

PDT-11

110/130 SYS EXER
CVKDBA0

AH-F401A-MC
COPYRIGHT 1979
FICHE 1 OF 1

MAY 1979
digital
MADE IN USA

This microfiche card contains a grid of frames. The frames are arranged in approximately 12 rows and 10 columns. Each frame contains a small, high-contrast image of a document page, likely a technical manual or system exercise. The text within the frames is too small to be legible, but the layout suggests a structured document with multiple pages of content. The frames are separated by thin white lines, and the overall card has a dark background.

.REM 2

IDENTIFICATION

PRODUCT CODE: AC-F400A-MC
PRODUCT NAME: CVKDBAO PDT-11 110/130 SYS EX
PRODUCT DATE: APRIL 1979
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) 1979 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DECX/11

HISTORY

REVISION

DATE

A

APRIL, 1979

TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION.
1.1	PROGRAM PURPOSE (ABSTRACT).
1.2	SYSTEM REQUIREMENTS.
1.3	RELATED DOCUMENTS AND STANDARDS.
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES.
1.5	ASSUMPTIONS.
1.6	RUNNING SYSTEMS PROGRAMS.
2.0	OPERATING INSTRUCTIONS.
2.1	LOADING AND STARTING PROCEDURES.
2.2	SPECIAL ENVIRONMENTS.
2.3	OPERATIONAL SWITCH SETTINGS
2.4	PROGRAM OPTIONS.
2.4.1	DEVICE MAP--TESTING CONTROL
2.5	EXECUTION TIMES.
2.6	POWER FAIL.
3.0	ERROR INFORMATION.
3.1	ERROR REPORTING PROCEDURE.
3.2	ERROR HALTS.
4.0	PROGRESS AND PERFORMANCE REPORTS.
5.0	DEVICE INFORMATION TABLES.
6.0	DEVICE CONNECTOR LAYOUT.
7.0	SUMMARY OF TESTS.

1.0 GENERAL PROGRAM INFORMATION.

1.1 PROGRAM PURPOSE (ABSTRACT).

THE CVKDBAO PDT-11 110/130 SYSTEM EXERCISER TESTS THE PROCESSOR'S ABILITY TO OPERATE ALL ITS PERIPHERALS IN INTERRUPT MODE AT THE SAME TIME WITH EMPHASIS ON THE TU58 TAPE SUBSYSTEMS. IT SHOULD BE NOTED THAT THIS PROGRAM IS NOT A DIAGNOSTIC OF THE PERIPHERALS BUT A SERIES OF SYSTEM INTERACTION TESTS.

THE PROGRAM IS DEFAULTED TO EXERCISE THE CLUSTER TERMINALS AND THE COMMUNICATION (COMM) PORT IN INTERNAL LOOPBACK MODE (MAINTENANCE MODE). THE DEFAULT CAN BE CHANGED TO EXERCISE THE CLUSTER TERMINALS AND/OR THE COMM PORT IN EXTERNAL LOOPBACK (SPECIAL LOOPBACK CONNECTORS REQUIRED). SEE PROGRAM OPTIONS SEC. 2.4.

TESTING OF THE PHYSICAL CLUSTER TERMINALS OR COMM DEVICE WILL NOT BE PERFORMED.

THE PRINTER WILL BE EXERCISED IF SIZING DETERMINES IT TO BE PRESENT OR IF AN EXTERNAL LOOPBACK CONNECTOR IS INSTALLED.

AFTER THE MEMORY HAS BEEN TESTED AND ALL THE PERIPHERALS ARE BEING EXERCISED AND INTERRUPTING, THE TU58 SUBSYSTEM TESTS WILL BEGIN. SEE TEST SUMMARY SEC. 7.0.

1.2 SYSTEM REQUIREMENTS.

HARDWARE REQUIREMENTS:

PDT-11 110/130 SYSTEM
8K MEMORY (8K WORDS) - MINIMUM
EXTERNAL LOOPBACK CONNECTORS (OPTIONAL) FOR COMM AND CLUSTER TERMINAL PORTS.
(SEE PROGRAM OPTIONS SEC. 2.4)

SOFTWARE REQUIREMENTS:

THIS EXERCISER IS DESIGNED TO RUN IN ANY OF THE FOLLOWING WAYS:

STAND ALONE
WITH APT MONITOR
WITH XXDP MONITOR

IT IS NOT DESIGNED TO RUN WITH THE DIAGNOSTIC SUPERVISOR.

1.3 RELATED DOCUMENTS AND STANDARDS.

DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS
APT
SYSMAC

175-003-009-02
MD-11-DZZMA
MD-11-DZQAC

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

THIS EXERCISER ASSUMES THAT THE POWER UP SELF TEST RUNS ERROR FREE.

1.5 ASSUMPTIONS

IF THE PROGRAM IS TO BE LOADED AND EXECUTED FROM THE APT MANUFACTURING SYSTEM, IT IS ASSUMED THAT LOCATION \$DEVN HAS BEEN INITIALIZED TO THE CONFIGURATION OF THE PDT-11 TO BE TESTED.

1.6 RUNNING SYSTEMS PROGRAMS

UNLESS THE SYSTEMS PROGRAMS ARE KNOWN TO INITIALIZE THE BAUD RATES OF EACH DEVICE THRU THE PARAMETER REGISTER, THE OPERATOR SHOULD POWER DOWN AND UP AGAIN WHEN FINISHED RUNNING THIS EXERCISER. THIS IS TO RESTORE THE PDT-11 TO ITS DEFAULT BAUD RATES. OTHERWISE, THE DEVICES WILL ATTEMPT TO RUN AT WHATEVER BAUD RATES WERE LAST USED BY THIS EXERCISER.

2.0 OPERATING INSTRUCTIONS.

2.1 LOADING AND STARTING PROCEDURES.

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED MEDIA, OR PROCEDURE FOR LOADING PROGRAMS UNDER XXDP MEDIA.

THE PROGRAM WILL AUTO-START AFTER LOADING TO LOCATION 'START'.

IT CAN BE STARTED FROM LOCATION 200.

NOTE: ALL RESTARTS MUST BE FROM LOCATION 240.

THE FOLLOWING EXAMPLE SHOWS A TYPICAL PROGRAM STARTUP ON A 24K SYSTEM WITHOUT A PRINTER AND THE OPERATOR DROPPING CLUSTER TERMINAL NUMBER TWO FROM THE TEST.

CVKDBAO PDT-11 110/130 SYS EX. 0 PASSES 0 TOTAL ERRORS (0 SOFT)

SWR = XXXXXX NEW =

THE PROGRAM SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (SWREG AT LOCATION 176) FROM THE CONSOLE. XXXXXX IN THE ABOVE TYPEOUT IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER. REFER TO SECTION 2.3, STEPS 3 AND 4, FOR OPERATOR ACTION AND THE OPERATIONAL SWITCH SETTINGS. IN ORDER TO DROP A DEVICE FROM THE TEST, AS IN THIS EXAMPLE, THE OPERATOR MUST SET BIT 14 (40000) IN THE SWITCH REGISTER.

IF BIT 14 (40000) IS SET IN THE SOFTWARE SWITCH REGISTER, THE PROGRAM WILL ALLOW THE OPERATOR TO CHANGE LOCATION \$DEVN VIA THE CONSOLE. IF BIT 14 (40000) IS SET, THE PROGRAM WILL TYPE THE FOLLOWING:

\$DEVN = XXXXXX NEW =

XXXXXX IN THE ABOVE TYPEOUT IS THE OCTAL CONTENTS OF THE DEVICE MAP, LOCATION \$DEVN. REFER TO SECTION 2.3, STEPS 3 AND 4, FOR THE METHOD USED TO CHANGE THE LOCATION. ALL REFERENCES IN SECTION 2.3, STEP 3, TO LOCATION 176 OR SOFTWARE SWITCH REGISTER WILL IMPLY LOCATION \$DEVN. REFER TO SECTION 2.4.1 FOR A DEFINITION OF EACH BIT IN LOCATION \$DEVN. FOR THIS EXAMPLE, TO DROP CLUSTER TERMINAL NUMBER TWO FROM THE TEST, THE OPERATOR MUST SET BIT 13 (20000) INTO LOCATION \$DEVN.

THE PROGRAM, AFTER THE OPERATOR HAS PERFORMED THE ABOVE, WILL AUTOSIZE MEMORY, DETERMINE WHAT OPTIONS ARE PRESENT (PRINTER, CLUSTER AND TU58), WHAT DEVICES ARE NOT TO BE TESTED VIA LOCATION \$DEVN, AND THEN TYPEOUT ON THE CONSOLE TERMINAL ITS FINDINGS. FOR THE EXAMPLE ABOVE, THE PROGRAM WILL TYPE THE FOLLOWING:

24K MEMORY PRESENT
PRINTER NOT PRESENT
CLUSTER TERMINAL #2 TESTING DROPPED

INSERT SCRATCH CASSETTES, TYPE 'P' TO PROCEED

(PROGRAM HALTS AND WAITS FOR 'P' AND THEN CONTINUES)

- NOTE1: IF THE OPERATOR INDICATES VIA THE SWITCH REGISTER THAT THE CONSOLE TERMINAL IS A VT132, ALL REFERENCES IN THIS DOCUMENT AND ALL TYPEOUTS BY THE PROGRAM DESCRIBING OR INDICATING THE VT100 WILL IMPLY VT132.
- NOTE2: BECAUSE HALTS ARE PROHIBITED UNDER APT, IT WILL ASSUME SCRATCH CASSETTES ARE ALREADY IN THE DRIVES AND BYPASS PRINTING THE WARNING MESSAGE AND THE HALT.
- NOTE3: IF RUNNING IN XXDP CHAIN MODE, THE PROGRAM WILL SET BIT 8 IN \$DEVN TO DROP TESTING ON TU58 TAPE UNIT 0 (LOAD DEVICE). THE WARNING MESSAGE AND HALT WILL BE BYPASSED BY THE PROGRAM. IF BIT 9 IN \$DEVN IS SET TO A ZERO, THE PROGRAM WILL TEST TU58 TAPE UNIT 1. IT IS ASSUMED THAT A SCRATCH CASSETTE IS IN TAPE DRIVE UNIT 1.
- NOTE4: IF TU58 TESTING IS INHIBITED (1400 IN \$DEVN), OR THE SYSTEM IS A PDT-11 110, THE CASSETTE MESSAGE AND HALT ARE BYPASSED.
- NOTE5: IF THE PROGRAM DETERMINES AN OPTION IS NOT PRESENT AND \$DEVN INDICATES THE OPTION IS TO BE TESTED, THE PROGRAM WILL NOT TEST THE DEVICE.

THE PROGRAM WILL BEGIN ACTUAL TESTING AT THIS POINT.
SEE SUMMARY OF TESTS SEC. 7.0.

DURING THE TESTS, THE PROGRAM WILL PRINT PROGRESS REPORTS (SEE SEC. 4.0) IF BIT 12 IS SET IN THE SWREG (SEE SEC 2.3).

END OF PASS AND TOTAL ERROR COUNT IS PRINTED AT THE CONCLUSION OF TESTING. THE PROGRAM WILL THEN JUMP TO THE BEGINNING OF THE ACTUAL TESTING AND THE SEQUENCE IS REPEATED UNTIL STOPPED BY THE OPERATOR

2.2 SPECIAL ENVIRONMENTS.

BEFORE INITIATING PDT-11 TESTING UNDER APT, THE UNIT UNDER TEST MUST BE INITIALIZED EITHER BY THE KEYBOARD RESET FUNCTION OR BY TURNING THE UNIT UNDER TEST OFF AND THEN ON.

IF SYNCHRONOUS (SYNC) COMMUNICATION (COMM) TESTS ARE RUN (BIT10 IN \$DEVN = 0) WHEN RUNNING UNDER APT, THE ASYNCHRONOUS (ASYN) COMM TESTS SHOULD ALSO BE RUN (BIT11 IN \$DEVN = 0). THIS WILL LEAVE THE COMM PORT SETUP IN ASYN MODE SO APT WILL BE ABLE TO COMMUNICATE WITH THE UNIT UNDER TEST (UUT).

IF THE PROGRAM IS TO BE RUN UNDER APT, SET LOCATION \$ENV TO 001 AND LOCATION \$ENVN TO THE DESIRED SETTINGS FROM THE BITS LISTED BELOW:

BIT5=0	(000)	ENABLE CONSOLE OUTPUT - FIRST PASS ONLY
BIT5=1	(040)	DISABLE CONSOLE OUTPUT
BIT7=0	(000)	USE PROGRAM DEFAULT SWITCH SETTINGS (000000)
BIT7=1	(200)	USE APT SWITCH REGISTER (\$SWREG) (SEE SECTION 2.3 FOR OPERATIONAL SWITCH SETTINGS)

IF BIT 5 EQUALS ZERO, THE PROGRAM WILL REPORT DURING THE FIRST PASS OF THE PROGRAM, AUTO-SIZING INFORMATION, DEVICES DROPPED FROM THE TEST AS SPECIFIED BY \$DEVN, ERRORS, AND THE END OF PASS PRINTOUT. BIT 5 IS UNCONDITIONALLY SET AT THE END OF PASS TO ENSURE THAT TYPEOUTS ARE NOT SENT OVER THE COMM PORT TO APT. IF THIS IS NOT DONE, APT WOULD REPORT A HUNG DIAGNOSTIC. SETTING BIT 7 EQUAL TO A 1, WILL ALLOW THE PROGRAM TO USE LOCATION \$SWREG AS THE SOFTWARE SWITCH REGISTER (SEE SECTION 2.3 FOR OPERATIONAL SWITCH SETTING).

IF IT IS NECESSARY TO ALLOW A NUMBER OF TUS8 SOFT ERRORS TO OCCUR BEFORE REPORTING AN ERROR TO APT, SET LOCATION \$USWR TO THE OCTAL NUMBER OF ALLOWABLE ERRORS. IF THE PROGRAM DETECTS A SOFT ERROR ON THE TUS8 TAPE DRIVE GREATER THAN THE NUMBER IN \$USWR, THE ERROR WILL BE REPORTED TO APT. LOCATION \$USWR IS NORMALLY SET TO ZERO WHICH WILL CAUSE ALL SOFT ERRORS TO BE REPORTED TO APT.

THE DEVICE MAP (\$DEVN) SHOULD BE SET UP AS DESCRIBED IN SECTION 2.4.1

2.3 OPERATIONAL SWITCH SETTINGS

THIS PROGRAM SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (SWREG AT LOC. 176) FROM THE CONSOLE. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G <^G>; (THIS WILL ALLOW THE CONSOLE TO ENTER DATA INTO LOC. 176).
- 2) THE MACHINE WILL THEN TYPE: ' SWR=XXXXXX NEW=' (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE CONSOLE:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED). LEADING ZEROS NEED NOT BE TYPED, AND IF MORE THAN 6 DIGITS ARE TYPED, THE LAST 6 WILL BE USED. IF A <CR> IS THE FIRST KEY DEPRESSED, THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U <^U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 3.
 - C) IF THE INPUT CHARACTER IS NOT ONE OF THE CHARACTERS MENTIONED ABOVE, A QUESTION MARK (?) WILL BE TYPED FOLLOWED BY A CARRIAGE RETURN AND A LINE FEED. ANY INPUT TYPED UP TO THE UNRECOGNIZABLE CHARACTER WILL BE IGNORED AND THE PROGRAM WILL SEND YOU BACK TO STEP 3 ABOVE.
- 4) THE DIAGNOSTIC WILL CONTINUE ON TYPING <CR>.

SOFTWARE SWITCH REGISTER OPTIONS (SWREG)

BIT 15 SET =	100000	= HALT ON ERROR
14 SET =	40000	= CHANGE \$DEVN VIA CONSOLE (ONLY AT PROGRAM START TIME).
13 SET =	20000	= INHIBIT ERROR TYPEOUTS
12 SET =	10000	= ENABLE PERFORMANCE REPORTS
10 SET =	2000	= BELL AND BLINKING BOLD TYPEOUT ON ERROR
09 SET =	1000	= CONSOLE TERMINAL IS A VT132 (SEE SECTION 2.1 NOTE1)

2.4 PROGRAM OPTIONS.

TYPING CONTROL C (^C) ON THE CONSOLE KEYBOARD AFTER THE PROGRAM HAS STARTED TESTING THE PDT-11, WILL CAUSE THE PROGRAM TO BE RESTARTED. REFER TO SECTION 2.1 OF THIS DOCUMENT FOR STARTING PROCEDURES.

2.4.1 DEVICE MAP--TESTING CONTROL

A DEVICE MAP (\$DEVN) IS EXAMINED BY THE PROGRAM TO DETERMINE WHICH DEVICES WILL BE TESTED, THE CLUSTER TERMINALS, THE ASYNCHRONOUS (ASYN)/SYNCHRONOUS (SYNC) COMMUNICATION (COMM) PORT, THE PRINTER, CLOCK, VT100, MEMORY AND THE TUS8.

THE DEFAULT VALUE = 000017 IT CAN BE CHANGED TO DO THE FOLLOWING:

\$DEVN

000017= INTERNAL LOOPBACK FOR ALL CLUSTER TERMINALS.
INTERNAL LOOPBACK FOR COMM PORT.
TEST CLUSTER OPTION
TEST PRINTER, ASYN/SYNC COMM PORTS, VT100,
TEST TUS8 DRIVE 0 AND 1, CLOCK, AND MEMORY.

BIT 15 SET=	100000=	DROP PRINTER TESTS
14 SET=	40000=	DROP CLUSTER TERM #3 TESTS
13 SET=	20000=	#2
12 SET=	10000=	#1
11 SET=	4000=	DROP ASYN COMM TESTS
10 SET=	2000=	DROP SYNC COMM TESTS
9 SET=	1000=	DROP TUS8 DRIVE 1 TESTS
8 SET=	400=	DROP TUS8 DRIVE 0 TESTS
7 SET=	200=	DROP CLOCK TESTS
6 SET=	100=	DROP VT100 TESTS
5 SET=	40=	TUS8 SEQUENTIAL BLOCK TESTING (0-777 OCTAL)
4 SET=	20=	DROP MEMORY TESTS

BIT 3 SET=	10=	INT. LOOPBACK (MAINT MODE) FOR COMM PORT
2 SET=	4=	FOR CLUSTER TERM #1
1 SET=	2=	FOR #2
0 SET=	1=	FOR #3

NOTES:

1. THE CLUSTER TERMINALS ARE INTENDED TO RUN IN LOOPBACK MODE ONLY.
2. THE COMM PORT IS INTENDED TO RUN IN LOOPBACK MODE ONLY.
3. THE PRINTER IS TESTED IN TRANSMIT MODE ONLY.
IT WILL BE EXERCISED IF SIZING DETERMINES IT IS PRESENT
OR LOOPBACK CONNECTOR INSTALLED AND BIT 15 = 0.
4. THE CLUSTER TERMINALS CAN NOT BE TESTED IN EXTERNAL LOOPBACK
WHEN THE COMM PORT IS INITIALIZED TO RUN IN INTERNAL LOOPBACK (\$DEVN xxxx10).

