT6 OF LKS NOT SET AFTER RESET/
4553	021436	020106	045514	020123			
4554	021444	047516	020124	042523			
4555	021452	020124	043101	042524			
4556	021460	020122	042522	042523			
4557	021466	000124					
4558	021470	040503	020116	047516	EM42:	.ASCIZ	/CAN NOT CLEAR BIT7 OF LKS/
4559	021476	020124	046103	040505			
4560	021504	020122	044502	033524			
4561	021512	047440	020106	045514			
4562	021520	000123					
4563	021522	044502	033524	047440	EM43:	.ASCIZ	/BIT7 OF LKS DOES NOT SET/
4564	021530	020106	045514	020123			
4565	021536	047504	051505	047040			
4566	021544	052117	051440	052105			
4567	021552	000					
4568	021553	122	041524	044440	EM44:	.ASCIZ	/RTC INTERRUPT AT PRIORITY 7/
4569	021560	052116	051105	052522			
4570	021566	052120	040440	020124			
4571	021574	051120	047511	044522			
4572	021602	054524	033440	000			
4573	021607	122	041524	044440	EM45:	.ASCIZ	/RTC INTERRUPTS WHEN DISABLED/
4574	021614	052116	051105	052522			
4575	021622	052120	020123	044127			
4576	021630	047105	042040	051511			
4577	021636	041101	042514	000104			
4578	021644				EM47:		
4579	021644	052122	020103	047111	EM46:	.ASCIZ	/RTC INTERRUPT DID NOT OCCUR/
4580	021652	042524	051122	050125			
4581	021660	020124	044504	020104			
4582	021666	047516	020124	041517			
4583	021674	052503	000122				
4584	021700	052122	020103	047504	EM50:	.ASCIZ	/RTC DOUBLE INTERRUPT/
4585	021706	041125	042514	044440			
4586	021714	052116	051105	052522			
4587	021722	052120	000				
4588	021725	122	051505	052105	EM51:	.ASCIZ	/RESET DID NOT INTERRUPT/
4589	021732	042040	042111	047040			
4590	021740	052117	044440	052116			
4591	021746	051105	052522	052120			
4592	021754	000					
4593	021755	122	041524	044440	EM52:	.ASCIZ	/RTC INTERRUPT DID NOT CLEAR WITH BIT7 OF LKS/
4594	021762	052116	051105	052522			
4595	021770	052120	042040	042111			
4596	021776	047040	052117	041440			
4597	022004	042514	051101	053440			
4598	022012	052111	020110	044502			
4599	022020	033524	047440	020106			
4600	022026	045514	000123				
4601	022032	046103	041517	020113	EM53:	.ASCIZ	/CLOCK REPEATABILITY/
4602	022040	042522	042520	052101			
4603	022046	041101	046111	052111			
4604	022054	000131					
4605	022056	046530	052111	044440	EM54:	.ASCIZ	/XMIT INTERRUPTS WHEN DISABLED/
4606	022064	052116	051105	052522			
4607	022072	052120	020123	044127			

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 104
 CZDLDE.P11 10-JUN-80 15:20 TERMINAL OUTPUT TEST