5. BITS 14-10 SETTINGS (\$DEVN) WILL OVERRIDE BITS 3-0 SETTINGS.

2.5 EXECUTION TIMES.

ASSUMING ALL DEVICES AND NO TAPE SUBSYSTEMS PRESENT:

1'ST PASS:

APPROXIMATELY 80 SEC TO TEST VT100, CLOCK, PRINTER, COMM DEVICES (SYNC/ASYN) AND CLUSTER TERMINALS.

SUBSEQUENT PASSES:

SAME AS ABOVE.

ASSUMING ALL DEVICES AND TAPE SUBSYSTEM (TU58) PRESENT:

1'ST PASS:

MAXIMUM TIME OF APPROXIMATELY 10.5 MINUTES TO TEST ALL DEVICES AS ABOVE AND BOTH TU58 TAPE DRIVES. TEST TIME WILL VARY CONSIDERABLY WITHIN THE 10.5 MINUTES WHILE TESTING THE TU58 TAPE DRIVES BECAUSE OF RANDOM BLOCK SELECTION CAUSING VARYING SEARCH TIMES. THIS CAN BE MINIMIZED BY SEQUENTIAL BLOCK TESTING IF SELECTED (BITS IN \$DEVN = 1).

SUBSEQUENT PASSES:

SAME AS ABOVE FOR THE TAPE SUBSYSTEM.

2.6 POWER FAIL

THERE IS NO POWER FAIL AUTO-RESTART CAPABILITY IN THE PDT-11.

3.0 ERROR INFORMATION.

A "WATCH-DOG" TIMER IS USED BY THE PROGRAM AT VARIOUS TIMES TO ENSURE THAT THE VT100 IS RUNNING. IF XOFF IS TRANSMITTED MANUALLY WHILE THE PROGRAM IS EXECUTING THE "WATCH-DOG" TIMER CODE AND XON IS NOT TRANSMITTED MANUALLY IN THE ALLOTTED AMOUNT OF TIME, A TIMEOUT WILL OCCUR AND THE PROGRAM WILL REPORT A "VT100 HUNG" ERROR MESSAGE. THEREFORE, THE OPERATOR SHOULD NOT PRESS ANY KEYS ON THE VT100 KEYBOARD WHICH WILL CAUSE XOFF TO BE SENT.

WHEN TESTING THE TUSB, THE SOFT ERROR COUNT WILL BE INCREMENTED (FOLLOWING THE ERROR REPORT), IF THE TUSB INDICATES IN ITS SUCCESS CODE BYTE (CONTAINED IN THE END PACKET) THAT THE READ OR WRITE FUNCTION SUCCEEDED WITH RETRIES.

ALL DEVICE ERRORS ON THE TUSB WILL CAUSE THE PROGRAM TO REPORT THE ERROR AND THEN DROP TESTING OF THE TUSB UNTIL THE NEXT PROGRAM PASS. A HARD READ OR WRITE ERROR DETECTED ON THE TUSB DRIVES WILL CAUSE THE PROGRAM TO REPORT THE ERROR AND DROP THE FAILING DRIVE FOR THIS PASS. IF ANOTHER DRIVE IS AVAILABLE, THE PROGRAM WILL START TESTING ON THIS DRIVE FOLLOWING THE MESSAGE "TUSB TAPE UNIT X DONE" (X=UNIT NUMBER). A SOFT READ OR WRITE ERROR WILL CAUSE THE PROGRAM TO CONTINUE TESTING ON THIS DRIVE FOLLOWING THE ERROR MESSAGE. A SOFT ERROR IS WHEN THE TUSB INDICATES THAT THE FUNCTION SUCCEEDED BUT WITH RETRIES.

3.1 ERROR REPORTING PROCEDURE.

NOTE: VT100 LINES 2-13 ARE USED FOR THE SCROLLING REGION FOR ERROR AND PERFORMANCE REPORTS. (LINES 14-24 ARE USED FOR DISPLAY TESTS, IF ENABLED.)

TYPICAL ERROR PRINTOUTS ARE SHOWN BELOW:
(ALL VALUES TYPED ARE OCTAL.)

CLUSTER	TERM	#3	DATA	COMP	ERR
ERROR #	ERR PC	EXPECT	RECV		
000011	006516	000200	000100		

TU58 ERROR PRINTOUTS ARE DESCRIBED BELOW:

TU58 DEVICE ERROR
ERROR # ERR PC ERR WRD OP-CODE UNIT #
AAAAAA BBBBBB CCCCCC DDDDDD EEEEE

AAAAAA= THE OCTAL NUMBER OF THE ERROR

BBBBBB= THE ADDRESS AT WHICH THE PROGRAM DETECTED THE FAILURE

CCCCCC= THE DEVICE ERROR WORD. A (1) IN ANY OF THE BITS
IN THE WORD INDICATES THE ERROR DETECTED. EACH
BIT IS DEFINED AS FOLLOWS:

BIT15=1 (100000) WHEN WAITING FOR A FLAG BYTE, AN
UNDEFINED CHARACTER WAS RECEIVED.
BIT14=1 (040000) WHEN READING A BLOCK, MORE THAN
4 DATA PACKETS WERE RECEIVED.
BIT13=1 (020000) WHEN WRITING A BLOCK, MORE THAN
4 CONT FLAG BYTES WERE RECEIVED.
BIT12=1 (010000) WHEN READING A BLOCK, A CONT FLAG
BYTE WAS RECEIVED.
BIT11=1 (004000) WHEN WRITING A BLOCK, A DATA FLAG
BYTE WAS RECEIVED.
BIT10=1 (002000) AN INIT FLAG BYTE WAS RECEIVED
DURING THE OPERATION.
BIT09=1 (001000) WHEN INITING THE DEVICE, THE 2ND
CHARACTER RECEIVED WAS NOT A CONT
FLAG BYTE.
BIT08=1 (000400) NOT USED
BIT07=1 (000200) TU58 HUNG (TIMEOUT ERROR)
BIT06=1 (000100) RECEIVER DONE FLAG NOT SET ON
RECEIVER INTERRUPT.
BIT05=1 (000040) PROGRAM RECEIVE BUFFER OVERFLOWED.
THE TU58 TRANSMITTED TOO MANY CHARACTERS.
BIT04=1 (000020) TRANSMIT READY NOT SET ON TRANSMIT
INTERRUPT.
BIT03=1 (000010) TRANSMIT INTERRUPTED WITH TRANSMIT
INTERRUPT ENABLE CLEARED.
BIT02=1 (000004) NOT USED
BIT01=1 (000002) NOT USED
BIT00=1 (000001) NOT USED

DDDDDD= THE FUNCTION BEING EXECUTED WHEN ERROR WAS DETECTED.
000004= THE PROGRAM WAS INITIALIZING THE TU58
000003= THE PROGRAM WAS WRITING A BLOCK
000403= THE PROGRAM WAS DOING A WRITE AND VERIFY
ON A BLOCK. (TU58 COMMAND)
000002= THE PROGRAM WAS READING A BLOCK.
000402= THE PROGRAM WAS READING A BLOCK WITH
INCREASED THRESHOLD. (TU58 COMMAND)

EEEEEE= THE TU58 UNIT NUMBER UNDER TEST
000000= TU58 DRIVE UNIT 0
000001= TU58 DRIVE UNIT 1

TU58 WRITE ERROR

ERROR # ERR PC OP-CODE UNIT # BLOCK # PKT WRD SUCCESS BYT CNT SUMMARY
AAAAAA BBBBBB CCCCCC DDDDDD EEEEE EFFFFF GGGGGG HHHHHH IIIIII

AAAAAA= THE OCTAL NUMBER OF THE ERROR

BBBBBB= THE ADDRESS AT WHICH THE PROGRAM DETECTED A FAILURE.

CCCCCC= THE FUNCTION THAT WAS BEING EXECUTED WHEN THE ERROR WAS DETECTED.

000003= THE PROGRAM WAS WRITING A BLOCK

000403= THE PROGRAM WAS DOING A WRITE VERIFY ON A BLOCK

DDDDDD= THE TU58 UNIT NUMBER UNDER TEST

000000= TU58 DRIVE UNIT 0

000001= TU58 DRIVE UNIT 1

EEEEEE= THE BLOCK NUMBER THAT WAS BEING WRITTEN

FFFFFF= ACCUMULATED FUNCTION AND END PACKET ERROR WORD.
A (1) IN ANY OF THE BITS IN THE WORD INDICATES THE ERROR(S) DETECTED. EACH BIT IS DEFINED AS FOLLOWS:

BIT15=1 (100000) AN ERROR WAS DETECTED WHILE READING DATA PACKET 4

BIT14=1 (040000) AN ERROR WAS DETECTED WHILE READING DATA PACKET 3

BIT13=1 (020000) AN ERROR WAS DETECTED WHILE READING DATA PACKET 2

BIT12=1 (010000) AN ERROR WAS DETECTED WHILE READING DATA PACKET 1

BIT11=1 (004000) NOT USED

BIT10=1 (002000) NOT USED

BIT09=1 (001000) LESS THAN 4 DATA PACKETS RECEIVED DURING READ OPERATION.

BIT08=1 (000400) LESS THAN 4 CONTINUE FLAG BYTES RECEIVED DURING WRITE OPERATION

END PACKET ERROR BITS

BIT07=1 (000200) PACKET CHECKSUM FAILURE

BIT06=1 (000100) PACKET BYTE COUNT INCORRECT

BIT05=1 (000040) PACKET OP-CODE BYTE NOT EQUAL
END PACKET OP-CODE

BIT04=1 (000020) TU58 REPORTING FAILURE DURING OPERATION. SEE GGGGGG.

BIT03=1 (000010) UNIT NUMBER NOT EQUAL UNIT UNDER TEST

BIT02=1 (000004) PACKET SEQUENCE NUMBER NOT 0

BIT01=1 (000002) PACKET DATA BYTE COUNT NOT EQUAL TO A BLOCK (1000 OCTAL)

BIT00=1 (000001) PACKET SUMMARY BITS INDICATES AN ERROR

GGGGGG= THE SUCCESS CODE BYTE AS READ FROM THE TUSB IN THE END PACKET.

000000= NORMAL SUCCESS (TUSB DETECTED NO ERRORS)
000001= SUCCESS BUT WITH RETRIES (SOFT ERROR)
000377= FAILED SELF-TEST
000376= PARTIAL OPERATION (END OF MEDIUM)
000370= BAD UNIT NUMBER
000367= NO CARTRIDGE
000365= WRITE PROTECTED
000357= DATA CHECK ERROR
000340= SEEK ERROR
000337= MOTOR STOPPED
000320= BAD OP-CODE
000311= BAD RECORD NUMBER

HMMMMH= THE DATA BYTE COUNT WORD AS READ FROM THE TUSB IN THE END PACKET. THIS WORD INDICATES THE NUMBER OF READ/WRITE DATA BYTES TRANSFERRED DURING THE OPERATION. IF ALL OF THE DATA WAS TRANSFERRED, THIS WORD SHOULD BE EQUAL TO 1000 OCTAL (# OF BYTES IN A BLOCK).

IIIIII= THE SUMMARY WORD AS READ FROM THE TUSB IN THE END PACKET.

10XXXX= SPECIAL CONDITIONS (ERRORS) SEE GGGGGG.
04XXXX= TRANSFER ERROR
02XXXX= MOTION ERROR
01XXXX= LOGIC ERROR
XXXX=RESERVED BITS

TUSB READ ERROR

ERROR #	ERR PC	OP-CODE	UNIT #	BLOCK #	PKT WRD	DAT WRD	SUCCESS	BYT CNT	SUMMARY
AAAAAA	BBBBBB	CCCCCC	DDDDDD	EEEEEE	FFFFFF	GGGGGG	HMMMMH	IIIIII	JJJJJJ

AAAAAA= THE OCTAL NUMBER OF THE ERROR.

BBBBBB= THE ADDRESS AT WHICH THE FAILURE WAS DETECTED

CCCCCC= THE FUNCTION BEING EXECUTED WHEN ERROR WAS DETECTED
000002= THE PROGRAM WAS READING A BLOCK
000402= THE PROGRAM WAS READING A BLOCK WITH INCREASED THRESHOLD.

DDDDDD= THE TUSB UNIT NUMBER UNDER TEST
000000= TUSB DRIVE UNIT 0
000001= TUSB DRIVE UNIT 1

EEEEEE= THE BLOCK NUMBER THAT WAS BEING READ

FFFFFF= ACCUMULATED FUNCTION AND END PACKET ERROR WORD.
A (1) IN ANY OF THE BITS IN THE WORD INDICATES THE ERROR(S) DETECTED. EACH BIT IS DEFINED AS FOLLOWS:

BIT15=1 (100000) AN ERROR WAS DETECTED WHILE

BIT14=1 (040000) READING DATA PACKET 4
AN ERROR WAS DETECTED WHILE
READING DATA PACKET 3
BIT13=1 (020000) AN ERROR WAS DETECTED WHILE
READING DATA PACKET 2
BIT12=1 (010000) AN ERROR WAS DETECTED WHILE
READING DATA PACKET 1
BIT11=1 (004000) NOT USED
BIT10=1 (002000) NOT USED
BIT09=1 (001000) LESS THAN 4 DATA PACKETS RECEIVED
DURING READ OPERATION.
BIT08=1 (000400) LESS THAN 4 CONTINUE FLAG BYTES
RECEIVED DURING WRITE OPERATION

END PACKET ERROR BITS

BIT07=1 (000200) PACKET CHECKSUM FAILURE
BIT06=1 (000100) PACKET BYTE COUNT INCORRECT
BIT05=1 (000040) PACKET OP-CODE BYTE NOT EQUAL
END PACKET OP-CODE
BIT04=1 (000020) TUSB REPORTING FAILURE DURING
OPERATION. SEE HHHHH.
BIT03=1 (000010) UNIT NUMBER NOT EQUAL UNIT UNDER TEST
BIT02=1 (000004) PACKET SEQUENCE NUMBER NOT 0
BIT01=1 (000002) PACKET DATA BYTE COUNT NOT
EQUAL TO A BLOCK (1000 OCTAL)
BIT00=1 (000001) PACKET SUMMARY BITS INDICATES AN
ERROR.

GGGGG=

THIS WORD CONTAINS THE ACCUMULATED ERROR BITS
FOR EACH OF THE DATA PACKETS READ. A (1) IN ANY
OF THE BITS INDICATES A FAILURE WAS DETECTED.

DATA PACKET 4 ERROR BITS

BIT14=1 (040000) DATA PACKET CHECKSUM FAILURE
BIT13=1 (020000) DATA CHECKSUM FAILURE (READ
CHECKSUM NOT EQUAL WRITE)
BIT12=1 (010000) DATA PACKET BYTE COUNT INCORRECT

DATA PACKET 3 ERROR BITS

BIT11=1 (004000) DATA PACKET CHECKSUM FAILURE
BIT10=1 (002000) DATA CHECKSUM FAILURE (READ
CHECKSUM NOT EQUAL WRITE)
BIT09=1 (001000) DATA PACKET BYTE COUNT INCORRECT

DATA PACKET 2 ERROR BITS

BIT08=1 (000400) DATA PACKET CHECKSUM FAILURE
BIT07=1 (000200) DATA CHECKSUM FAILURE (READ
CHECKSUM NOT EQUAL WRITE)
BIT06=1 (000100) DATA PACKET BYTE COUNT INCORRECT

DATA PACKET 1 ERROR BITS

BIT05=1 (000040) DATA PACKET CHECKSUM FAILURE
BIT04=1 (000020) DATA CHECKSUM FAILURE (READ
CHECKSUM NOT EQUAL WRITE)
BIT03=1 (000010) DATA PACKET BYTE COUNT INCORRECT

MMMMM= THE SUCCESS CODE BYTE AS READ FROM THE TUS8 IN
THE END PACKET.

000000= NORMAL SUCCESS (TUS8 DETECTED NO ERRORS)
000001= SUCCESS BUT WITH RETRIES (SOFT ERROR)
000377= FAILED SELF-TEST
000376= PARTIAL OPERATION (END OF MEDIUM)
000370= BAD UNIT NUMBER
000367= NO CARTRIDGE
000365= WRITE PROTECTED
000357= DATA CHECK ERROR
000340= SEEK ERROR
000337= MOTOR STOPPED
000320= BAD OP-CODE
000311= BAD RECORD NUMBER

IIIIII= THE DATA BYTE COUNT WORD AS READ FROM THE TUS8
IN THE END PACKET. THIS WORD INDICATES THE
NUMBER OF DATA BYTES TRANSFERRED DURING THE
OPERATION. IF ALL THE DATA PACKETS WERE READ, THIS
NUMBER SHOULD EQUAL 1000 OCTAL (# OF BYTES IN A BLOCK)

JJJJJJ= THE SUMMARY WORD AS READ FROM THE TUS8 IN THE
END PACKET.

10XXXX= SPECIAL CONDITIONS (ERRORS) SEE MMMMM.
04XXXX= TRANSFER ERROR
02XXXX= MOTION ERROR
01XXXX= LOGIC ERROR
XXXX=RESERVED BITS

BITS 15,13, AND 10 OF THE SWITCH REGISTER (SWREG) CONTROL THE
SEQUENCE OF EVENTS AFTER AN ERROR IS DETECTED.

BIT 15 SET: CAUSES THE PROGRAM TO HALT IN THE ERROR ROUTINE.
IF THE PROGRAM IS CONTINUED BY TYPING 'P',
THE PROGRAM WILL PROCEED.

BIT 13 SET: DISABLES THE PRINTING OF THE ERROR MESSAGE.

BIT 10 SET: CAUSES THE BELL TO RING ON ERROR.

THE ERROR ROUTINE SUPPORTS THE CONTROL G <^G> FUNCTION.
REFER TO SECTION 2.3 FOR DETAILS.

NOTE: SINCE THERE ARE NO SPECIFIC TEST NUMBERS, APT WILL
REPORT ALL ERRORS AS TEST 0 ERRORS.

3.2 ERROR HALTS.

THE ONLY HALTS IN THE EXERCISER ARE AFTER PRINTING THE CASSETTE WARNING MESSAGE AND IN THE ERROR ROUTINE. THE LATTER HALT IS EXECUTED ONLY IF BIT 15 OF THE SWITCH REGISTER (SWREG) IS SET WHEN AN ERROR OCCURS.

4.0 PROGRESS AND PERFORMANCE REPORTS

NOTE: THE SCROLLING REGION FOR PROGRESS AND ERROR REPORTS WILL BE LINES 2-13. LINES 14-24 WILL BE A SEPERATE SCROLLING REGION FOR VT100 DISPLAY TESTS. LINE 1 WILL ALWAYS DISPLAY THE PROGRAM TITLE, PASS COUNT, AND ERROR COUNT.