SEQ 0103

4608	022100	047105	042040	051511	
4609	022106	041101	042514	000104	
4610	022114	046530	052111	044440	EM56: .ASCIZ /XMIT INTERRUPTS AT PRIORITY 7/
4611	022122	052116	051105	052522	
4612	022130	052120	020123	052101	
4613	022136	050040	044522	051117	
4614	022144	052111	020131	000067	
4615	022152	046530	052111	044440	EM57: .ASCIZ /XMIT INTERRUPTS WITH ENABLE CLEAR/
4616	022160	052116	051105	052522	
4617	022166	052120	020123	044527	
4618	022174	044124	042440	040516	
4619	022202	046102	020105	046103	
4620	022210	040505	000122		
4621	022214				EM55:
4622	022214	046530	052111	042040	EM60: .ASCIZ /XMIT DID NOT INTERRUPT/
4623	022222	042111	047040	052117	
4624	022230	044440	052116	051105	
4625	022236	052522	052120	000	
4626	022243	130	044515	020124	EM61: .ASCIZ /XMIT RE-INTERRUPTED/
4627	022250	042522	044455	052116	
4628	022256	051105	052522	052120	
4629	022264	042105	000		
4630	022267	114	040517	044504	EM62: .ASCIZ /LOADING TBUF DID NOT CLEAR INTERRUPT/
4631	022274	043516	052040	052502	
4632	022302	020106	044504	020104	
4633	022310	047516	020124	046103	
4634	022316	040505	020122	047111	
4635	022324	042524	051122	050125	
4636	022332	000124			
4637	022334	041522	051126	040440	EM63: .ASCIZ /RCVR ACTIVE NOT SET/
4638	022342	052103	053111	020105	
4639	022350	047516	020124	042523	
4640	022356	000124			
4641	022360	041522	051126	042040	EM64: .ASCIZ /RCVR DONE NEVER SET/
4642	022366	047117	020105	042516	
4643	022374	042526	020122	042523	
4644	022402	000124			
4645	022404	041522	051126	040440	EM65: .ASCIZ /RCVR ACTIVE NOT CLEARED WITH DONE SET/
4646	022412	052103	053111	020105	
4647	022420	047516	020124	046103	
4648	022426	040505	042522	020104	
4649	022434	044527	044124	042040	
4650	022442	047117	020105	042523	
4651	022450	000124			
4652	022452	042522	042523	020124	EM66: .ASCIZ /RESET DID NOT CLEAR RCVR DONE/
4653	022460	044504	020104	047516	
4654	022466	020124	046103	040505	
4655	022474	020122	041522	051126	
4656	022502	042040	047117	000105	
4657	022510	042122	020122	047105	EM67: .ASCIZ /RDR ENABLE DID NOT CLEAR RCVR DONE/
4658	022516	041101	042514	042040	
4659	022524	042111	047040	052117	
4660	022532	041440	042514	051101	
4661	022540	051040	053103	020122	
4662	022546	047504	042516	000	
4663	022553	122	040505	044504	EM70: .ASCIZ /READING RBUF DID NOT CLEAR RCVR DONE/

4664	022560	043516	051040	052502	
4665	022566	020106	044504	020104	
4666	022574	047516	020124	046103	
4667	022602	040505	020122	041522	
4668	022610	051126	042040	047117	
4669	022616	000105			
4670	022620	041522	051126	044440	EM71: .ASCIZ /RCVR INTERRUPTS WITH ENABLE CLEAR/
4671	022626	052116	051105	052522	
4672	022634	052120	020123	044527	
4673	022642	044124	042440	040516	
4674	022650	046102	020105	046103	
4675	022656	040505	000122		
4676	022662	041522	051126	044440	EM73: .ASCIZ /RCVR INTERRUPTS AT PRIORITY 7/
4677	022670	052116	051105	052522	
4678	022676	052120	020123	052101	
4679	022704	050040	044522	051117	
4680	022712	052111	020131	000067	
4681	022720	041522	051126	044440	EM74: .ASCIZ /RCVR INT RQST PASSED WITH ENABLE CLEAR/
4682	022726	052116	051040	051521	
4683	022734	020124	040520	051523	
4684	022742	042105	053440	052111	
4685	022750	020110	047105	041101	
4686	022756	042514	041440	042514	
4687	022764	051101	000		
4688	022767				EM72:
4689	022767	122	053103	020122	EM75: .ASCIZ /RCVR DID NOT INTERRUPT/
4690	022774	044504	020104	047516	
4691	023002	020124	047111	042524	
4692	023010	051122	050125	000124	
4693	023016	041522	051126	051040	EM76: .ASCIZ /RCVR RE-INTERRUPTED/
4694	023024	026505	047111	042524	
4695	023032	051122	050125	042524	
4696	023040	000104			
4697	023042	042522	042101	047111	EM77: .ASCIZ /READING RBUF DID NOT CLEAR INTERRUPT/
4698	023050	020107	041122	043125	
4699	023056	042040	042111	047040	
4700	023064	052117	041440	042514	
4701	023072	051101	044440	052116	
4702	023100	051105	052522	052120	
4703	023106	000			
4704	023107	122	051505	052105	EM100: .ASCIZ /RESET DID NOT CLEAR RCVR INTERRUPT/
4705	023114	042040	042111	047040	
4706	023122	052117	041440	042514	
4707	023130	051101	051040	053103	
4708	023136	020122	047111	042524	
4709	023144	051122	050125	000124	
4710	023152	047447	023522	043040	EM101: .ASCIZ /'OR' FLAG DID NOT SET/
4711	023160	040514	020107	044504	
4712	023166	020104	047516	020124	
4713	023174	042523	000124		
4714	023200	042447	051122	051117	EM102: .ASCIZ /'ERROR' NOT SET WITH 'OR' FLAG/
4715	023206	020047	047516	020124	
4716	023214	042523	020124	044527	
4717	023222	044124	023440	051117	
4718	023230	020047	046106	043501	
4719	023236	000			

4720	023237	107	042522	045501	EM103: .ASCIZ /BREAK DID NOT XMIT ALL ZEROES/
4721	023244	042040	042111	047040	
4722	023252	052117	054040	044515	
4723	023260	020124	046101	020114	
4724	023266	042532	047522	051505	
4725	023274	000			
4726	023275	102	042522	045501	EM 04: .ASCIZ /BREAK DID NOT SET 'FR' ERROR/
4727	023302	042040	042111	047040	
4728	023310	052117	051440	052105	
4729	023316	023440	051106	020047	
4730	023324	051105	047522	000122	
4731	023332	040504	040524	041440	EM105: .ASCIZ /DATA COMPARE ERROR/
4732	023340	046517	040520	042522	
4733	023346	042440	051122	051117	
4734	023354	000			
4735	023355	104	052101	020101	FM106: .ASCIZ /DATA COMPARE ERROR WITH CABLE/
4736	023362	047503	050115	051101	
4737	023370	020105	051105	047522	
4738	023376	020122	044527	044124	
4739	023404	041440	041101	042514	
4740	023412	000			
4741	023413	124	046511	047505	EM107: .ASCIZ /TIMEOUT IN EXERCISER TEST/
4742	023420	052125	044440	020116	
4743	023426	054105	051105	044503	
4744	023434	042523	020122	042524	
4745	023442	052123	000		
4746	023445	111	041516	051117	EM110: .ASCIZ /INCORRECT RECEIVE COUNT/
4747	023452	042522	052103	051040	
4748	023460	041505	044505	042526	
4749	023466	041440	052517	052116	
4750	023474	000			
4751	023475	104	052101	020101	EM111: .ASCIZ /DATA COMPARE ERROR IN EXERCISER/
4752	023502	047503	050115	051101	
4753	023510	020105	051105	047522	
4754	023516	020122	047111	042440	
4755	023524	042530	041522	051511	
4756	023532	051105	000		
4757	023535	124	040522	020120	EM112: .ASCIZ /TRAP CATCHER/
4758	023542	040503	041524	042510	
4759	023550	000122			
4760	023552	047516	041440	045514	EM113: .ASCIZ /NO CLK INTERRUPT IN EXERCISER/
4761	023560	044440	052116	051105	
4762	023566	052522	052120	044440	
4763	023574	020116	054105	051105	
4764	023602	044503	042523	000122	
4765	023610	042442	051122	051117	EM114: .ASCIZ /'FROR' NOT SET WITH 'FR' FLAG/
4766	023616	020042	047516	020124	
4767	023624	042523	020124	044527	
4768	023632	044124	021040	051106	
4769	023640	020042	046106	043501	
4770	023646	000			
4771	023647	122	053103	040440	EM115: .ASCIZ /RCV ACTIVE NOT CLEAR WITH INIT/
4772	023654	052103	053111	020105	
4773	023662	047516	020124	046103	
4774	023670	040505	020122	044527	
4775	023676	044124	044440	044516	