THE FOLLOWING REPORTS ARE ENABLED BY BIT 12 IN THE SWITCH REGISTER (SWREG LOC 176) AND WILL APPEAR FOLLOWING LOADING AND STARTING AS DESCRIBED IN SECTION 2.1 (ASSUMING ALL DEVICES ARE TO BE TESTED AND NO ERRORS):

MEMORY TESTS DONE	ALL MEMORY FROM LOCATION 0 TO THE BEGINNING OF THE XXDP MONITOR/ABSOLUTE LOADER HAS BEEN TESTED.
VT100 RUNNING	ESCAPE SEQUENCES FOR TERMINAL IDENTIFY, DIAGNOSTIC STATUS, AND CURSOR REPORTS HAVE BEEN SENT AND RECEIVED BACK FROM THE VT100. THE VT100 DISPLAY TESTS (VISUAL VERIFY ONLY) ARE LEFT RUNNING IN INTERRUPT MODE AT 4800 BAUD.
CLOCK RUNNING	100 INTERRUPTS HAVE BEEN DETECTED BEFORE EXERCISING THE NEXT DEVICE. THE CLOCK CONTINUES RUNNING IN INTERRUPT MODE.
PRINTER RUNNING (DONE)	5 LINES HAVE BEEN PRINTED WITH ERROR BIT CHECKING ONLY. THE PRINTER RUNS CONTINUOUSLY IN INTERRUPT MODE AT 1200 BAUD. ITS ACTUAL PERFORMANCE IS A VISUAL CHECK IF PRINTER CONNECTED. IF THE PRINTER WAS RUNNING IN EXTERNAL LOOPBACK MODE, THE PROGRAM WILL REPORT "PRINTER DONE" AND NO FURTHER TESTING WILL BE PERFORMED THIS PASS.
SYNC COMM DONE	CHARACTERS 0 THRU 377 HAVE BEEN TRANSMITTED AND RECEIVED WITH ODD PARITY ENABLED. INTERRUPTS ARE THEN DISABLED AND THE SYNC COMM IS NO LONGER EXERCISED. THIS IS SO THAT THE ASYNC COMM CAN BE EXERCISED FOR THE DURATION OF THE PASS.
ASYNC COMM RUNNING	CHARS 0 THRU 377 HAVE BEEN TRANSMITTED AND RECEIVED BEFORE EXERCISING THE NEXT DEVICE. IT CONTINUES RUNNING IN INTERRUPT MODE AT 9600 BAUD WITH ODD PARITY ENABLED AND REPEATS THE 0 THRU 377 CYCLE
CLUSTER TERMINAL #1 RUNNING	CHARACTERS 0 THRU 377 HAVE BEEN TRANSMITTED AND RECEIVED BEFORE EXERCISING THE NEXT DEVICE. THE TERMINAL KEEPS RUNNING CONTINUOUSLY AT 300 BAUD IN INTERRUPT MODE AND REPEATS THE 0 THRU 377 CYCLE

CLUSTER TERMINAL #2 RUNNING
 SAME AS #1

CLUSTER TERMINAL #3 RUNNING
 SAME AS #1

TU58 RUNNING THIS MESSAGE INDICATES THAT THE TU58
 HAS RESPONDED TO THE INITIALIZATION
 SEQUENCE.

TU58 TAPE UNIT 0 DONE THIS MESSAGE INDICATES THAT TEN
 BLOCKS HAVE BEEN WRITTEN AND READ ON
 THIS UNIT. NO FURTHER TESTING IS DONE
 ON THIS UNIT SO THAT THE NEXT UNIT
 CAN BE TESTED IF AVAILABLE.

TU58 TAPE UNIT 1 DONE THIS MESSAGE INDICATES THAT TEN
 BLOCKS HAVE BEEN WRITTEN AND READ ON
 THIS UNIT. THIS IS THE LAST DEVICE
 UNDER TEST FOR THIS PASS, THEREFORE,
 THE TESTING IS STOPPED FOR THIS PASS.

END OF PASS #1 TOTAL ERRORS THIS PASS: 0
 SOFT ERRORS THIS PASS: 0 (...AND ENTIRE PROCESS REPEATS)

5.0 DEVICE REGISTERS.

PARAMETER REGISTER:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORT SELECT				BAUD RATE				USR!	USR!	PAR!	PAR!	CHAR	!MAIN!		
								LT2!	LT1!	TYP!	EN!	LENGTH	!		

177420 (WP ONLY)

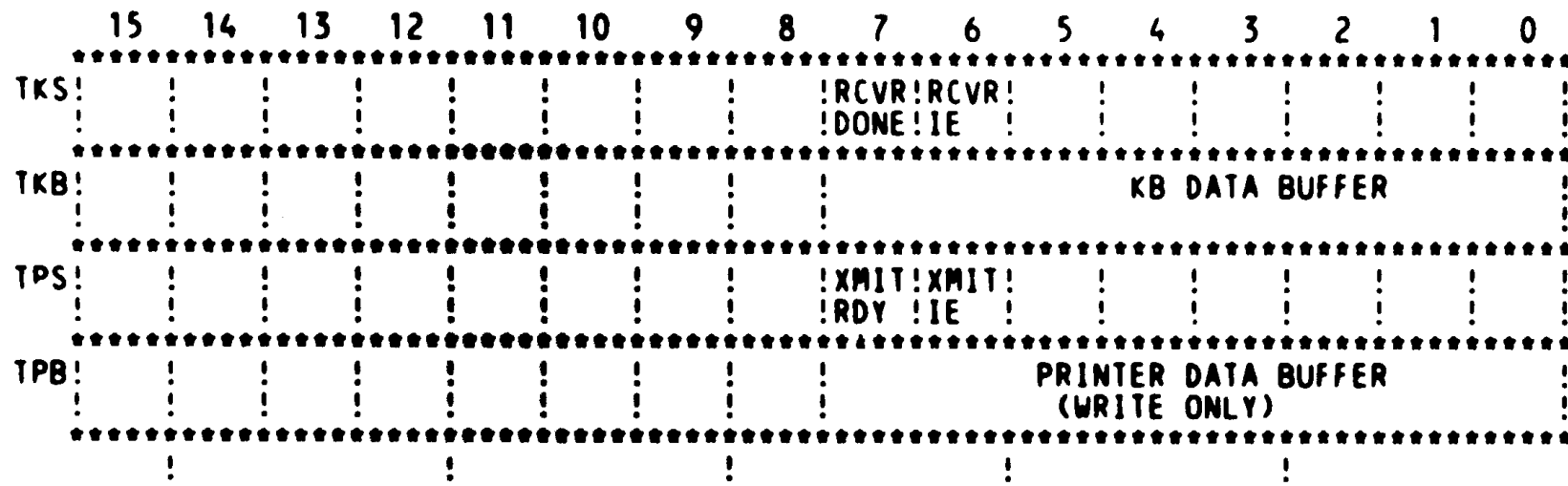
LINE CLOCK:

VECTOR: 100

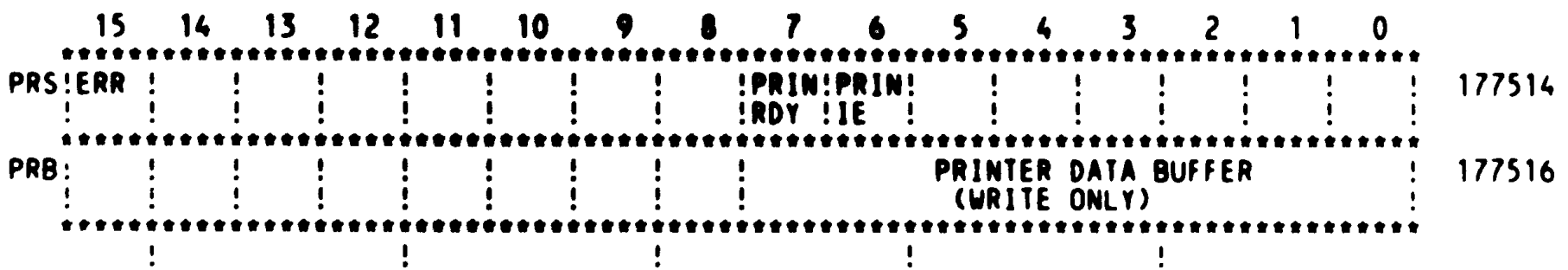
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								!CLK							
								!IE							

177546

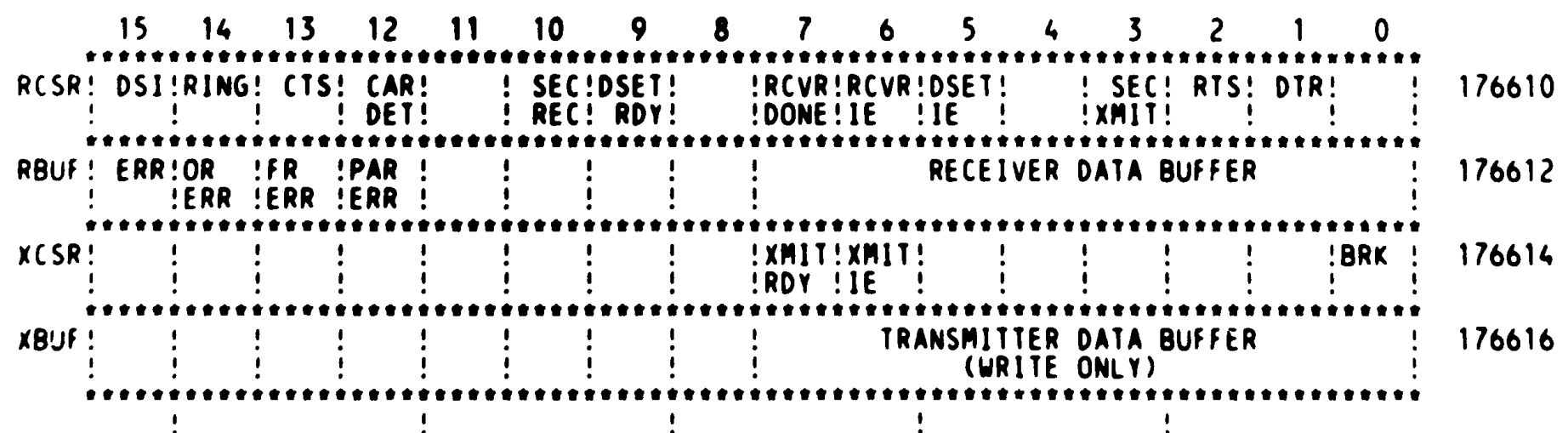
CONSOLE: ADDR: 177560-177566 VECTOR: 60/64
 CLUSTER #1: 176500-176506 300/304
 #2: 176510-176516 310/314
 #3: 176520-176526 320/324



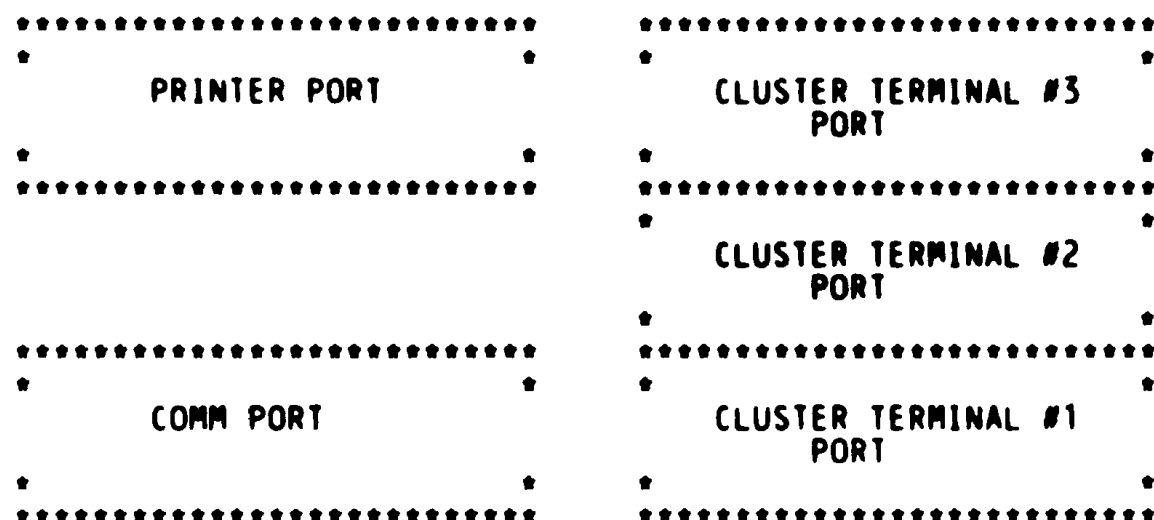
PRINTER: VECTOR: 200



ASYNC MODEM: VECTOR: 330/334



6.0 DEVICE CONNECTOR LAYOUT



7.0 SUMMARY OF TESTS (SEE PROGRAM LISTING FOR DETAILS ON SPECIFIC TESTS)

PHASE 1

MEMORY TESTS A. LOC 0 THRU END OF PROGRAM IS TESTED FOR TIMEOUT.
 B. END OF PROGRAM TO BOTTOM OF XXDP MONITOR/ABSOLUTE LOADER
 IS TESTED WITH A 125252 AND 052525 OCTAL PATTERN.
 C. IF 28K PRESENT, MEMORY FROM 28K TO 30K IS SIMILARLY TESTED.

PHASE 2

ALL AVAILABLE PERIPHERALS ARE EXERCISED IN INTERRUPT MODE UNTIL END OF PASS.

SEE SECTION 2.4 FOR DETAILS OF SETTING UP THE DEVICE MAP (\$DEVN) FOR DEVICE TO BE TESTED.
SEE SECTION 4.0 FOR DETAILS ON PROGRESS REPORTS. THE PROGRAM FLOW FOR PHASE 2 IS AS FOLLOWS:

START VT100	VT100 LINK VERIFIED USING ESCAPE SEQUENCES. DISPLAY TESTS LEFT RUNNING.
START LINE CLOCK	INTERRUPTS ARE DETECTED.
START PRINTER	CONTINUOUS LINES ARE PRINTED.
START SYNC COMM PORT	CHARS 27 THRU 377 " " " " 1 PASS ONLY
START ASYNC COMM PORT	CHARS 0 THRU 377 TESTED CONTINUOUSLY.
START CLUSTER TERMINAL #1	CHARS 0 THRU 377 ARE XMITTED, REC'D AND TESTED.
START CLUSTER TERMINAL #2	SAME
START CLUSTER TERMINAL #3	SAME

PHASE 3

WHILE THE ABOVE DEVICES ARE INTERRUPTING RANDOMLY, TUS8 SUBSYSTEM TESTS BEGIN:

THE PROGRAM INITIALIZES THE TUS8 BY SENDING THE TUS8 "INIT" SEQUENCE AND WAITING FOR THE APPROPRIATE RESPONSE FROM THE TUS8.

FOR THE PURPOSE OF THIS EXERCISER, THE TAPE IS DIVIDED INTO TWO TAPE RANGES; THE FIRST HALF AND THE SECOND HALF OF TAPE. THE PROGRAM GENERATES A RANDOM NUMBER TO DETERMINE WHICH HALF OF THE TAPE IS TO BE EXERCISED DURING THIS PASS OF THE PROGRAM. THIS TAPE RANGE WILL BE USED IN TESTING BOTH TUS8 DRIVES FOR THIS PASS OF THE PROGRAM. ALL BLOCK NUMBERS TO BE EXERCISED ARE GENERATED RANDOMLY AND THEY ARE MASKED TO THE TAPE RANGE SELECTED. IF BIT 5 IN \$DEVN IS SET, THE BLOCK NUMBERS GENERATED WILL BE SEQUENTIAL INSTEAD OF RANDOM, AND THE TAPE WILL NOT BE DIVIDED INTO TAPE RANGES. ALL TRANSFERS TO THE TUS8 ARE ONE BLOCK TRANSFERS (1000 OCTAL BYTES). ALL DATA TO BE WRITTEN IN A BLOCK WILL BE GENERATED RANDOMLY. THE WRITE FUNCTION WILL BE RANDOMLY SELECTED FOR EACH BLOCK AS WRITE OR WRITE VERIFY. THE READ FUNCTION WILL BE RANDOMLY SELECTED FOR EACH BLOCK AS READ OR READ WITH INCREASED THRESHOLD. IF BIT 5 IN \$DEVN IS SET, THE WRITE AND READ FUNCTION WILL STRICTLY BE A WRITE AND READ FUNCTION AND NOT A WRITE VERIFY OR READ WITH INCREASED THRESHOLD. FOR EACH BLOCK TO BE EXERCISED, THE PROGRAM WILL WRITE THE BLOCK AND THEN READ AND CHECK THE BLOCK TO BE CORRECT. THE PROGRAM EXERCISES TEN RANDOM BLOCKS (SEQUENTIAL IF \$DEVN BIT 5 = 1) PER TUS8 TAPE DRIVE BEFORE EXERCISING THE NEXT TUS8 TAPE DRIVE THE SAME WAY.

z

```
6638 .TITLE CVKDBAO PDT-11 110/130 SYS EX.  
(1) :*COPYRIGHT (C) 1979  
(1) :*DIGITAL EQUIPMENT CORP.  
(1) :*MAYNARD, MASS. 01754  
(1) :*  
(1) :*PROGRAM BY BRUCE HANSEN  
(1) :*  
(1) :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
(1) :*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.  
(1) :*  
6639  
6640 .SBTTL OPERATIONAL SWITCH SETTINGS  
6641  
6642 .          SWITCH          USE  
6643 .          -----          ---  
6644 .          15             HALT ON ERROR  
6645 .          14             CHANGE $DEVN VIA KEYBOARD (PROGRAM START  
6646 .                          TIME ONLY)  
6647 .          13             INHIBIT ERROR TYPEOUTS  
6648 .          12             ENABLE PERFORMANCE REPORTS  
6649 .          10             BELL AND REVERSE VIDEO TYPEOUT ON ERROR  
6650 .          09             CONSOLE TERMINAL IS A VT132.  
6651  
6652  
6653  
6654 .SBTTL BASIC DEFINITIONS  
(1) :*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***  
(1) 001100 STACK= 1100  
(1) .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL  
(1) .EQUIV IOT,SCOPE     ;;BASIC DEFINITION OF SCOPE CALL  
(1)  
(1) :*MISCELLANEOUS DEFINITIONS  
(1) 000011 HT= 11      ;;CODE FOR HORIZONTAL TAB  
(1) 000012 LF= 12      ;;CODE FOR LINE FEED  
(1) 000015 CR= 15      ;;CODE FOR CARRIAGE RETURN  
(1) 000200 CRLF= 200    ;;CODE FOR CARRIAGE RETURN-LINE FEED  
(1) 177776 PS= 177776 ;;PROCESSOR STATUS WORD  
(1) .EQUIV PS,PSW  
(1) 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER  
(1) 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER  
(1) 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER  
(1) 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER  
(1)  
(1) :*GENERAL PURPOSE REGISTER DEFINITIONS  
(1) 000000 R0= %0      ;;GENERAL REGISTER  
(1) 000001 R1= %1      ;;GENERAL REGISTER  
(1) 000002 R2= %2      ;;GENERAL REGISTER  
(1) 000003 R3= %3      ;;GENERAL REGISTER  
(1) 000004 R4= %4      ;;GENERAL REGISTER  
(1) 000005 R5= %5      ;;GENERAL REGISTER  
(1) 000006 R6= %6      ;;GENERAL REGISTER  
(1) 000007 R7= %7      ;;GENERAL REGISTER  
(1) 000006 SP= %6      ;;STACK POINTER  
(1) 000007 PC= %7      ;;PROGRAM COUNTER
```