4832 024350 047507 042117 020040
4833 024356 020040 040502 000104
4834 024364 042524 052123 020043
4835 024372 020040 051105 020122
4836 024400 041520 020040 041522
4837 024406 051123 020040 020040
4838 024414 051124 047101 020123
4839 024422 020040 041522 000126
4840 024430 042524 052123 020043
4841 024436 020040 051105 020122
4842 024444 041520 020040 041522
4843 024452 051123 020040 020040
4844 024460 046117 050104 020103
4845 024466 020040 051124 050101
4846 024474 040440 051104 000
4847 024502 005015 055103 046104
4848 024510 042504 020060 046104
4849 024516 030461 053455 030454
4850 024524 032061 020064 043115
4851 024532 020115 046123 000125
4852 024540 005015 042040 053105
4853 024542 020040 051505 052440
4854 024550 041511 051105 052040
4855 024556 042116 020124 000040
4856 024564 051505 001072 001016 002640
4857 024572 001072 001016 002642
4858 024600 000000 001072 001016 002634
4859 024602 001072 001016 002636
4860 024610 000000 001072 001016 002674
4861 024612 001072 001016 001020
4862 024620 000000 001022 000000
4863 024622 001072 001016 002674
4864 024624 002336 002616 000000
4865 024626 001072 001016 002634
4866 024628 001026 000000 002634
4867 024630 001072 001016 000000
4868 024632 001072 001026 000000
4869 024634 001072 001016 002634
4870 024636 001024 001026 000000
4871 024638 001072 001016 002634
4872 024640 002442 002440 000000
4873 024642 001072 001016 002634
4874 024644 002620 002622 000000
4875 024646 000000
4876 000001
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886

DH110: .ASCIZ /TEST# ERR PC RCSR TRANS RCV/
DH112: .ASCIZ /TEST# ERR PC RCSR OLDPC TRAP ADR/
M1: .EVEN
.ASCIZ <CR><LF>/CZDLDE0 DL11-W,1144 MFM SLU/
M2: .EVEN
M2A: .ASCII <CR><LF>
.ASCIZ / DEVICES UNDER TEST /
DT1: .EVEN
.WORD \$TESTN,\$ERRPC,TCSR,0
DT2: .WORD \$TESTN,\$ERRPC,TBUF,0
DT6: .WORD \$TESTN,\$ERRPC,RCSR,0
DT7: .WORD \$TESTN,\$ERRPC,RBUF,0
DT10: .WORD \$TESTN,\$ERRPC,LKS,0
DT40: .WORD \$TESTN,\$ERRPC,\$GDADR,\$BDADR,0
DT53: .WORD \$TESTN,\$ERRPC,LKS,FIRST,SECND,0
DT103: .WORD \$TESTN,\$ERRPC,RCSR,\$BDDAT,0
DT105: .WORD \$TESTN,\$ERRPC,RCSR,\$GDDAT,\$BDDAT,0
DT110: .WORD \$TESTN,\$ERRPC,RCSR,XMTCNT,RCVCNT,0
DT112: .WORD \$TESTN,\$ERRPC,RCSR,OLDPC,BDVECT,0
ENDADR: .WORD 0 ;END ADDRESS FOR CHECKSUM AND SAVE AREA
;OF WRAP AROUND CONSOLE TEST
.END