(1)			
(1)		;*PRIORITY LEVEL DEFINITIONS	
(1)	000000	PR0= 0	::PRIORITY LEVEL 0
(1)	000040	PR1= 40	::PRIORITY LEVEL 1
(1)	000100	PR2= 100	::PRIORITY LEVEL 2
(1)	000140	PR3= 140	::PRIORITY LEVEL 3
(1)	000200	PR4= 200	::PRIORITY LEVEL 4
(1)	000240	PR5= 240	::PRIORITY LEVEL 5
(1)	000300	PR6= 300	::PRIORITY LEVEL 6
(1)	000340	PR7= 340	::PRIORITY LEVEL 7
(1)			
(1)		;*SWITCH REGISTER SWITCH DEFINITIONS	
(1)	100000	SW15= 100000	
(1)	040000	SW14= 40000	
(1)	020000	SW13= 20000	
(1)	010000	SW12= 10000	
(1)	004000	SW11= 4000	
(1)	002000	SW10= 2000	
(1)	001000	SW09= 1000	
(1)	000400	SW08= 400	
(1)	000200	SW07= 200	
(1)	000100	SW06= 100	
(1)	000040	SW05= 40	
(1)	000020	SW04= 20	
(1)	000010	SW03= 10	
(1)	000004	SW02= 4	
(1)	000002	SW01= 2	
(1)	000001	SW00= 1	
(1)		.EQUIV SW09,SW9	
(1)		.EQUIV SW08,SW8	
(1)		.EQUIV SW07,SW7	
(1)		.EQUIV SW06,SW6	
(1)		.EQUIV SW05,SW5	
(1)		.EQUIV SW04,SW4	
(1)		.EQUIV SW03,SW3	
(1)		.EQUIV SW02,SW2	
(1)		.EQUIV SW01,SW1	
(1)		.EQUIV SW00,SW0	
(1)			
(1)		;*DATA BIT DEFINITIONS (BIT00 TO BIT15)	
(1)	100000	BIT15= 100000	
(1)	040000	BIT14= 40000	
(1)	020000	BIT13= 20000	
(1)	010000	BIT12= 10000	
(1)	004000	BIT11= 4000	
(1)	002000	BIT10= 2000	
(1)	001000	BIT09= 1000	
(1)	000400	BIT08= 400	
(1)	000200	BIT07= 200	
(1)	000100	BIT06= 100	
(1)	000040	BIT05= 40	
(1)	000020	BIT04= 20	
(1)	000010	BIT03= 10	
(1)	000004	BIT02= 4	
(1)	000002	BIT01= 2	
(1)	000001	BIT00= 1	

```

(1) .EQUIV BIT09,BIT9
(1) .EQUIV BIT08,BIT8
(1) .EQUIV BIT07,BIT7
(1) .EQUIV BIT06,BIT6
(1) .EQUIV BIT05,BIT5
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0

(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;: "T" BIT
(1) 000014 TRTVEC= 14 ;:TRACE TRAP
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;:POWER FAIL
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;:"TRAP" TRAP
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR

6655
6656 ;:*****
6657 ;:SPECIAL ASCII CHARACTERS
6658 ;:*****
6659
6660 000033 ESC= 33 ;:ESCAPE
6661 000021 XON= 21
6662 000023 XOFF= 23
6663
6664 ;:*****
6665 ;:* PDT-11 110/130 DEVICE VECTORS
6666 ;:*****
6667
6668 000200 PRVEC= 200 ;:PRINTER
6669
6670 000100 CLKVEC= 100 ;:LINE CLOCK
6671
6672 000330 AMRVEC= 330 ;:ASYNC MODEM RECVR
6673 000334 AMXVEC= 334 ;: XMITR
6674
6675 000340 SMRVEC= 340 ;:SYNC MODEM RECVR
6676 000344 SMXVEC= 344 ;: XMITR
6677
6678 000260 TURVEC= 260 ;:TU58 RECVR
6679 000264 TUTVEC= 264 ;:TU58 XMITR
6680
6681 000300 TK1VEC= 300 ;:CLUSTER TERM #1 VECTORS
6682 000304 TP1VEC= 304
6683 000310 TK2VEC= 310 ;: #2
6684 000314 TP2VEC= 314
6685 000320 TK3VEC= 320 ;: #3
6686 000324 TP3VEC= 324
  
```

```

6687
6688
6689
6690
6691
6692      177420      PARAM= 177420      ;PARAMETER REGISTER      (WRITE ONLY)
6693
6694      177546      CLK= 177546      ;LINE CLOCK
6695
6696      177514      PRS= 177514      ;PRINTER STATUS REG
6697      177516      PRB= 177516      ;          BUFFER      (WRITE ONLY)
6698
6699      176610      AMRC= 176610      ;ASYNC MODEM RECVR STATUS REG
6700      176612      AMRB= 176612      ;          BUFFER
6701      176614      AMXC= 176614      ;          XMITR STATUS REG
6702      176616      AMXB= 176616      ;          BUFFER      (WRITE ONLY)
6703
6704      176620      SMRC= 176620      ;SYNC MODEM RECVR STATUS REG
6705      176622      SMRB= 176622      ;          BUFFER      (READ ONLY)
6706      176622      SMPAR= 176622      ;          PARAMETER REG      (WRITE ONLY)
6707      176624      SMXC= 176624      ;          XMITR STATUS REG
6708      176626      SMXB= 176626      ;          BUFFER      (WRITE ONLY)
6709
6710      177170      TURCSR= 177170      ;TUS8 RECEIVER CSR
6711      177172      TURXDB= 177172      ;          COMMON DATA BUFFER
6712      177174      TUXCSR= 177174      ;          TRANSMITTER CSR
6713
6714      176500      TK1S= 176500      ;CLUSTER TERMINAL #1
6715      176502      TK1B= 176502
6716      176504      TP1S= 176504
6717      176506      TP1B= 176506
6718
6719      176510      TK2S= 176510      ;          #2
6720      176512      TK2B= 176512
6721      176514      TP2S= 176514
6722      176516      TP2B= 176516
6723
6724      176520      TK3S= 176520      ;          #3
6725      176522      TK3B= 176522
6726      176524      TP3S= 176524
6727      176526      TP3B= 176526
    
```

```

6729
6730
6731
6732
6733
6734
6735
6736      010000
6737      000000
6738      050000
6739      060000
6740      020000
6741      030000
6742      040000
6743
6744
6745
6746      000000
6747      000400
6748      001000
6749      001400
6750      002000
6751      002400
6752      003000
6753      003400
6754      004000
6755      004400
6756      005000
6757      005400
6758      006000
6759      006400
6760      007000
6761      007400
6762
6763
6764
6765
6766
6767      000060
6768      000020
6769
6770
6771
6772      000000
6773      000004
6774      000010
6775      000014
6776
6777
6778
6779      000002
6780
    
```

```

:*****
:      PARAMETER REGISTER EQUATES      WRITE ONLY
:*****

:BIT 14-12 PORT SELECT
:
CONSOL= 10000      :SELECT CONSOLE TERMINAL
CT1= 0             :CLUSTER TERMINAL #1
CT2= 50000        :#2
CT3= 60000        :#3
PRT= 20000        :PRINTER PORT
AMOD= 30000       :ASYNC COMM PORT
ULITE= 40000      :USER LIGHTS

:BIT 11-8 BAUD RATE (DELTA = 400)
:
B50= 0             :50 BAUD
B75= 400           :75
B110= 1000         :110
B134= 1400         :134.5
B150= 2000         :150
B300= 2400         :300
B600= 3000         :600
B1200= 3400        :1200
B1800= 4000        :1800
B2000= 4400        :2000
B2400= 5000        :2400 (CLUSTER DEFAULT)
B3600= 5400        :3600
B4800= 6000        :4800
B7200= 6400        :7200
B9600= 7000        :9600 (PRINTER AND ASYNC DEFAULT)
B19200= 7400       :19200

:IF 110 BAUD SELECTED, 2 STOP BITS ARE ASSUMED.

:BITS 4-5 PARITY CONTROL
:
EPAR= 60           :EVEN PARITY ENABLE
OPAR= 20           :ODD PARITY ENABLE (DEFAULT)

:BITS 3-2 CHARACTER LENGTH (DELTA = 4)
:
CHAR5= 0           :5 BITS/CHAR
CHAR6= 4           :6
CHAR7= 10          :7
CHAR8= 14          :8 (PRINTER, TERMINAL, MODEM DEFAULT)

:BIT 1 MAINTENANCE BIT
:
MAINT= BIT1        :ENABLE MAINT MODE
                  :PRINTER AND TUS8 DO NOT HAVE MAINT MODE
    
```

```

6782          ;*****
6783          ;*          ASYNC MODEM BIT DEFINITIONS
6784          ;*****
6785
6786          ;RCSR
6787          ;
6788          100000 DSI=      BIT15      ;DATA SET INTERRUPT
6789          040000 RING=     BIT14      ;RING
6790          020000 CTS=      BIT13      ;CLEAR TO SEND
6791          010000 CDET=     BIT12      ;CARRIER DETECTED
6792          002000 SREC=     BIT10      ;SECONDARY RECD/SUPERVISORY RECD
6793          001000 DSRDY=    BIT9       ;DATA SET RDY
6794          000200 DONE=     BIT7       ;RECVR DONE
6795          000100 IE=       BIT6       ;RECVR IE
6796          000040 DSIE=     BIT5       ;DATA SET IE
6797          000010 SXMIT=    BIT3       ;SECONDARY XMIT/SUPERV XMIT
6798          000004 RTS=     BIT2       ;REQ TO SEND
6799          000002 DTR=     BIT1       ;DATA TERMINAL RDY
6800
6801          ;RBUF
6802          ;
6803          100000 ERR=      BIT15      ;ERROR
6804          040000 ORERR=   BIT14      ;OVERRUN ERROR
6805          020000 FRERR=   BIT13      ;FRAMING ERROR
6806          010000 PARERR=  BIT12      ;PARITY ERROR
6807
6808          ;XCSR
6809          ;
6810          000200 RDY=      BIT7       ;XMIT RDY
6811          000100 IE=      BIT6       ;XMIT IE
6812          000001 BREAK=   BIT0       ;BREAK
    
```

```

6814          ;*****
6815          ;*          SYNC MODEM BIT DEFINITIONS
6816          ;*****
6817
6818          ;RCSR
6819          :
6820          100000 DSC= BIT15          ;DATA SET CHANGE
6821          040000 RING= BIT14         ;RING
6822          020000 CTS= BIT13          ;CLEAR TO SEND
6823          010000 CDET= BIT12         ;CARRIER DETECTED
6824          004000 RACT= BIT11         ;RECVR ACTIVE
6825          002000 SREC= BIT10         ;SECONDARY RECD/SUPERVISORY RECD
6826          001000 DSRDY= BIT9         ;DATA SET RDY
6827          000400 STSYN= BIT8         ;STRIP SYNC
6828          000200 DONE= BIT7          ;RECVR DONE
6829          000100 IE= BIT6           ;RECVR IE
6830          000040 DSIE= BIT5          ;DATA SET IE
6831          000020 SRSYN= BIT4         ;SEARCH SYNC
6832          000010 SXMIT= BIT3         ;SECONDARY XMIT/SUPERV XMIT
6833          000004 RTS= BIT2           ;REQ TO SEND
6834          000002 DTR= BIT1          ;DATA TERMINAL RDY
6835
6836          ;RBUF
6837          :
6838          100000 ERR= BIT15           ;ERROR
6839          040000 ORERR= BIT14         ;OVERRUN ERROR
6840          010000 PARERR= BIT12        ;PARITY ERROR
6841
6842          ;PARCSR PARAMETER CONT REG
6843          :
6844          000000 SYNC2= 0             ;2 SYNC CHARS (DEFAULT)
6845          040000 SYNC1= BIT14         ;1
6846
6847          ;BITS 10-11 WORD LENGTH
6848          :
6849          000000 CHR5= 0               ;5 BITS/CHAR
6850          002000 CHR6= 2000           ;6
6851          004000 CHR7= 4000           ;7
6852          006000 CHR8= 6000           ;8 (DEFAULT)
6853
6854          ;BITS 8-9 PARITY CONTROL
6855          :
6856          001400 ENVPAR= 1400         ;ENABLE EVEN PARITY
6857          001000 ODDPAR= 1000        ;ENABLE ODD PARITY (DEFAULT)
6858
6859          ;XCSR
6860          :
6861          100000 DNA= BIT15            ;DATA NOT AVAIL
6862          010000 MM= BIT12            ;MAINT MODE
6863          004000 MCLK= BIT11          ;MAINT CLOCK
6864          000400 MR= BIT8             ;MASTER RESET
6865          000200 RDY= BIT7           ;XMIT RDY
6866          000100 IE= BIT6            ;XMIT IE
6867          000040 DNAIE= BIT5         ;DATA NOT AVAIL IE
6868          000020 SEND= BIT4          ;SEND
6869          000010 HFDUP= BIT3         ;1=HALF DUPLEX, 0=FULL DUPLEX (DEFAULT = 0 FOR TESTS)
    
```

```

6870
6871
6872
6873
6874
6875
6876
6877
6878      000200
6879      000100
6880
6881
6882
6883      000200
6884      000100
6885      000001
6886
6887
6888
6889      000020
6890      000004
6891      000001
6892      000002
6893      000012
6894      000200
6895      000010
6896      001000
6897
6898
6899
6900      000000
6901      000001
6902      000002
6903      000003
6904      000007
6905      000100
6906
6907
(1)
(1)      000000
(1)
(1)
(1)
(1)      000174
(1) 000174 000000
(1) 000176 000000
6908
6909
6910
6911
6912      000174
6913 000174 000000
6914 000176 000000
6915
6916
6917      000200
    
```

```

*****
;*      TU58 BIT DEFINITIONS
*****

:RCSR
:
RDONE= BIT7      ;RECVR DONE
IE=      BIT6      ;RECVR IE

:TXCS
:
RDY=      BIT7      ;XMIT RDY
IE=      BIT6      ;XMIT IE
BREAK= BIT0      ;XMIT BREAK

:LEVEL 1 CODES (FLAG BYTE)
:
TCONT= 20      ;CONTINUE
TINIT= 4      ;INIT PROTOCOL
TDATA= 1      ;DATA PACKET
TCMDPK= 2      ;COMMAND PACKET
TMSG= 12      ;COMMAND PACKET SIZE
TMSIZ= 128.   ;DATA PACKET SIZE
TRTRY= 10     ;RETRY COUNTER FOR ERRORS
TWC= 512.     ;READ/WRITE WORD COUNT

:LEVEL 2 CODES (OPCODE)
:
TNOP= 0      ;NO OPERATION
INITT= 1     ;INITIALIZE
TREAD= 2     ;READ FUNCTION
TWRITE= 3    ;WRITE FUNCTION
TDIAGN= 7    ;DIAGNOSE FUNCTION
TENDPK= 100  ;END PACKET FROM PERIPHERAL

.SBTTL TRAP CATCHER
      .=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      .=174
DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
*****
;*      SWITCH REGISTER
*****

      .=174
DISPREG: .WORD 0      ;SOFTWARE DISPLAY REG
SWREG: .WORD 0      ;SOFTWARE SWITCH REG

      .=200
    
```

```
6918 000200 000137 001716          JMP      START          ;USE ONLY ONCE. WILL BE OVERLAID BY
6919                                     ;PRINTER VECTOR ADDR.
6920                                     .=240
6921 000240 000137 001716          JMP      START          ;RESTART ADDR
6922                                     .=1000
6923                                     .SBTTL  APT PARAMETER BLOCK
6924                                     ;*****
(1)                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                     ;*****
(1) 001000                          .SX=.      ;;SAVE CURRENT LOCATION
(1) 000024 000024                      .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200                      200      ;;FOR APT START UP
(1) 000044 000044                      .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000                      $APTHDR  ;;POINT TO APT HEADER BLOCK
(1) 001000 001000                      .=.SX    ;;RESET LOCATION COUNTER
(2)                                     ;*****
(1)                                     ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)                                     ;INTERFACE SPEC.
(1) 001000                          $APTHD:
(1) 001000 000000                      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001170                      $MBADR: .WORD $MAIL     ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 001166                      $STMT:  .WORD 630.     ;;RUN TIME OF LONGEST TEST
(1) 001006 001166                      $PASTM: .WORD 630.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000000                      $UNITM: .WORD 0        ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000030                      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
6925 .SBTTL  ACT11 HOOKS
(1)                                     ;*****
(2)                                     ;HOOKS REQUIRED BY ACT11
(1) 001014                          $$VPC=.      ;SAVE PC
(1) 000046 000046                      .=46
(1) 000046 011560                      $ENDAD   ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1) 000052 000052                      .=52
(1) 000052 000000                      .WORD 0      ;;2)SET LOC.52 TO ZERO
(1) 001014 001014                      .=$$VPC     ;; RESTORE PC
```

6942
 (1)
 (2)
 (1)
 (1)
 (1)
 (1)
 (1) 001100 001100
 (1) 001100 000000
 (1) 001102 000
 (1) 001103 000
 (1) 001104 000000
 (1) 001106 000000
 (1) 001110 000000
 (1) 001112 000000
 (1) 001114 000
 (1) 001115 001
 (1) 001116 000000
 (1) 001120 000000
 (1) 001122 000000
 (1) 001124 000000
 (1) 001126 000000
 (1) 001130 000000
 (1) 001132 000000
 (1) 001134 000
 (1) 001135 000
 (1) 001136 000000
 (1) 001140 177570
 (1) 001142 177570
 (1) 001144 177560
 (1) 001146 177562
 (1) 001150 177564
 (1) 001152 177566
 (1) 001154 000
 (1) 001155 002
 (1) 001156 012
 (1) 001157 000
 (1) 001160 177607 000377
 (1) 001164 077
 (1) 001165 015
 (1) 001166 000012
 (2)
 (2)
 (2)
 (3)
 (2)
 (2) 001170
 (2) 001170 000000
 (2) 001172 000000
 (2) 001174 000000
 (2) 001176 000000
 (2) 001200 000000
 (2) 001202 000000
 (2) 001204 000000
 (2) 001206 000000
 (2) 001210

```

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

      . =1100
$CMTAG:      ;; START OF COMMON TAGS
      .WORD 0
$TSTNM: .BYTE 0      ;; CONTAINS THE TEST NUMBER
$ERFLG: .BYTE 0      ;; CONTAINS ERROR FLAG
$ICNT: .WORD 0       ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR: .WORD 0      ;; CONTAINS SCOPE LOOP ADDRESS
$LPERR: .WORD 0      ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL: .WORD 0      ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB: .BYTE 0      ;; CONTAINS ITEM CONTROL BYTE
$ERMAX: .BYTE 1      ;; CONTAINS MAX. ERRORS PER TEST
$ERRPC: .WORD 0      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR: .WORD 0      ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR: .WORD 0      ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT: .WORD 0      ;; CONTAINS 'GOOD' DATA
$BDDAT: .WORD 0      ;; CONTAINS 'BAD' DATA
      .WORD 0      ;; RESERVED--NOT TO BE USED
      .WORD 0
$AUTOB: .BYTE 0      ;; AUTOMATIC MODE INDICATOR
$INTAG: .BYTE 0      ;; INTERRUPT MODE INDICATOR
      .WORD 0
$SWR: .WORD DSWR      ;; ADDRESS OF SWITCH REGISTER
$DISPLAY: .WORD DDISP ;; ADDRESS OF DISPLAY REGISTER
$TKS: 177560          ;; TTY KBD STATUS
$TKB: 177562          ;; TTY KBD BUFFER
$TPS: 177564          ;; TTY PRINTER STATUS REG. ADDRESS
$TPB: 177566          ;; TTY PRINTER BUFFER REG. ADDRESS
$NULL: .BYTE 0       ;; CONTAINS NULL CHARACTER FOR FILLS
$FILLS: .BYTE 2      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC: .BYTE 12     ;; INSERT FILL CHARS. AFTER A 'LINE FEED'
$TPFLG: .BYTE 0      ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
$QUES: .ASCII /?/    ;; QUESTION MARK
$CRLF: .ASCII <15>   ;; CARRIAGE RETURN
$LF: .ASCIZ <12>     ;; LINE FEED
;*****
.SBTTL APT MAILBOX-ETABLE
;*****
.EVEN
$MAIL:      ;; APT MAILBOX
$MSGTY: .WORD AMSGTY ;; MESSAGE TYPE CODE
$FATAL: .WORD AFATAL ;; FATAL ERROR NUMBER
$TESTN: .WORD ATESTN ;; TEST NUMBER
$PASS: .WORD APASS   ;; PASS COUNT
$DEVCT: .WORD ADEVCT ;; DEVICE COUNT
$UNIT: .WORD AUNIT   ;; I/O UNIT NUMBER
$MSGAD: .WORD AMSGAD ;; MESSAGE ADDRESS
$MSGLG: .WORD AMSGLG ;; MESSAGE LENGTH
$ETABLE:      ;; APT ENVIRONMENT TABLE
    
```