ABASE = 176500	911#	1024	1065	
ACDW1 = 000000	1024			
ACDW2 = 000000	1024			
ACPUOP = 000000	1024	1039		
ADDW0 = 000000	1024			
ADDW1 = 000000	1024			
ADDW10 = 000000	1024			
ADDW11 = 000000	1024			
ADDW12 = 000000	1024			
ADDW13 = 000000	1024			
ADDW14 = 000000	1024			
ADDW15 = 000000	1024			
ADDW2 = 000000	1024			
ADDW3 = 000000	1024			
ADDW4 = 000000	1024			
ADDW5 = 000000	1024			
ADDW6 = 000000	1024			
ADDW7 = 000000	1024			
ADDW8 = 000000	1024			
ADDW9 = 000000	1024			
ADEVCT = 000000	1024	1030		
ADEVM = 000000	1024	1066		
ADR 004132	1738#	1741		
ADRTBL 002702	1543#	1554	1736	
AENV = 000000	1024	1035		
AENVM = 000000	1024	1036		
AFATAL = 000000	1024	1027		
AMADR1 = 000000	1024	1052		
AMADR2 = 000000	1024	1056		
AMADR3 = 000000	1024	1059		
AMADR4 = 000000	1024	1062		
AMAMS1 = 000000	1024	1046		
AMAMS2 = 000000	1024	1054		
AMAMS3 = 000000	1024	1057		
AMAMS4 = 000000	1024	1060		
AMSGAD = 000000	1024	1032		
AMSGLG = 000000	1024	1033		
AMSGTY = 000000	1024	1026		
AMTYP1 = 000000	1024	1047		
AMTYP2 = 000000	1024	1055		
AMTYP3 = 000000	1024	1058		
AMTYP4 = 000000	1024	1061		
APASS = 000000	1024	1029		
APRIOR = 000000	1024			
APTCSU = 000040	3937#	3970		
APTENV = 000001	3713	3892	3935#	3963
APTSIZ = 000200	1617	3934#		
APTSP0 = 000100	3894	3936#	3965	
APTSZD 003636	1641	1643	1669#	
ASWREG = 000000	1024	1037		
ATESTN = 000000	1024	1028		
AUNIT = 000000	1024	1031		
AUSWR = 000400	913#	1024	1038	
AVECT1 = 000300	912#	1024	1063	
AVECT2 = 000000	1024	1064		
BDVECT 002622	1506#	3681*	4881	

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 113
 CZDLDE.P11 10-JUN-80 15:20

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0111

EM104	023275	1420	4726#
EM105	023332	1425	4731#
EM106	023355	1430	4735#
EM107	023413	1435	4741#
EM11	020117	1126	4422#
EM110	023445	1440	4746#
EM111	023475	1445	4751#
EM112	023535	1450	4757#
EM113	023552	1455	4760#
EM114	023610	1460	4765#
EM115	023647	1465	4771#
EM116	023706	1470	4777#
EM117	023745	1475	4783#
EM12	020162	1131	4428#
EM13	020213	1136	4433#
EM14	020246	1141	4438#
EM15	020307	1146	4444#
EM16	020352	1151	4450#
EM17	020403	1156	4455#
EM2	017654	1091	4390#
EM20	020436	1161	4460#
EM21	020477	1166	4466#
EM22	020542	1171	4472#
EM23	020577	1176	4477#
EM24	020630	1181	4482#
EM25	020663	1186	4487#
EM26	020724	1191	4493#
EM27	020767	1196	4499#
EM3	017700	1096	4394#
EM30	021026	1201	4505#
EM31	021057	1206	4510#
EM32	021112	1211	4515#
EM33	021160	1216	4522#
EM34	021222	1221	4528#
EM35	021260	1226	4533#
EM36	021310	1231	4537#
EM37	021342	1236	4542#
EM4	017745	1101	4401#
EM40	021402	1241	4548#
EM41	021430	1246	4552#
EM42	021470	1251	4558#
EM43	021522	1256	4563#
EM44	021553	1261	4568#
EM45	021607	1266	4573#
EM46	021644	1271	4579#
EM47	021644	1276	4578#
EM5	017767	1106	4405#
EM50	021700	1281	4584#
EM51	021725	1286	4588#
EM52	021755	1291	4593#
EM53	022032	1296	4601#
EM54	022056	1301	4605#
EM55	022214	1306	4621#
EM56	022114	1311	4610#
EM57	022152	1316	4615#
EM6	020024	1111	4410#