(2)	001210	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
(2)	001211	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
(2)	001212	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
(2)	001214	000000	\$USWR:	.WORD	AUSWR	::USER SWITCHES
(2)	001216	000000	\$CPUOP:	.WORD	ACPUOP	::CPU TYPE,OPTIONS
(2)			::*			BITS 15-11=CPU TYPE
(2)			::*			11/04=01,11/05=02,11/20=03,11/40=04,11/45-05
(2)			::*			11/70=06,PDQ=07,Q=10
(2)			::*			BIT 10=REAL TIME CLOCK
(2)			::*			BIT 9=FLOATING POINT PROCESSOR
(2)			::*			BIT 8=MEMORY MANAGEMENT
(2)	001220	000	\$MAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001221	000	\$MTYP1:	.BYTE	AMTYP1	::MEM. TYPE,BLK#1
(2)			::*			MEM.TYPE BYTE -- (HIGH BYTE)
(2)			::*			900 NSEC CORE=001
(2)			::*			300 NSEC BIPOLAR=002
(2)			::*			500 NSEC MOS=003
(2)	001222	000000	\$MADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			::*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPF" ABOVE
(2)	001224	000	\$MAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001225	000	\$MTYP2:	.BYTE	AMTYP2	::MEM. TYPE,BLK#2
(2)	001226	000000	\$MADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001230	000	\$MAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001231	000	\$MTYP3:	.BYTE	AMTYP3	::MEM. TYPE,BLK#3
(2)	001232	000000	\$MADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001234	000	\$MAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001235	000	\$MTYP4:	.BYTE	AMTYP4	::MEM. TYPE,BLK#4
(2)	001236	000000	\$MADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001240	000300	\$VECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001242	000000	\$VECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001244	176500	\$BASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001246	000017	\$DEV:	.WORD	ADEV	::DEVICE MAP
(2)	001250		\$SETEND:			
(2)			.MEXIT			
(3)	001250	001	TYPFLG:	.BYTE	1	::CLEAR WHEN DISPLAY PROCESS IS ACTIVE
(3)						::(MORE COMMENTS AT "STYPE")
(3)	001251	000	ESCFLG:	.BYTE	0	::SET TO 1 WHEN DISPLAY PROCESS IS TYPING AN ESCAPE SEQUE
(3)	001252	000	GFLAG:	.BYTE	0	::SET TO 1 WHEN CONTROL G RECEIVED WHILE IN TYPE ROUTINE
(3)	001253	000	XFLAG:	.BYTE	0	::SET TO 1 IF XOFF HAS BEEN RECEIVED
(3)	001254	000000	VTDONE:	.WORD	0	::SET TO 1 WHEN DISPLAY PROCESS FINISHES A PASS
(3)	001256	000000	SERTTL:	.WORD	0	::TOTAL SOFT ERROR COUNT
(3)	001260	000000	SERUN0:	.WORD	0	::SOFT ERROR COUNTER UNIT 0
(3)	001262	000000	SERUN1:	.WORD	0	::SOFT ERROR COUNTER UNIT 1
(3)	001264	000000	KILFLG:	.WORD	0	::EOP DEVICE KILL FLAG
(3)	001266	000000	DEVACT:	.WORD	0	::DEVICE ACTIVE FLAG
(3)	001270	000000	DELFLG:	.WORD	0	::VIDEO PAUSE CONTROL FLAG
(3)	001272	000000	DELCNT:	.WORD	0	::VIDEO PAUSE CONTROL COUNT
(3)	001274	000000	DEVCHG:	.WORD	0	::CHANGE DEVICE MAP VIA KBD IF SET

```
(1) .SBTTL ERROR POINTER TABLE
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;* EM ;;POINTS TO THE ERROR MESSAGE
(1) ;* DH ;;POINTS TO THE DATA HEADER
(1) ;* DT ;;POINTS TO THE DATA
(1) ;* DF ;;POINTS TO THE DATA FORMAT
(1)
(1) 001276 $ERRTB:
6943 ;; GLOBAL DATA
6944 ;ERR 01 - MEMORY TIMEOUT ON READ (0-$LSTAD)
6945 001276 017171 020545 021254 .WORD EM1,DH2,DT2,0
001304 000000
6946 ;ERR 02 - MEMORY TIMEOUT ON WRITE [$LSTAD - (MAXMEM-LOADER AREA)]
6947 001306 017171 020545 021254 .WORD EM1,DH2,DT2,0
001314 000000
6948 ;ERR 03 - MEMORY DATA COMPARE ERROR [$LSTAD - (MAXMEM-LOADER AREA)]
6949 001316 017210 020571 021264 .WORD EM2,DH3,DT3,0
001324 000000
6950 ;ERR 04 - MEMORY TIMEOUT ON WRITE (160000 - 167776)
6951 001326 017171 020545 021254 .WORD EM1,DH2,DT2,0
001334 000000
6952 ;ERR 05 - MEMORY DATA COMPARE ERROR (160000 - 167776)
6953 001336 017210 020571 021264 .WORD EM2,DH3,DT3,0
001344 000000
6954 ;ERR 06 - SYNCHRONOUS COMM DID NOT RECEIVE SYNC CHARACTER
6955 001346 017242 020526 021246 .WORD EM3,DH1,DT1,0
001354 000000
6956 ;ERR 07 - PRINTER STATUS ERROR
6957 001356 017306 020633 021300 .WORD EM4,DH4,DT4,0
001364 000000
6958 ;ERR 10 - CLUSTER TERMINAL #1 DATA COMPARE ERROR
6959 001366 017333 020664 021310 .WORD EM5,DH5,DT5,0
001374 000000
6960 ;ERR 11 - CLUSTER TERMINAL #2 DATA COMPARE ERROR
6961 001376 017371 020664 021322 .WORD EM6,DH5,DT6,0
001404 000000
6962 ;ERR 12 - CLUSTER TERMINAL #3 DATA COMPARE ERROR
6963 001406 017427 020664 021334 .WORD EM7,DH5,DT7,0
001414 000000
6964 ;ERR 13 - ASYNCHRONOUS COMM DATA COMPARE ERROR
6965 001416 017465 020664 021346 .WORD EM8,DH5,DT8,0
001424 000000
6966 ;ERR 14 - SYNCHRONOUS COMM DATA COMPARE ERROR
6967 001426 017516 020664 021346 .WORD EM9,DH5,DT8,0
001434 000000
6968 ;ERR 15 - CLOCK HUNG
6969 001436 017546 020526 021246 .WORD EM23,DH1,DT1,0
001444 000000
6970 ;ERR 16 - PRINTER HUNG
```

6971	001446	017557	020633	021300	.WORD EM24,DH4,DT4,0
6972	001454	000000			:ERR 17 - CLUSTER TERMINAL #1 HUNG
6973	001456	017574	020526	021246	.WORD EM25,DH1,DT1,0
6974	001464	000000			:ERR 20 - CLUSTER TERMINAL #2 HUNG
6975	001466	017611	020526	021246	.WORD EM26,DH1,DT1,0
6976	001474	000000			:ERR 21 - CLUSTER TERMINAL #3 HUNG
6977	001476	017626	020526	021246	.WORD EM27,DH1,DT1,0
6978	001504	000000			:ERR 22 - SYNCHRONOUS COMM HUNG
6979	001506	017643	020526	021246	.WORD EM28,DH1,DT1,0
6980	001514	000000			:ERR 23 - ASYNCHRONOUS COMM HUNG
6981	001516	017662	020526	021246	.WORD EM29,DH1,DT1,0
6982	001524	000000			:ERR 24 - SYNCHRONOUS COMM DATA AVAILABLE NOT SET (SPARE ERROR LOCATION)
6983	001526	017702	020526	021246	.WORD EM32,DH1,DT1,0
6984	001534	000000			:ERR 25 - NO RESPONSE FROM THE VT100
6985	001536	020076	020526	021246	.WORD EM125,DH1,DT1,0
6986	001544	000000			:ERR 26 - SYNCHRONOUS COMM - RECEIVER DONE NOT SET
6987	001546	020006	020526	021246	.WORD EM34,DH1,DT1,0
6988	001554	000000			:ERR 27 - ROM TIMED OUT AT ADDRESS 170000
6989	001556	020045	020526	021246	.WORD EM127,DH1,DT1,0
6990	001564	000000			:ERR 30 - VT100 REPORTED SELF-TEST FAILURE
6991	001566	020125	020526	021246	.WORD EM130,DH1,DT1,0
6992	001574	000000			:ERR 31 - OPTION PRESENT (STP) NOT DETECTED BY THE VT100
6993	001576	020166	020721	021360	.WORD EM131,DH31,DT31,0
6994	001604	000000			:ERR 32 - UNRECOGNIZED RESPONSE FROM VT100 (ESC SEQ NOT EQUAL EXPECTED)
6995	001606	020241	020526	021246	.WORD EM132,DH1,DT1,0
6996	001614	000000			:ERR 33 - SYNCHRONOUS COMM MODEM LOOPBACK SIGNALS NOT SET IN CSR
6997	001616	020302	020664	021370	.WORD EM133,DH5,DT33,0
6998	001624	000000			:ERR 34 - VT100 HUNG
6999	001626	020063	020526	021246	.WORD EM134,DH1,DT1,0
7000	001634	000000			:ERR 35 - ASYNCHRONOUS COMM MODEM LOOPBACK SIGNALS NOT SET IN CSR
7001	001636	020362	020664	021370	.WORD EM140,DH5,DT33,0
7002	001644	000000			:ERR 36 - TUS8 DEVICE ERROR
7003	001646	020443	020750	021402	.WORD EM135,DH32,DT34,0
7004	001654	000000			:ERR 37 - TUS8 WRITE ERROR (HARD ERROR)
7005	001656	020465	021020	021416	.WORD EM136,DH33,DT35,0
7006	001664	000000			:ERR 40 - TUS8 READ ERROR (HARD ERROR)
7007	001666	020506	021127	021442	.WORD EM137,DH34,DT36,0
7007	001674	000000			

7008						
7009	001676	020465	021020	021416	;ERR 41 - TUSB WRITE ERROR (SOFT ERROR - SUCCESS WITH RETRIES)	
	001704	000000			.WORD EM136,DH33,DT35,0	
7010						
7011	001706	020506	021127	021442	;ERR 42 - TUSB READ ERROR (SOFT ERROR - SUCCESS WITH RETRIES)	
	001714	000000			.WORD EM137,DH34,DT36,0	

```

7013      .SBTTL PROGRAM
7014
7015 001716 012706 001100 START: MOV #STACK,SP ;SET UP STACK POINTER
7035 001722 012746 000340 MOV #PR7,-(SP) ;:PUT NEW PS ON STACK
(1) 001726 012746 001734 MOV #64$,-(SP) ;:PUT NEW PC ON STACK
(1) 001732 000002 RTI ;:POP NEW PC AND PS
(1) 001734
7036 001734 122737 000001 001210 64$: CMPB #APTENV,$ENV ;CHECK IF RUNNING UNDER APT
7037 001742 001401 BEQ START1 ;IF YES THEN SKIP RESET INSTRUCTION
7038 001744 000005 RESET ;CLEAR THE WORLD !!!!!
7039 001746 012737 013462 000060 START1: MOV #TKSRV1,TKVEC ;SETUP KEYBOARD INTERRUPT VECTOR
7040 001754 012737 000340 000062 MOV #PR7,TKVEC+2
7041 001762 105737 177565 TSTB @#177565 ;TEST ODD BYTE ON CONSOLE IN CASE LOADED BY APT.
7042 ;THIS WILL RESTORE COMM/CONSOLE ADDRESSES TO NORMAL.
7044
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
(1) 001766 012706 001100 MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
(1) 001772 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
(1) 001774 022706 001140 CMP #SWR,R6 ;:DONE?
(1) 002000 001374 BNE -6 ;:LOOP BACK IF NO
(1) 002002 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
(1) ;:INITIALIZE A FEW VECTORS
(1) 002006 012737 025570 000030 MOV #ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
(1) 002014 012737 000340 000032 MOV #340,@#EMTVEC+2 ;:LEVEL 7
(1) 002022 012737 027112 000034 MOV #STRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
(1) 002030 012737 000340 000036 MOV #340,@#TRAPVEC+2 ;:LEVEL 7
(2) ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002036 013746 000004 MOV @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
(2) 002042 012737 002076 000004 MOV #64$,@#ERRVEC ;:SET UP ERROR VECTOR
(2) 002050 012737 177570 001140 MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
(2) 002056 012737 177570 001142 MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
(2) 002064 022777 177777 177046 CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
(2) 002072 001012 BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;:AND THE HARDWARE SWR IS NOT = -1
(2) 002074 000403 BR 65$ ;:BRANCH IF NO TIMEOUT
(2) 002076 012716 002104 64$: MOV #65$,(SP) ;:SET UP FOR TRAP RETURN
(2) 002102 000002 RTI
(2) 002104 012737 000176 001140 65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
(2) 002112 012737 000174 001142 MOV #DISPREG,DISPLAY
(2) 002120 012637 000004 66$: MOV (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
(1)
(2) 002124 005037 001176 CLR $PASS ;:CLEAR PASS COUNT
(2) 002130 132737 000200 001211 BITB #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
(2) 002136 001403 BEQ 67$ ;:YES,USE NON-APT SWITCH
(2) 002140 012737 001212 001140 MOV #SSWREG,SWR ;:NO,USE APT SWITCH REGISTER
(2) 002146
7045
7046 002146 012700 001250 1$: MOV #TYPFLG,R0 ;CLEAR FLAGS
7047 002152 005020 CLR (R0)+ ;CLEAR THE LOCATION
7048 002154 022700 001276 CMP #SERRTB,R0 ;CHECK TO SEE IF DONE
7049 002160 001374 BNE 1$ ;CLEAR NEXT LOCATION IF NOT DONE
7050 002162 105037 014344 CLRB XIEFLG
7051 002166 012737 000777 007040 MOV #777,BLKNUM
7052
7053 ;TYPE THE PROGRAM NAME--INITIALIZE VT100 ATTRIBUTES
    
```

```

7054 002174 112737 000001 001250      MOVB    #1,TYPFLG      ;SET "TYPE" UP TO NOT NEED TO USE ESCAPE
7055                                     ;SEQUENCES UNTIL VT100 TESTING STARTS.
7056 002202 104401 015670      TYPE,   TITLE         ;INITIALIZE SCREEN AND TYPE TITLE
7057 002206 005737 000042      TST    @#42           ;ARE WE RUNNING UNDER XXDP/ACT?
7058 002212 001023 000000      BNE    2$             ;BRANCH IF YES
7059 002214 122737 000001 001210      CMPB   #APTENV,$ENV   ;ARE WE RUNNING UNDER APT?
7060 002222 001417 000000      BEQ    2$             ;BRANCH IF YES
7061 002224 023727 001140 000176      CMP    SWR,#SWREG     ;SOFTWARE SWITCH REGISTER SELECTED??
7062 002232 001031 000000      BNE    3$             ;BRANCH IF NOT
7063 002234 104406 000000      GTSWR                     ;GET SOFTWARE SWITCH REGISTER SETTINGS
7064 002236 032777 040000 176674      .BIT   #BIT14,@SWR    ;CHECK IF DEVICE MAP IS TO BE CHANGED
7065 002244 001424 000000      BEQ    3$             ;IF 0 DO NOT CHANGE DEVICE MAP
7066 002246 005237 001274      INC    DEVCHG         ;SET SOFTWARE FLAG
7067 002252 104406 000000      GTSWR                     ;GO GET NEW VALUE
7068 002254 005037 001274      CLR    DEVCHG         ;CLEAR DEVICE MAP CHANGE FLAG
7069 002260 000416 000000      BR     3$
7070 002262 112737 000001 001134 2$:   MOVB   #1,$AUTOB      ;SET AUTO-MODE INDICATOR
7071 002270 022737 011560 000042      CMP    #SENDAD,@#42   ;CHECK IF ACT11
7072 002276 001407 000000      BEQ    3$             ;IF YES ASSUME SCRATCH DRIVES IN 0 & 1
7073 002300 122737 000020 000041      CMPB   #20,@#41      ;CHECK IF XXDP CHAIN MODE
7074 002306 001003 000000      BNE    3$             ;IF NOT CHAIN MODE THEN CONTINUE
7075 002310 052737 000400 001246      BIS    #BIT8,$DEVM    ;DROP LOAD MEDIA FROM THE DEVICE MAP
7076 002316 004737 011602 000000 3$:   JSR    PC,SIZE        ;SEE WHO IS ON THE SYSTEM
7077 002322 123727 001210 000001      CMPB   $ENV,#APTENV   ;ARE WE RUNNING UNDER APT?
7078 002330 001421 000000      BEQ    LOOP           ;BR IF YES & DONT HALT
7079 002332 005737 000042      TST    @#42           ;CHECK IF LOADED UNDER XXDP/ACT11
7080 002336 001016 000000      BNE    LOOP           ;IF YES SKIP WARNING AND HALT MESSAGE
7081 002340 005737 012620      TST    TUPRES         ;CHECK IF TU58 IS PRESENT
7082 002344 001413 000000      BEQ    LOOP           ;TU58 IS NOT PRESENT
7083 002346 032737 000400 001246      BIT    #BIT8,$DEVM    ;TU58 TAPE UNIT 0 THERE?
7084 002354 001404 000000      BEQ    4$             ;BR IF YES
7085 002356 032737 001000 001246      BIT    #BIT9,$DEVM    ;TU58 TAPE UNIT 1 THERE?
7086 002364 001003 000000      BNE    LOOP           ;BR IF NO
7087 002366 104401 016714 000000 4$:   TYPE   ,WARNING      ;INSERT SCRATCH TU58 CASSETTES
7088 002372 000000 000000      HALT
7089 002374 012706 001100 000000 LOOP: MOV    #STACK,SP      ;LOOP HERE AFTER EOP & RESET STACK
7090 002400 004737 013270 000000      JSR    PC,TIMER1     ;TIMER TO ALLOW 'P' TO PRINT BEFORE RESET
7091
7092 002404 000005 000000      RESET                  ;CLEAR ALL INTERRUPT ENABLES
7093 002406 012737 000102 000100      MOV    #CLKVEC+2,CLKVEC ;SETUP CLOCK TRAP CATCHER
7094 002414 005037 000102 000000      CLR    CLKVEC+2      ;
7095
7096 002420 005037 001264 000000      CLR    KILFLG        ;CLEAR EOP DEVICE KILL FLAG
7097 002424 005037 001266 000000      CLR    DEVACT        ;CLEAR DEVICE ACTIVE FLAG
7098
7099 002430 000000 000000 000000 5$:
(1) 002430 012746 000000      MOV    #PRO,-(SP)     ;;PUT NEW PS ON STACK
(1) 002434 012746 002442      MOV    #64$,-(SP)    ;;PUT NEW PC ON STACK
(1) 002440 000002      RTI                   ;;POP NEW PC AND PS
(1) 002442
7100
7101 002442 032737 000010 001246      BIT    #BIT3,$DEVM    ;CHECK IF EXTERNAL LOOPBACK
7102 002450 001007 000000      BNE    1$             ;IF NOT SETUP FOR INTERNAL LOOPBACK
7103 002452 042737 000002 176610      BIC    #DTR,AMRC     ;TOGGLE DTR TO SETUP COMM PORT
7104 002460 052737 000002 176610      BIS    #DTR,AMRC     ;
7105 002466 000406 000000      BR     2$             ;GO SETUP COMM PORT

```

```

7106 002470 042737 000006 176610 1$: BIC #<RTS!DTR>,AMRC ;TOGGLE DTR AND RTS TO SETUP COMM PORT
7107 002476 052737 000006 176610 BIS #<RTS!DTR>,AMRC ;
7108 002504 012737 037014 177420 2$: MOV #<AMOD!B9600!CHAR8>,PARAM ;SETUP COMM PARAMETER REGISTER
7109 002512 005737 176610 TST AMRC ;CLEAR DSI BIT IF SET
7110 002516 004737 013270 JSR PC,TIMER1 ;TIMER TO ALLOW PREVIOUS XFERS TO FINISH
7111 002522 122737 000001 001210 CMPB #APTENV,$ENV ;CHECK TO SEE IF APT
7112 002530 001011 BNE 3$ ;IF NOT APT SET KBD INT ENA ACTIVE
7113 002532 005737 001176 TST $PASS ;CHECK IF FIRST PASS
7114 002536 001406 BEQ 3$ ;IF FIRST PASS SET KBD INT ENA
7115 002540 105037 001135 CLRB $INTAG ;SET NON-INTERRUPT MODE INDICATOR
7116 002544 042777 000100 176372 BIC #IE,@$TKS ;CLEAR RECEIVER INT ENABLE FOR KEYBOARD
7117 002552 000406 BR 4$ ;GO CLEAR PASS ERRORS
7118 002554 112737 000001 001135 3$: MOVB #1,$INTAG ;SET INTERRUPT MODE INDICATOR
7119 002562 052777 000100 176354 BIS #IE,@$TKS ;ALLOW KEYBOARD INTERRUPTS
7120 002570 005037 026332 4$: CLR PERR ;PASS ERR COUNT FOR EOP TYPEOUT
7121 002574 005037 026334 CLR PSERR ;PASS SOFT ERR COUNT FOR EOP TYPEOUT
  
```

```
7123 .....  
7124 MEMORY TESTS  
7125 .....  
7126 :MEMORY FROM 0 TO THE LAST LOC OF PROGRAM (LSTAD) IS TESTED FOR TIMEOUT.  
7127 :MEMORY FROM 'LSTAD' TO THE BOTTOM OF THE XXDP MONITOR/ABS LOADER  
7128 :IS TESTED BY A PASS OF A 1010 PATT FOLLOWED BY A PASS OF 0101 PATTERN.  
7129 :IF THE SYSTEM HAS A 28K MEMORY, 28K TO 30K WILL BE TESTED SIMILARLY.  
7130 :.....  
7131  
7132 002600 032737 000020 001246 MEMTST: BIT #BIT4,$DEV# :CHECK IF MEMORY TEST DROPPED  
7133 002606 001402 BEQ 7$ :IF NOT GO DO THE MEMORY TEST  
7134 002610 000137 003244 JMP VTTST :GO CHECK FOR VT100 TESTING  
7135 002614 012737 013420 000004 7$: MOV #INTSRV,ERRVEC :SETUP TIMEOUT TRAP ADDR  
7136 002622 012737 000340 0000J6 MOV #PR7,ERRVEC+2  
7137  
7138 002630 013703 012640 MOV MAXMEM,R3  
7139 002634 162703 000300 SUB #300,R3 :TEST ONLY AS FAR AS ABS LOADER IF NOT XXDP  
7140 002640 005737 000042 TST @#42 :XXDP OR ACT?  
7141 002644 001402 BEQ 5$ :BRANCH IF NOT  
7142 002646 162703 005370 SUB #<1500.*2>-300,R3 :TEST ONLY TO BOTTOM OF XXDP MONITOR  
7143 002652 005004 5$: CLR R4 :ADDR POINTER  
7144 002654 005037 013460 1$: CLR INTFLG  
7145 002660 005714 TST (R4) :TEST LOCS 0 THRU END OF PGM.  
7146 002662 005737 013460 TST INTFLG :TIMEOUT?  
7147 002666 001403 BEQ 2$ :BR IF NO  
7148 002670 010437 003242 MOV R4,ADDR :FOR ERR TYP  
7149 002674 104001 ERROR+1 :TIMEOUT ON READ  
7150 002676 062704 000002 2$: ADD #2,R4 :BUMP ADDR  
7151 002702 020427 031666 CMP R4,#LSTAD :AT END?  
7152 002706 001362 BNE 1$ :BR IF NO  
7153  
7154 002710 005037 003234 CLR MEMCT  
7155 002714 012737 125252 003240 MOV #125252,PAT :DATA PATT FOR 1'ST PASS  
7156 002722 012704 031666 8$: MOV #LSTAD,R4 :ADDR POINTER. R/W LOCS LSTAD THRU  
7157 :BOT OF RT11 OR BOT OF ABS LOADER  
7158 002726 005037 013460 3$: CLR INTFLG  
7159 002732 013714 003240 MOV PAT,(R4) :WRITE  
7160 002736 005737 013460 TST INTFLG :TIMEOUT?  
7161 002742 001403 BEQ 4$ :BR IF NO  
7162 002744 010437 003242 MOV R4,ADDR :FOR ERR TYP  
7163 002750 104002 ERROR+2 :TIMEOUT ON WRITE  
7164 002752 011437 003236 4$: MOV (R4),MEMHLD :READ  
7165 002756 023737 003236 003240 CMP MEMHLD,PAT :COMPARE  
7166 002764 001403 BEQ 6$  
7167 002766 010437 003242 MOV R4,ADDR :FOR ERR TYP  
7168 002772 104003 ERROR+3 :COMPARE ERROR  
7169  
7170 002774 062704 000002 6$: ADD #2,R4 :BUMP POINTER  
7171 003000 020403 CMP R4,R3 :AT BOT OF ABS LOADER?  
7172 003002 001351 BNE 3$ :BR IN NO  
7173 003004 005737 003234 TST MEMCT  
7174 003010 001006 BNE 9$  
7175 003012 005237 003234 INC MEMCT  
7176 003016 012737 052525 003240 MOV #052525,PAT :DATA PATT FOR 2'ND PASS  
7177 003024 000736 BR 8$ :REPEAT  
7178
```

```

7179 003026 023727 012640 157776 9$:   CMP      MAXMEM,#157776 ;DO WE HAVE 28K?
7180 003034 001047          BNE      15$           ;BR IF NO & EXIT
7181
7182 003036 012737 125252 003240          MOV      #125252,PAT ;DATA PATT FOR 1'ST PASS OF HI MEM
7183 003044 012704 160000 10$:   MOV      #160000,R4 ;ADDR PTR TO TOP OF 28K
7184 003050 005037 013460 11$:   CLR      INTFLG
7185 003054 013714 003240          MOV      PAT,(R4)    ;WRITE
7186 003060 005737 013460          TST      INTFLG
7187 003064 001403          BEQ      12$
7188 003066 010437 003242          MOV      R4,ADDR    ;FOR ERR TYP
7189 003072 104004          ERROR+4 ;TIMEOUT
7190 003074 011437 003236 12$:   MOV      (R4),MEMHLD ;READ
7191 003100 023737 003236 003240          CMP      MEMHLD,PAT ;COMPARE
7192 003106 001403          BEQ      14$
7193 003110 010437 003242          MOV      R4,ADDR    ;FOR ERR TYP
7194 003114 104005          ERROR+5 ;COMPARE ERROR
7195
7196 003116 062704 000002 14$:   ADD      #2,R4
7197 003122 020427 167776          CMP      R4,#167776 ;AT END OF 30K?
7198 003126 001350          BNE      11$         ;BR IF NO
7199
7200 003130 005237 003234          INC      MEMCT
7201 003134 023727 003234 000003          CMP      MEMCT,#3
7202 003142 001404          BEQ      15$
7203 003144 012737 052525 003240          MOV      #052525,PAT ;DATA PATT FOR 2'ND HALF OF HI MEM
7204 003152 000734          BR       10$        ;REPEAT
7205
7206 003154 005037 013460 15$:   CLR      INTFLG    ;CLEAR FLAG TO DETFCT INTERRUPT
7207 003160 005737 170000          TST      @#170000 ;TEST ROM SPACE
7208 003164 005737 013460          TST      INTFLG    ;WAS THERE A TRAP?
7209 003170 001401          BEQ      16$        ;BRANCH IF NOT
7210 003172 104027          ERROR+27 ;ROM NOT PRESENT
7211
7212 003174 012737 000006 000004 16$:   MOV      #ERRVEC+2,ERRVEC ;RESTORE TIMEOUT VECTOR
7213 003202 005037 000006          CLR      ERRVEC+2
7214
7215 003206 032777 010000 175724          BIT      #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
7216 003214 001413          BEQ      VTTST
7217 003216 104401 017057          TYPE    ,MEMORY
7218 003222 104401 017074          TYPE    ,DUN
7219 003226 004737 013376          JSR      PC,GFLCHK ;CHECK IF ^G TYPED WHILE TYPING
7220 003232 000404          BR       VTTST
7221
7222 003234 000000          MEMCT: 0 ;0 = MEM FROM LSTAD TO MONITOR/ABS LOADER W/ 125252 PATT
7223 ;1 = SAME WITH 052525 PATT
7224 ;1 = MEM FROM 28K TO 30K WITH 125252 PAT
7225 ;2 = SAME WITH 052525 PATT
7226 ;3 = EXIT
7227
7228 003236 000000          MEMHLD: 0 ;PUT MEMORY READ HERE
7229 003240 000000          PAT: 0 ;PATT TO BE WRITTEN & READ
7230 003242 000000          ADDR: 0 ;ADDR UNDER TEST
    
```

```

7232          ;:*****
7233          ;:          START VT100
7234          ;:*****
7235
7236 003244 032737 000100 00'246 VTTST: BIT      #BIT6,$DEV  ;DROP TEST?
7237 003252 001142          BNE      CLKTST  ;BRANCH IF YES
7238 003254 122737 000001 001210 CMPB    #APTENV,@#SENV ;CHECK IF ON APT
7239 003262 001003          BNE      7$      ;BR IF NOT
7240 003264 005737 001176          TST     $PASS      ;CHECK IF FIRST PASS
7241 003270 001133          BNE      CLKTST  ;CAN ONLY RUN DISPLAY TESTS THE FIRST PASS
7242 003272 105037 001254 1$: CLRB    VTDONE   ;FLAG THAT DISPLAY PROCESS HAS NOT COMPLETED
7243 003276 004537 012666          JSR     R5,VITALK  ;SEND AND RECEIVE AN ESCAPE SEQUENCE
7244 003302 016331          STATRQ   ;TEXT TO SEND REQUEST DEVICE STATUS
7245 003304 016344          DSTATR   ;TEXT TO EXPECT--DEVICE STATUS REPORT
7246 003306 000524          BR      CLKTST  ;RETURN FOR XMITTER HUNG
7247 003310 000413          BR      2$      ;RETURN FOR RECEIVER HUNG OR UNRECOGNIZED RESPONSE
7248 003312 122737 000060 014012 CMPB    #'0,OPTCHR ;TEST THE OPTION CHARACTER THAT WAS RECEIVED
7249 003320 001407          BEQ     2$      ;BRANCH IF IT INDICATED OK STATUS
7250 003322 122737 000063 014012 CMPB    #'3,OPTCHR ;CHECK IF IT INDICATED BAD STAUTS
7251 003330 001402          BEQ     1$      ;BRANCH IF IT DID
7252 003332 104032          ERROR+32 ;UNRECOGNIZED ESCAPE SEQUENCE FOR STATUS REPORT
7253 003334 000401          BR      2$
7254 003336 104030          1$: ERROR+30 ;VT100 REPORTED SELF-TEST FAILURE
7255 003340 012737 016351 003372 2$: MOV     #TIDNTF,9$ ;SETUP FOR VT100 IDENTIFY SEQUENCE
7256 003346 032777 001000 175564 BIT     #BIT9,@SWR  ;CHECK IF VT132 OR VT100
7257 003354 001403          BEQ     8$      ;IF 0 THEN VT100
7258 003356 012737 016361 003372 MOV     #VIDNTF,9$ ;SETUP FOR VT132 IDENTIFY SEQUENCE
7259 003364 004537 012666          8$: JSR     R5,VITALK  ;INITIATE ANOTHER FESCAPE SEQUENCE
7260 003370 016340          IDENRQ   ;TEXT TO SEND--REQUEST TERMINAL ID
7261 003372 016351          9$: TIDNTF  ;EXPECTED TEXT--TERMINAL IDENTIFY
7262 003374 000471          BR      CLKTST  ;XMITTER HUNG RETURN
7263 003376 000435          BR      6$      ;RECEIVER HUNG OR UNRECOGNIZED RESPONSE
7264 003400 122737 000060 014012 CMPB    #'0,OPTCHR ;CHECK IF THE OPTION CHAR > ASCII 0
7265 003406 101004          BHI     3$      ;BRANCH IF NOT
7266 003410 122737 000067 014012 CMPB    #'7,OPTCHR ;CHECK IF IT WAS < ASCII 7
7267 003416 103002          BHIS   4$      ;BRANCH IF IT WAS (WITHIN RANGE)
7268 003420 104032          3$: ERROR+32 ;UNRECOGNIZED ESCAPE SEQUENCE FOR TERMINAL IDENTIFY
7269 003422 000423          BR      6$
7270 003424 132737 000001 014012 4$: BITB   #BIT0,OPTCHR ;MAKE SURE OPTION PRESENT WAS ASSERTED (STP)
7271 003432 001005          BNE     5$      ;BRANCH IF IT WAS
7272 003434 042737 177770 014012 BIC     #'^C7,OPTCHR ;MAKE CHARACTER NON-ASCII FOR ERROR REPORT
7273 003442 104031          ERROR+31 ;OPTION PRESENT (STP) NOT ASSERTED TO VT100
7274 003444 000412          BR      6$
7275 003446 005737 001176          5$: TST     $PASS      ;CHECK IF FIRST PASS
7276 003452 001007          BNE     6$      ;BRANCH IF NOT
7277 003454 113737 014012 016430 MOVB    OPTCHR,OPCTX ;MOVE THE ASCII CHARACTER INTO TEXT STRING
7278 003462 104401 016371          TYPE   ,OPTPRE  ;REPORT THE OPTION PRESENT PARAMETER
7279 003466 004737 013376          JSR    PC,GFLCHK  ;CHECK IF ^G TYPED DURING TYPEOUT
7280 003472 012737 014114 000064 6$: MOV     #TPSRV,TPVEC ;SET UP DISPLAY PROCESS INTERRUPT VECTOR
7281 003500 012737 000340 000066 MOV     #PR7,TPVEC+2 ;SET PRIORITY
7282 003506 012737 021470 014346 MOV     #DSPTXT,TXTPTR ;SET UP POINTER TO DISPLAY TEXT
7283 003514 105037 001250          CLRB   TYPFLG   ;FLAG THAT DISPLAY PROCESS IS ACTIVE
7284 003520 052737 000200 001266 BIS     #BIT7,DEVACT ;SET VT100 ACTIVE FLAG
7285 003526 052777 000100 175414 BIS     #IE,@$TPS  ;SET XMIT INTERRUPT ENABLE
7286          ; (RCVR IE ALREADY ON)
7287 003534 032777 010000 175376 BIT     #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
    
```

7288 003542 001406
7289 003544 104407
7290 003546 104401 016516
7291 003552 104401 017045
7292 003556 104410

BEQ CLKTST
SAVEVT
TYPE .DSPTST
TYPE .RUN
RESTVT

;ALLOW 2 TYPEOUTS ON SAME LINE

;RESTORE VT100 DISPLAY

```

7294
7295 .
7296
7297
7298 003560 032737 000200 001246 CLKTST: BIT #BIT7,$DEVH ;DROP TEST?
7299 003566 001045 BNE PRST ;:BR IF YES
7300 003570 122737 000001 001210 CMPB #APTENV,@#SENV ;CHECK IF APT
7301 003576 001003 BNE 1$ ;IF NOT APT DO THE TEST
7302 003600 005737 001176 TST $PASS ;APT - CHECK IF FIRST PASS
7303 003604 001036 BNE PRST ;:IF NOT SKIP THE TEST
7304 003606 012737 013420 000100 1$: MOV #INTSRV,CLKVEC ;SETUP VECTOR AREA
7305 003614 012737 000200 000102 MOV #200,CLKVEC+2 ;LOCKOUT OTHER INTER.
7306 003622 005037 013460 CLR INTFLG
7307 003627 052737 000100 177544 BIC #IE,CLK ;SET CLOCK LOOSE!
7308
7309 003634 004537 013156 ISR R5,TIMER ;SEE IF INTFLG GOES TO 100
7310 003640 013460 INTFLG
7311 003642 000100 100
7312 003644 104015 ERROR+15 ;CLK NOT RESPONDING
7313 003646 000415 BR PRST
7314 003650 052737 000100 001266 BIS #BIT6,DEVACT ;SET CLOCK ACTIVE FLAG
7315
7316 003656 032777 010000 175254 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
7317 003664 001406 BEQ PRST
7318 003666 104407 SAVEVT ;ALLOW 2 TYPEOUTS ON SAME LINE
7319 003670 104401 016571 TYPE ,CLOCK
7320 003674 104401 017045 TYPE ,RUN
7321 003700 104410 RESTVT ;RESTORE VT100 DISPLAY
  