TKVEC = 000060	907#																			
TMP1 002626	1513#	1712*	1728*	1734	1735*															
TMP2 002630	1514#	1646*	1654*	1664*	1669*	1673*	1693	3621	3664											
TMP3 002632	1515#	1711*	1725	1727*	1733*															
TOLER 007256	2434	2436#																		
TPSW 002652	1525#	1720	2526	2528*	2556*	3298	3302*	3387*												
TPVEC = 000064	908#																			
TRAPVE= 000034	906#	1591*	1592*																	
TRTVEC= 000014	901#																			
TSTDEV 004050	1715	1725#	1730	3625																
TSTDVM 003646	1671#	1676																		
TST1 004172	1722	1748	1756#	3667																
TST10 005044	1969#																			
TST11 005252	2022#																			
TST12 005414	2064#																			
TST13 005544	2108#																			
TST14 005614	2110	2112	2126#																	
TST15 005764	2128	2130	2171#																	
TST16 006154	2211#																			
TST17 006262	2213	2215	2233	2243#																
TST2 004244	1778#																			
TST20 006504	2245	2247	2299#																	
TST21 006660	2301	2303	2341#																	
TST22 007002	2343	2345	2373#																	
TST23 007136	2375	2377	2408#																	
TST24 007324	2410	2412	2448	2456#																
TST25 007432	2486#																			
TST26 007560	2523#																			
TST27 007722	2561#																			
TST3 004316	1799#																			
TST30 010066	2601#																			
TST31 010402	2698#																			
TST32 010504	2729#																			
TST33 010640	2770#																			
TST34 011050	2825#																			
TST35 011262	2889#																			
TST36 011476	2945#																			
TST37 011652	2991#																			
TST4 004466	1846#																			
TST40 012020	3034#																			
TST41 012214	3036	3044	3090#																	
TST42 012440	3092	3100	3146#																	
TST43 012632	3148	3150	3158	3197#																
TST44 013010	3251#																			
TST45 013132	3253	3280	3291#																	
TST46 013710	3426#																			
TST5 004714	1919#																			
TST6 004760	1938#																			
TST7 005012	1952#																			
TURBUF 003006	1547#																			
TURCSR 003004	1546#																			
TUTBUF 003012	1549#																			
TUTCSR 003010	1548#																			
TVECT 002650	1524#	2024	2025*	2058*	2458	2459*	2469*	2480*	2490	2491*	2505*	2517*	2525							
	2527*	2542*	2555*	2565	2566*	2588*	3295	3301*	3384*											
TYPE = 104401	1899	1901	3647	3711	3746	3763	3765	3768	3770	3774	3781	3823	3983							