```

```
7323 .....  
7324 : START PRINTER  
7325 : .....  
7326  
7327 003702 005737 012614 PRTST: TST PRPRES ;THERE?  
7328 003706 001465 BEQ SMTST ;:BR IF NO  
7329 003710 032737 100000 001246 BIT #BIT15,$DEVN ;:DROP TEST?  
7330 003716 001061 BNE SMTST ;:BR IF YES  
7331  
7332 003720 012737 023414 177420 MOV #<PRT!B1200!CHAR8>,PARAM ;SETUP BAUD RATE & CHAR LENGTH  
7333 003726 012737 014350 000200 MOV #PRSRV,PRVEC ;ELSE SETUP PRINTER VECTOR  
7334 003734 012737 000200 000202 MOV #200,PRVEC+2 ;LOCKOUT INT ERR  
7335  
7336 003742 005037 014514 CLR LINCT ;CLEAR LINE CTR  
7337  
7338 003746 012737 000015 014512 MOV #CR,PRCHR ;CHAR TO BE PRINTED  
7339 7339 003754 052737 000100 177514 BIS #IE,PRS ;START WITH 'DEL' (177) IF IT IS NEEDED TO CLEAR BUFF  
7340 ;SET PRINTER LOOSE!  
7341 ;WILL PRINT CONTINUOUS LINES  
7342 ;EA BEGINNING WITH ASCII 40 THRU 176  
7343 003762 004537 013156 JSR RS,TIMER ;SEE IF LINCT GOES TO 5  
7344 003766 014514 LINCT  
7345 003770 000005 5  
7346 003772 104016 ERROR+16 ;PRINTER NOT RESPONDING  
7347 003774 000432 BR SMTST  
7348  
7349 003776 005737 012616 TST PRPORT ;CHECK FOR EXTERNAL LOOP-BACK  
7350 004002 001404 BEQ 2$ ;IF NOT THEN CONTINUE  
7351 004004 042737 000100 177514 BIC #IE,PRS ;EXTERNAL - SHUT DOWN PRINTER  
7352 004012 000403 BR 3$ ;GO REPORT PRINTER DONE  
7353 004014 052737 000040 001266 2$: BIS #BITS,DEVACT ;SET PRINTER ACTIVE FLAG  
7354 004022 032777 010000 175110 3$: BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET  
7355 004030 001414 BEQ SMTST  
7356 004032 104407 SAVEVI ;ALLOW 2 TYPEOUTS ON SAME LINE  
7357 004034 104401 016561 TYPE ,PRINT  
7358 004040 005737 012616 TST PRPORT ;CHECK FOR EXTERNAL LOOP-BACK  
7359 004044 001403 BEQ 4$ ;IF NOT THEN CONTINUE  
7360 004046 104401 016605 TYPE ,PORT ;PRINT TERMINATION REPORT  
7361 004052 000402 BR 5$ ;RESTORE CURSOR AND DO NEXT TEST  
7362 004054 104401 017045 4$: TYPE ,RUN  
7363 004060 104410 5$: RESTVT ;RESTORE VT100 DISPLAY
```

```

7365          ;:*****
7366          ;: START SYNC COMM PORT
7367          ;:*****
7368
7369 004062 032737 002000 001246 SMTST: BIT    #BIT10,$DEV    ;DROP TEST?
7370 004070 001007                BNE     1$          ;BRANCH IF YES
7371 004072 122737 000001 001210 CMPB   #APTENV,$ENV  ;CHECK IF ON APT
7372 004100 001005                BNE     NOAPTS      ;BRANCH IF NOT
7373 004102 005737 001176                TST    $PASS       ;FIRST PASS?
7374 004106 001402                BEQ    NOAPTS      ;DO TEST IF 1ST PASS
7375 004110 000137 004642 1$:      JMP    AMTST       ;SKIP THE TEST
7376
7377 004114 005037 015342                NOAPTS: CLR   LSTCHR   ;CLEAR LAST CHARACTER FLAG FOR XMIT
7378 004120 052737 000400 176624      BIS   #MR,$MXC    ;MASTER RESET
7379 004126 052737 004000 176624      BIS   #MCLK,$MXC  ;MAINT CLOCK
7380
7381 004134 012737 007026 176622      MOV   #<CHRB!ODDPAR!026>,$SMPAR ;SETUP SYNC PARAMETER REGISTER
7382 004142 032737 000010 001246      BIT   #BIT3,$DEV  ;CHECK FOR EXTERNAL LOOPBACK
7383 004150 001407                BEQ    1$          ;IF 0 DO SYNC COMM MODEM CHECK
7384 004152 052737 014000 176624      BIS   #<MM!MCLK>,$SMXC ;IF 1 SET MAINT MODE FOR INT LOOPBACK
7385 004160 052737 000026 176620      BIS   #<RTS!DTR!SR SYN>,$SMRC ;SET SEARCH SYNC
7386 004166 000526                BR    2$          ;GO DO THE TEST
7387
7388 004170 012737 122006 004640 1$:   MOV   #<DSC!CTS!SREC!RTS!DTR>,$EXPMRC ;SETUP EXPECTED RESULTS
7389 004176 005737 176620                TST    $SMRC      ;CLEAR ANY FLAGS THAT MIGHT BE SET
7390 004202 012737 000006 176620      MOV   #<RTS!DTR>,$SMRC ;SET DTR AND RTS
7391 004210 013737 176620 004636      MOV   $SMRC,$SAVMRC ;READ CSR REGISTER AND SAVE
7392 004216 023737 004636 004640      CMP   $SAVMRC,$EXPMRC ;CHECK RESULTS TO BE EXPECTED
7393 004224 001402                BEQ    7$          ;IF EQUAL THEN CONTINUE
7394 004226 104033      ERROR+33 ;SYNC COMM MODEM SIGNAL ERROR
7395 004230 000574                BR    6$          ;EXIT THE TEST
7396 004232 042737 100000 004640 7$:   BIC   #DSC,$EXPMRC ;CLEAR EXPECTED DSC BIT FOR 2ND READ
7397 004240 013737 176620 004636      MOV   $SMRC,$SAVMRC ;READ SYNC COMM CSR REGISTER
7398 004246 023737 004636 004640      CMP   $SAVMRC,$EXPMRC ;CHECK THAT DSC WENT TO 0
7399 004254 001402                BEQ    8$          ;IF EQUAL THEN CONTINUE
7400 004256 104033      ERROR+33 ;SYNC COMM MODEM SIGNAL ERROR
7401 004260 000560                BR    6$          ;EXIT THE TEST
7402 004262 012737 151012 004640 8$:   MOV   #<DSC!RING!CDET!DSRDY!SXMIT!DTR>,$EXPMRC ;SETUP EXPECTED
7403 004270 012737 000012 176620      MOV   #<SXMIT!DTR>,$SMRC ;SET SECONDARY XMIT AND DTR
7404 004276 013737 176620 004636      MOV   $SMRC,$SAVMRC ;READ SYNC COMM CSR AND SAVE
7405 004304 023737 004636 004640      CMP   $SAVMRC,$EXPMRC ;CHECK RESULTS
7406 004312 001402                BEQ    9$          ;IF EQUAL THEN CONTINUE
7407 004314 104033      ERROR+33 ;SYNC COMM MODEM SIGNAL FAILURE
7408 004316 000541                BR    6$          ;EXIT THE TEST
7409 004320 042737 100000 004640 9$:   BIC   #DSC,$EXPMRC ;CLEAR EXPECTED DSC BIT
7410 004326 013737 176620 004636      MOV   $SMRC,$SAVMRC ;READ SYNC COMM CSR REGISTER
7411 004334 023737 004636 004640      CMP   $SAVMRC,$EXPMRC ;CHECK THE RESULTS
7412 004342 001402                BEQ    10$         ;CONTINUE IF EQUAL
7413 004344 104033      ERROR+33 ;SYNC COMM MODEM SIGNAL FAILURE
7414 004346 000525                BR    6$          ;EXIT THE TEST
7415 004350 012737 100002 004640 10$:  MOV   #<DSC!DTR>,$EXPMRC ;SETUP EXPECTED RESULTS
7416 004356 012737 000002 176620      MOV   #DTR,$SMRC  ;SET DATA TERMINAL READY
7417 004364 013737 176620 004636      MOV   $SMRC,$SAVMRC ;READ COMM CSR REGISTER
7418 004372 023737 004636 004640      CMP   $SAVMRC,$EXPMRC ;CHECK RESULTS TO EQUAL EXPECTED
7419 004400 001402                BEQ    11$         ;IF EQUAL THEN CONTINUE
7420 004402 104033      ERROR+33 ;SYNC COMM MODEM SIGNAL FAILURE

```

```

7421 004404 000506          BR      6$          ;EXIT THE TEST
7422
7423 004406 122737 000001 001210 11$:  CMPB   #APTENV,$ENV      ;CHECK IF ON APT
7424 004414 001006          BNE    12$          ;IF NOT LEAVE DTR SET ONLY
7425 004416 012737 000026 176620  MOV    #<DTR!RTS!SRSYN>,SMRC ;SET APT EXTERNAL LOOPBACK
7426 004424 005737 176620  TST    SMRC          ;CLEAR DSC BIT
7427 004430 000405          BR     2$          ;GO START THE TEST
7428 004432 012737 000022 176620 12$:  MOV    #<DTR!SRSYN>,SMRC ;SET NON-APT EXTERNAL LOOP SIGNAL
7429 004440 005737 176620  TST    SMRC          ;CLEAR DSC BIT IF SET
7430
7431 004444 052737 000020 176624 2$:  BIS    #SEND,SMXC      ;ENABLE XMIT SEND
7432 004452 012737 000026 176626  MOV    #026,SMXB      ;XMIT SYNC CHAR
7433
7434 004460 004537 013304          JSR    R5,TIMER2      ;WAIT FOR DONE
7435 004464 176620          SMRC
7436 004466 000200          DONE
7437 004470 104026          ERROR+26           ;NO DONE
7438 004472 000453          BR     6$          ;EXIT
7439
7440 004474 013737 176622 015246  MOV    SMRB,COMHLD    ;READ WORD
7441 004502 023727 015246 000026  CMP    COMHLD,#026    ;IS IT SYNC CHAR?
7442 004510 001402          BEQ    4$          ;BR IF YES
7443 004512 104006          ERROR+6           ;DID NOT REC SYNC CHAR
7444 004514 000424          BR     5$          ;EXIT
7445
7446 004516 004537 012642          4$:  JSR    R5,SETVEC      ;SETUP VECTOR AREA
7447 004522 000340          SMRVEC
7448 004524 015344          SMRSRV
7449 004526 015252          SMXSRV
7450
7451 004530 012737 000027 015244  MOV    #027,COMCHR    ;1'ST CHAR TO BE XMITTED
7452 004536 052737 000100 176620  BIS    #IE,SMRC      ;SET SYNC MODEM LOOSE!
7453 004544 052737 000100 176624  BIS    #IE,SMXC
7454
7455 004552 004537 013156          JSR    R5,TIMER      ;SEE IF COMCHR GOES TO ALL 1'S (DONE)
7456 004556 015244          COMCHR
7457 004560 177777          177777
7458 004562 104022          ERROR+22           ;SYNCH COMM NOT RESPONDING
7459 004564 000240          NOP
7460
7461 004566 012737 000400 176624 5$:  MOV    #MR,SMXC      ;MASTER RESET
7462 004574 032777 010000 174336  BIT    #BIT12,@SWR    ;DO TYPEOUT IF SET
7463 004602 001417          BEQ    AMTST        ;EXIT IF NOT
7464 004604 104407          SAVEVT            ;ALLOW 2 TYPEOUTS ON SAME LINE
7465 004606 104401 016534          TYPE    ,SCOM
7466 004612 104401 017074          TYPE    ,DUN
7467 004616 104410          RESTVT
7468 004620 000410          BR     AMTST        ;RESTORE VT100 DISPLAY
7469
7470 004622 012737 000400 176624 6$:  MOV    #MR,SMXC      ;MASTER RESET
7471 004630 005737 176620  TST    SMRC
7472 004634 000402          BR     AMTST        ;GO CHECK FOR ASYNC TEST
7473
7474 004636 000000          SAVMRC: .WORD 0    ;COMM RCVR CSR SAVED HERE
7475 004640 071000          EXPMRC: .WORD DSRDY!RING!CTS!CDET

```

```
7477 ;:*****
7478 ;: START ASYNC COMM PORT
7479 ;:*****
7480
7481 004642 032737 004000 001246 AMTST: BIT #BIT11,$DEV# ;DROP TEST?
7482 004650 001007 BNE 8$ ;IF YES THEN SKIP TEST
7483 004652 122737 000001 001210 CMPB #APTENV,$ENV ;CHECK IF ON APT
7484 004660 001005 BNE 2$ ;BRANCH IF NOT
7485 004662 005737 001176 TST $PASS ;BYPASS TEST AFTER FIRST PASS ON APT
7486 004666 001402 BEQ 2$
7487 004670 000137 005314 8$: JMP CLITST ;GO CHECK FOR CLUSTER OPTION
7488 ;
7489
7490 004674 032737 000010 001246 2$: BIT #BIT3,$DEV# ;CHECK FOR EXTERNAL LOOPBACK
7491 004702 001414 BEQ 1$ ;IF EXTERNAL GO DO MODEM TESTS
7492 004704 042737 000006 176610 BIC #<RTS!DTR>,$AMRC ;TOGGLE DTR AND RTS TO SETUP INTERNAL LOOP
7493 004712 052737 000006 176610 BIS #<RTS!DTR>,$AMRC ;
7494 004720 005737 176610 TST $AMRC ;CLEAR ANY BITS THAT MAY HAVE SET
7495 004724 012737 037036 177420 MOV #<AMOD!B9600!OPAR!CHAR8!MAINT>,$PARAM ;SETUP PARAMETERS
7496 004732 000523 BR TSTAM ;GO TEST ASYNC COMM PORT
7497
7498 004734 012737 037034 177420 1$: MOV #<AMOD!B9600!OPAR!CHAR8>,$PARAM ;SETUP PARAMETERS FOR EXT
7499 004742 012737 122006 004640 MOV #<DSI!CTS!SREC!RTS!DTR>,$EXPMRC ;SETUP EXPECTED CSR REG
7500 004750 005737 176610 TST $AMRC ;CLEAR ANY BITS THAT MAY BE SET
7501 004754 012737 000006 176610 MOV #<RTS!DTR>,$AMRC ;SET RTS AND DTR IN CSR REGISTER
7502 004762 013737 176610 004636 MOV $AMRC,$SAVMRC ;SAVE ASYN CSR REGISTER
7503 004770 023737 004636 004640 CMP $SAVMRC,$EXPMRC ;CHECK TO SEE IF EQUAL
7504 004776 001402 BEQ 3$ ;IF EQUAL THEN CONTINUE
7505 005000 104035 ERROR+35 ;ASYNC COMM MODEM FAILURE
7506 005002 000544 BR CLITST ;EXIT TO CLUSTER TEST
7507 005004 042737 100000 004640 3$: BIC #DSI,$EXPMRC ;CLEAR EXPECTED DSI BIT FOR 2ND READ
7508 005012 013737 176610 004636 MOV $AMRC,$SAVMRC ;READ ASYNC COMM AND SAVE
7509 005020 023737 004636 004640 CMP $SAVMRC,$EXPMRC ;CHECK THAT DSI WENT AWAY
7510 005026 001402 BEQ 4$ ;IF EQUAL THEN CONTINUE
7511 005030 104035 ERROR+35 ;ASYNC COMM MODEM FAILURE
7512 005032 000530 BR CLITST ;GO CHECK FOR CLUSTER
7513 005034 012737 151012 004640 4$: MOV #<DSI!RING!CDET!DSRDY!SXMIT!DTR>,$EXPMRC ;SETUP EXPECTED CSR
7514 005042 012737 000012 176610 MOV #<SXMIT!DTR>,$AMRC ;SET SEC XMIT AND DTR
7515 005050 013737 176610 004636 MOV $AMRC,$SAVMRC ;READ ASYNC COMM CSR AND SAVE
7516 005056 023737 004636 004640 CMP $SAVMRC,$EXPMRC ;CHECK TO SEE IF EQUAL
7517 005064 001402 BEQ 5$ ;IF EQUAL THEN CONTINUE
7518 005066 104035 ERROR+35 ;ASYNC COMM MODEM FAILURE
7519 005070 000511 BR CLITST ;GO CHECK FOR CLUSTER TERMINALS
7520 005072 042737 100000 004640 5$: BIC #DSI,$EXPMRC ;CLEAR DSI IN EXPECTER CSR
7521 005100 013737 176610 004636 MOV $AMRC,$SAVMRC ;READ ASYNC COMM CSR AND SAVE
7522 005106 023737 004636 004640 CMP $SAVMRC,$EXPMRC ;CHECK TO SEE IF EQUAL
7523 005114 001402 BEQ 6$ ;IF EQUAL THEN CONTINUE
7524 005116 104035 ERROR+35 ;ASYNC COMM MODEM FAILURE
7525 005120 000475 BR CLITST ;GO CHECK FOR CLUSTER TERMINALS
7526 005122 012737 100002 004640 6$: MOV #<DSI!DTR>,$EXPMRC ;SETUP EXPECTED ASYNC COMM CSR
7527 005130 012737 000002 176610 MOV #DTR,$AMRC ;SET DATA TERMINAL READY BY ITSELF
7528 005136 013737 176610 004636 MOV $AMRC,$SAVMRC ;READ ASYN COMM CSR AND SAVE IT
7529 005144 023737 004636 004640 CMP $SAVMRC,$EXPMRC ;CHECK TO SEE IF EQUAL
7530 005152 001402 BEQ 7$ ;IF EQUAL THEN CONTINUE
7531 005154 104035 ERROR+35 ;ASYNC COMM MODEM FAILURE
7532 005156 000456 BR CLITST ;GO CHECK FOR CLUSTER TERMINALS
```

```
7533
7534 005160 122737 000001 001210 7$: CMPB #APTENV,$ENV ;CHECK TO SEE IF APT
7535 005166 001003 BNE 9$ ;IF NOT CONTINUE
7536 005170 012737 000006 176610 MOV #<RTS!DTR>,AMRC ;SET APT EXTERNAL LOOPBACK BIT
7537 005176 005737 176610 9$: TST AMRC ;ISSUED TO CLEAR DSI BIT
7538
7539 005202 004537 012642 TSTAM: JSR R5,SETVEC ;SETUP VECTOR AREA
7540 005206 000330 AMRVEC
7541 005210 015112 AMRSRV
7542 005212 015074 AMXSRV
7543
7544 005214 005037 015244 CLR COMCHR ;CHAR TO BE XMITTED
7545 005220 005037 015250 CLR COMCNT ;CLEAR COUNT OF CHARACTER PASSES
7546 005224 013737 176612 015246 MOV AMRB,COMHLD ;CLR OUT ANY PREVIOUS STORED WORD
7547 005232 052737 000100 176610 BIS #IE,AMRC ;SET ASYNC MODEM LOOSE!
7548 005240 052737 000100 176614 BIS #IE,AMXC
7549
7550 005246 004537 013156 JSR R5,TIMER ;SEE IF COMCNT GOES TO 1
7551 005252 015250 COMCNT
7552 005254 000001 1
7553 005256 104023 ERROR+23 ;ASYNC COMM NOT RESPONDING
7554 005260 000415 BR CL1TST
7555 005262 052737 000020 001266 BIS #BIT4,DEVACT ;SET ASYNC COMM ACTIVE FLAG
7556
7557 005270 032777 010000 173642 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
7558 005276 001406 BEQ CL1TST
7559 005300 104407 SAVEVT ;ALLOW 2 TYPEOUTS ON SAME LINE
7560 005302 104401 016546 TYPE ,ACOM
7561 005306 104401 017045 TYPE ,RUN
7562 005312 104410 RESTVT ;RESTORE VT100 DISPLAY
7563
7564 ;:*****
7565 ;: START CLUSTER TERMINAL #1
7566 ;:*****
7567
7568 005314 005737 012612 CL1TST: TST CLPRES ;OPTION PRESENT?
7569 005320 001002 BNE 1$ ;:BR IF YES
7570 005322 000137 006074 JMP TUTST ;GO CHECK FOR TUS8
7571
7572 005326 032737 000010 001246 1$: BIT #BIT3,$DEVM ;CHECK FOR INT.EXT LOOPBACK
7573 005334 001404 BEQ 2$ ;IF EXTERNAL GO SET DTR
7574 005336 052737 000006 176610 BIS #<RTS!DTR>,AMRC ;INTERNAL SET RTS AND DTR
7575 005344 000403 BR 3$ ;GO SEE IF TUS8 ACTIVE
7576 005346 052737 000002 176610 2$: BIS #DTR,AMRC ;EXTERNAL - ONLY SET DTR
7577
7578 005354 012737 002416 014636 3$: MOV #<B300!CHAR8!MAINT>,CTPARAM ;300 BAUD
7579 005362 032737 010000 001246 BIT #BIT12,$DEVM ;DROP TEST ?
7580 005370 001063 BNE CL2TST ;BRANCH IF YES
7581 005372 013700 014636 MOV CTPARM,RO ;GET CLUSTER PARAMETERS
7582 005376 052700 000000 BIS #CT1,RO ;SELECT CLUSTER TERMINAL 1
7583 005402 032737 000001 001246 BIT #BIT0,$DEVM ;CHECK IF INTERNAL LOOPBACK
7584 005410 001002 BNE 4$ ;IF INTERNAL GO LOAD PARAMETER REGISTER
7585 005412 042700 000002 BIC #MAINT,RO ;EXTERNAL CLEAR INTERNAL LOOPBACK BIT
7586 005416 010037 177420 4$: MOV RO,PARAM ;SETUP CLUSTER PARAMETER 1
7587
7588 005422 004537 012642 JSR R5,SETVEC ;SETUP INTERR VECTORS
```