\$RDOCT= ***** U	4293													
\$RDSZ = 000010	4212#													
\$RTNAD 014462	3657#													
\$R2A = ***** U	4293													
\$SAVRE= ***** U	4293													
\$SAVR6 015260	3800*	3810	3820*	3821*	3828#									
\$SCOPE 015272	1587	3845#												
\$SETUP= 000137	934#	1586	1587	1589	1591	1593	1595	1596	1598	3638	3846	4115	4249	
\$STUP = 177777	934#													
\$SVLAD 015374	3956	3862	3868#											
\$SVPC = 000500	950#	955												
\$SWR = 161000	915#	1016	1017	1596	1598	1599	1757	1779	1800	1847	1920	1939	1953	
	1970	2023	2065	2109	2127	2172	2212	2244	2300	2342	2374	2409	2457	
	2487	2524	2562	2602	2699	2730	2771	2826	2890	2946	2992	3035	3091	
	3147	3198	3252	3292	3427	3633	3639	3650	3656	3658	3839	3840	3841	
	3842	3847	3859	3861	3862	3863	3868	3871	3874	3877				
\$SWREG 001110	1037#	1619												
\$SWRMK= 000000	3842													
\$TESTN 001072	1028#	1568*	3869*	4861	4863	4865	4867	4869	4871	4873	4875	4877	4879	
	488*													
\$TAB 001046	1009#	4011	4017	4113	4124	4141	4195	4201						
\$TKS 001044	1008#	4009	4015	4113	4122	4138	4162*	4193	4199					
\$TN = 000047	914#	1753	1757#	1775	1779#	1796	1800#	1843	1847#	1916	1920#	1935	1939#	
	1949	1953#	1966	1970#	2019	2023#	2061	2065#	2104	2105	2109#	2123	2127#	
	2168	2172#	2208	2212#	2240	2244#	2296	2300#	2338	2342#	2370	2374#	2405	
	2409#	2453	2457#	2483	2487#	2520	2524#	2558	2562#	2598	2602#	2695	2699#	
	2726	2730#	2767	2771#	2822	2826#	2886	2890#	2942	2946#	2988	2992#	3031	
	3035#	3087	3091#	3143	3147#	3194	3198#	3248	3252#	3288	3292#	3423	3427#	
\$TPB 001052	1011#	4008*	4031											
\$TPFLG 001057	1015#	3957	4031											
\$TPS 001050	1010#	2844	4006	4031										
\$TRAP 017246	1591	4260#												
\$TRAP2 017270	4271#	4287												
\$TRP = 000011	4275#	4284#	4285#	4286#	4287#	4288	4289#	4290	4291#	4292#	4293#			
\$TRPAD 017302	4265	4282#												
\$TSTM 000504	974#													
\$TSTM 001002	988#	2104*	3619*	3638*	3703	3838	3868*	3869	3874	3877				
\$TTYIN 017202	4220	4221	4238	4242#										
\$TYPBN= ***** U	4287													
\$TYPDS= ***** U	4287													
\$TYPE 015712	3911	3957#	4275	4283										
\$TYPEC 016124	3987	3994	4001	4006#	4007	4164								
\$TYPEX 016242	4024	4026	4029#											
\$TYPOC 016270	4061#	4284												
\$TYPON 016304	4060	4063#	4286											
\$TYPOS 016244	4056#	4285												
\$UNIT 001100	1031#	1571*	1729*	1732*	3669*									
\$UNITM 000510	976#													
\$USWR 001112	1038#	1573	1575*	3228	3276									
\$VECT1 001136	1063#	1682												
\$VECT2 001140	1064#													
\$XTSTR 015304	3850#													
\$BGET4= 000000	3650#													
\$OFILL 016467	4057*	4061*	4071	4106#										
\$LOCAT= ***** U	3847													
. = 024750	919#	925	926#	930#	936#	940#	945#	950	951#	953#	955#	961	962#	

.MAIN. MACY11 30A(1052) 10-JUN-80 15:21 PAGE 121
CZSLDE.P11 10-JUN-80 15:20

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0119

964#	966#	985#	1020	1485#	1486#	1496#	1543#	1544#	1584	1598	1599	2616
2711	2745	2784	2842	2908	2963	3009	3057	3113	3171	3214	3323	3658
3659#	3784#	3803	3827	3877	3933#	4031	4113	4242#	4243	4249	4352#	4847#
1085#												
961#	966											

.SERRT 001146
.SX = 000500

.KT11	1#		
.SETUP	1#	800#	934
.SWRHI	1#		
.SAC11	1#	800#	946
.SAPT8	1#	800#	1021#
.SAPTH	1#	800#	956
.SAPTY	1#		
.SASTA	1#		
.SCATC	1#		
.SCMTA	1#	800#	979
.SDB2D	1#		
.SDB20	1#		
.SDIV	1#		
.SEOP	1#	800#	3629
.SERRO	1#		
.SERRT	1#	800#	3738
.SMULT	1#		
.SPOWE	1#		
.SRAND	1#		
.SRDDE	1#		
.SRDOC	1#		
.SREAD	1#	800#	4110
.SR2AZ	1#		
.SSAVE	1#		
.SSB2D	1#		
.SSB20	1#		
.SSCOP	1#	800#	3833
.SSIZE	1#		
.SSUPR	1#		
.STRAP	1#	800#	4252
.STYPB	1#		
.STYPD	1#		
.STYPE	1#	800#	3940
.STYPO	1#	800#	4031
.S4OCA	1#		
.1170	1#		

. ABS. 024750 000

ERRORS DETECTED: 0

CZDLDE.BIN,CZDLDE.LST/CRF/SOL/NL:TOC CZDLDE.SML,CZDLDE.P11
RUN-TIME: 45 61 4 SECONDS
RUN-TIME RATIO: 231/112-2.0
CORE USED: 33K (65 PAGES)