7589	005426	000300				TK1VEC		
7590	005430	014536				TK1SRV		
7591	005432	014520				TP1SRV		
7592								
7593	005434	005037	014630			CLR	T1CHR	;CHAR TO BE XMITTED
7594	005440	005037	014634			CLR	T1CNT	;CLEAR COUNT OF CHARACTER PASSES
7595	005444	013737	176502	014632		MOV	TK1B,T1HLD	;READ CHAR (CLEAR STALE DATA)
7596	005452	052737	000100	176500		BIS	#IE,TK1S	;SET CLUSTER TERM #1 LOOSE!
7597	005460	052737	000100	176504		BIS	#IE,TP1S	
7598								
7599	005466	004537	013156			JSR	R5,TIMER	;SEE IF T1CNT GOES TO 1
7600	005472	014634				T1CNT		
7601	005474	000001				1		
7602	005476	104017				ERROR+17		;CLUSTER TERM 1 NOT RESPONDING
7603	005500	000417				BR	CL2TST	
7604	005502	052737	000010	001266		BIS	#BIT3,DEVACT	;SET CLUSTER TERMINAL 1 ACTIVE FLAG
7605								
7606	005510	032777	010000	173422		BIT	#BIT12,@SWR	;SKIP TYPEOUT IF NOT SET
7607	005516	001410				BEQ	CL2TST	
7608	005520	104407				SAVEVT		;ALLOW 3 TYPEOUTS ON SAME LINE
7609	005522	104401	016465			TYPE	,CLTERM	
7610	005526	104401	016510			TYPE	,T1	
7611	005532	104401	017045			TYPE	,RUN	
7612	005536	104410				RESTVT		;RESTORE VT100 DISPLAY

```

7614 ;:*****
7615 : START CLUSTER TERMINAL #2
7616 ;:*****
7617
7618 005540 032737 020000 001246 CL2TST: BIT #BIT13,$DEV  ;DROP TEST?
7619 005546 001063 BNE CL3TST ;:BR IF YES
7620
7621 005550 013700 014636 MOV CTPARM,RO ;GET THE CLUSTER PARAMETER
7622 005554 052700 050000 BIS #CT2,RO ;SELECT CLUSTER TERMINAL 2
7623 005560 032737 000002 001246 BIT #BIT1,$DEV ;CHECK IF INTERNAL LOOPBACK
7624 005566 001002 BNE 1$ ;IF INTERNAL GO LOAD PARAMETER
7625 005570 042700 000002 BIC #MAINT,RO ;EXTERNAL CLEAR INTERNAL LOOPBACK BIT
7626 005574 010037 177420 1$: MOV RO,PARAM ;SETUP CLUSTER TERMINAL 2
7627
7628 005600 004537 012642 JSR R5,SETVEC ;SETUP INTERR VECTORS
7629 005604 000310 TK2VEC
7630 005606 014656 TK2SRV
7631 005610 014640 TP2SRV
7632
7633 005612 005037 014750 CLR T2CHR ;CHAR TO BE XMITTED
7634 005616 005037 014754 CLR T2CNT ;CLEAR COUNT OF CHARACTER PASSES
7635 005622 013737 176512 014752 MOV TK2B,T2HLD ;READ CHAR (CLEAR STALE DATA)
7636 005630 052737 000100 176510 BIS #IE,TK2S ;SET CLUSTER TERM #2 LOOSE!
7637 005636 052737 000100 176514 BIS #IE,TP2S
7638
7639 005644 004537 013156 JSR R5,TIMER ;SEE IF T2CNT GOES TO 1
7640 005650 014754 T2CNT
7641 005652 000001 1
7642 005654 104020 ERROR+20 ;CLUSTER TERM 2 NOT RESPONDING
7643 005656 000417 BR CL3TST
7644 005660 052737 000004 001266 BIS #BIT2,DEVACT ;SET CLUSTER TERMINAL 2 ACTIVE FLAG
7645
7646 005666 032777 010000 173244 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
7647 005674 001410 BEQ CL3TST
7648 005676 104407 SAVEVT ;ALLOW 3 TYPEOUTS ON SAME LINE
7649 005700 104401 016465 TYPE ,CLTERM
7650 005704 104401 016512 TYPE ,T2
7651 005710 104401 017045 TYPE ,RUN
7652 005714 104410 RESTVT ;RESTORE VT100 DISPLAY
  
```

```

7654      ;:*****
7655      ;:          START CLUSTER TERMINAL #3
7656      ;:*****
7657
7658 005716 032737 040000 001246 CL3TST: BIT      #BIT14,$DEV
7659 005724 001063          BNE      TUTST      ;:DROP TEST?
7660          ;:BR IF YES
7661 005726 013700 014636          MOV      CTPARM,RO  ;:GET THE CLUSTER PARAMETERS
7662 005732 052700 060000          BIS      #CT3,RO    ;:SET CLUSTER TERMINAL 3 SELECT
7663 005736 032737 000004 001246 BIT      #BIT2,$DEV  ;:CHECK IF INTERNAL LOOPBACK
7664 005744 001002          BNE      1$        ;:IF INTERNAL GO LOAD PARAMETER REGISTER
7665 005746 042700 000002          GIC      #MAINT,RO  ;:EXTERNAL CLEAR INTERNAL LOOPBACK BIT
7666 005752 010037 177420          1$: MOV      RO,PARAM  ;:SETUP CLUSTER TERMINAL 3
7667
7668 005756 004537 012642          JSR      R5,SETVEC  ;:SETUP INTERR VECTORS
7669 005762 000320          TK3VEC
7670 005764 014774          TK3SRV
7671 005766 014756          TP3SRV
7672
7673 005770 005037 015066          CLR      T3CHR      ;:CHAR TO BE XMITTED
7674 005774 005037 015072          CLR      T3CNT      ;:CLEAR COUNT OF CHARACTER PASSES
7675 006000 013737 176522 015070 MOV      TK3B,T3HLD  ;:READ CHAR (CLEAR STALE DATA)
7676 006006 052737 000100 176520 BIS      #IE,TK3S    ;:SET CLUSTER TERM #3 LOOSE!
7677 006014 052737 000100 176524 BIS      #IE,TP3S
7678
7679 006022 004537 013156          JSR      R5,TIMER   ;:SEE IF T3CNT GOES TO 1
7680 006026 015072          T3CNT
7681 006030 000001          1
7682 006032 104021          ERROR+21         ;:CLUSTER TERM 3 NOT RESPONDING
7683 006034 000417          BR      TUTST      ;:CHECK FOR TU58
7684 006036 052737 000002 001266 BIS      #BIT1,DEVACT ;:SET CLUSTER TERMINAL 3 ACTIVE FLAG
7685
7686 006044 032777 010000 173066 BIT      #BIT12,@SWR ;:SKIP TYPEOUT IF NOT SET
7687 006052 001410          BEQ      TUTST      ;:CHECK FOR TU58
7688 006054 104407          SAVEVT          ;:ALLOW 3 TYPEOUTS ON SAME LINE
7689 006056 104401 016465          TYPE      ,CLTERM
7690 006062 104401 016514          TYPE      ,T3
7691 006066 104401 017045          TYPE      ,RUN
7692 006072 104410          RESTVT          ;:RESTORE VT100 DISPLAY
7693
7694
7695      ;:*****
7696      ;:*          START TU58 TESTING
7697      ;:*****
7698
7699 006074 005737 012620          TUTST: TST      TUPRES ;:CHECK TO SEE IF TU58 PRESENT
7700 006100 001002          BNE      TUINIT    ;:IF PRESENT CHECK TO SEE IF ACTIVE
7701 006102 000137 011230          JMP      EOP        ;:TU58 NOT PRESENT END OF TEST
7702
7703 006106 005037 007032          TUINIT: CLR      UNITNM ;:SET TU58 UNIT NUMBER TO 0
7704 006112 032737 000400 001246 BIT      #BIT8,$DEV  ;:CHECK IF UNIT 0 SELECTED
7705 006120 001410          BEQ      2$        ;:IF SELECTED START TESTING
7706 006122 032737 001000 001246 BIT      #BIT9,$DEV  ;:CHECK IF UNIT 1 SELECTED
7707 006130 001402          BEQ      1$        ;:IF SELECTED BUMP UNIT NUMBER AND START
7708 006132 000137 011230          JMP      EOP        ;:NEITHER UNIT SELECTED ABORT TEST
7709 006136 005237 007032          1$: INC      UNITNM ;:SET UNIT TO UNIT 1

```

7710	006142	005037	177170		2\$:	CLR	TURCSR	;CLEAR RECEIVER TU58 CSR REGISTER
7711	006146	005037	177174			CLR	TUXCSR	;CLEAR TRANSMIT TU58 CSR REGISTER
7712	006152	005737	177172			TST	TURXDB	;CLEAR RECEIVER FLAG IF SET
7713	006156	012737	030210	007054		MOV	#TURDBF,TURPNT	;SETUP BUFFER ADDRESS FOR STORING CHAR
7714	006164	013737	007054	007056		MOV	TURPNT,TUBUFP	;SETUP BUFFER ADDRESS FOR GETTING CHAR
7715	006172	005037	007064			CLR	TU58ER	;CLEAR INTERRUPT SERVICE ERROR WORD
7716	006176	012737	000004	007030		MOV	#TINIT,OPCODE	;SETUP OPCODE FOR ERROR PRINTOUT
7717	006204	004537	012642			JSR	R5,SETVEC	;GO SETUP TU58 VECTORS
7718	006210	000260				TURVEC		;TU58 RECEIVER VECTOR ADDRESS
7719	006212	015600				TURCVR		;TU58 RECEIVER INT SERVICE ROUTINE
7720	006214	015444				TUXMIT		;TU58 TRANSMIT INT SERVICE ROUTINE
7721	006216	012737	177777	007102		MOV	#-1,BRKFLG	;SET BREAK FLAG
7722	006224	004537	007532			JSR	R5,SNPKT	;GO INIT THE DEVICE AND SET RECEIVER INT ENA
7723	006230	007152				.WORD	TUIMSG	
7724	006232	012700	000002			MOV	#2,RO	;SETUP TO RECEIVE 1 UNKNOWN CHAR FROM TU58
7725	006236	004537	007622		3\$:	JSR	R5,GETCHR	;GO GET A CHARACTER FROM TU58
7726	006242	000415				BR	4\$;DEVICE ERROR
7727	006244	122737	000020	007104		CMPB	#TCONT,TUCHR	;CHECK IF CONTINUE
7728	006252	001414				BEQ	5\$;IF CONTINUE THEN DEVICE READY
7729	006254	005300				DEC	RO	;DECREMENT UNKNOWN CHAR COUNTER
7730	006256	001367				BNE	3\$;GO GET NEXT CHAR - SHOULD BE CONT
7731	006260	052737	001000	007064		BIS	#BIT9,TU58ER	;SET BIT INDICATING TU58 INIT PROBLEM
7732	006266	005037	177170			CLR	TURCSR	;CLEAR INTERRUPT ENABLES
7733	006272	005037	177174			CLR	TUXCSR	
7734	006276	104036			4\$:	ERROR+36		;DEVICE ERROR
7735	006300	000137	011230			JMP	EOP	;EXIT TU58 TEST CAN NOT INIT DEVICE
7736	006304	042737	000100	177170	5\$:	BIC	#IE,TURCSR	;CLEAR RECEIVER INT ENABLE
7737	006312	004737	026336			JSR	PC,\$RAND	;GO GENERATE RANDOM TAPE RANGE
7738	006316	005037	007044			CLR	HILOWF	;SET TAPE RANGE TO LOWER HALF
7739	006322	105737	026436			TSTB	\$LONUM	;IF BIT 7=0 THEN RANGE IS LOW
7740	006326	100003				BPL	6\$;GO START TESTING ON LOW RANGE
7741	006330	052737	000100	007044		BIS	#BIT6,HILOWF	;SET HIGH RANGE BIT
7742	006336	032777	010000	172574	6\$:	BIT	#BIT12,@SWR	;CHECK IF PERFORMAMNCE REPORT ENABLED
7743	006344	001406				BEQ	TSTT58	;IF NOT SKIP PERFORMANCE REPORT
7744	006346	104407				SAVEVT		;SAVE THE CURSOR POSITION
7745	006350	104401	016452			TYPE	,TU58M	;TYPE TU58 RUNNING
7746	006354	104401	01'045			TYPE	,RUN	
7747	006360	104410				RESTVT		;RESTORE THE CURSOR POSITION
7748	006362	012737	000012	007046	TSTT58:	MOV	#10,.,BLKCNT	;SETUP TO DO 10 BLOCKS PER DRIVE
7749	006370	042737	000100	177170	TSTTU:	BIC	#IE,TURCSR	;CLEAR RECEIVER INTERRUPT ENABLE
7750	006376	042737	000100	177174		BIC	#IE,TUXCSR	;CLEAR TRANSMIT INTERRUPT ENABLE
7751	006404	005737	177172			TST	TURXDB	;CLEAR RECEIVE FLAG IF IT WAS SET
7752	006410	012737	030210	007054		MOV	#TURDBF,TURPNT	;SETUP BUFFER ADDRESS FOR STORING CHAR
7753	006416	013737	007054	007056		MOV	TURPNT,TUBUFP	;SETUP BUFFER ADDRESS FOR GETTING CHAR
7754	006424	052737	000100	177170		BIS	#IE,TURCSR	;SET RECEIVER INTERRUPT ENABLE
7755	006432	005037	007064			CLR	TU58ER	;CLEAR TU58 INT SRV ERROR FLAG
7756	006436	005037	007070			CLR	CONTFI	;CLEAR CONTINUE FLAG COUNTER
7757	006442	005037	007072			CLR	DATAFL	;CLEAR DATA PACKET COUNTER
7758	006446	005037	007062			CLR	PKTERR	;CLEAR DATA/END PACKET ERROR FLAG
7759	006452	004537	007156			JSR	R5,BLDPKT	;GO BUILD WRITE COMMAND PACKET
7760	006456	000003				.WORD	TWRITE	
7761	006460	012700	007136			MOV	#WRTTBL,RO	;SETUP POINTER TO WRITE PACKET ADDRESSES
7762	006464	012037	006476		1\$:	MOV	(RO)+,2\$;GET ADDRESS OF PACKET AND SAVE
7763	006470	001411				BEQ	3\$;IF ALL DONE - WAIT FOR END PACKET
7764	006472	004537	007532			JSR	R5,SNPKT	;GO SEND THE COMMAND OR DATA PACKET
7765	006476	007026			2\$:	.WORD	FLAG	;THIS ADDRESS WILL CHANGE

7766	006500	011037	006512		MOV	(R0),22\$;GET ADDRESS OF NEXT PACKET TO BUILD	
7767	006504	001403			BEQ	3\$;IF TERMINATOR WAIT FOR END PACKET	
7768	006506	004537	007426		JSR	R5,WTDPAK	;GO BUILD WRITE DATA PACKET	
7769	006512	027170		22\$:	.WORD	WTPK1	;THIS ADDRESS WILL CHANGE	
7770								
7771	006514	004537	007742	3\$:	JSR	R5,GETPKT	;GO WAIT AND SERVICE PACKET	
7772	006520	000403			BR	4\$;DEVICE ERROR	
7773	006522	000760			BR	1\$;CONT - SEND NEXT PACKET IF ANY	
7774	006524	000404			BR	5\$;END PACKET ERROR DETECTED	
7775	006526	000414			BR	7\$;END PACKET NO ERRORS DETECTED	
7776	006530	104036		4\$:	ERROR+36		;DEVICE ERROR	
7777	006532	000137	011230		JMP	EOP	;EXIT TO END OF PASS IF CONTINUED	
7778								
7779	006536	122737	000001	007074	5\$:	CMPB	#1,SUCCESS	;CHECK IF SUCCESS WITH RETRIES
7780	006544	001402			BEQ	6\$;IF YES THEN PRINT ERROR AND CONTINUE	
7781	006546	104037			ERROR+37		;REPORT FATAL ERROR FOR THIS DRIVE	
7782	006550	000442			BR	13\$;GO TO NEXT DRIVE IF AVAILABLE	
7783	006552	004537	011142	6\$:	JSR	R5,SOFTER	;GO CHECK IF SOFT ERROR TO BE REPORTED	
7784	006556	104041			ERROR+41		;REPORT SOFT ERROR DURING WRITE	
7785	006560	005037	007064	7\$:	CLR	TU58ER	;CLEAR ERROR FLAG FOR INT SRV	
7786	006564	005037	007062		CLR	PKTERR	;CLEAR PACKET ERROR FLAG	
7787	006570	004537	007156		JSR	R5,BLDPKT	;GO BUILD READ PACKET	
7788	006574	000002			.WORD	TREAD		
7789	006576	004537	007532		JSR	R5,SNDPKT	;GO SEND THE PACKET	
7790	006602	007026			.WORD	FLAG	;ADDRESS OF COMMAND PACKET	
7791	006604	004537	007742	8\$:	JSR	R5,GETPKT	;GO WAIT AND SERVICE PACKET	
7792	006610	000403			BR	9\$;DEVICE ERROR	
7793	006612	000774			BR	8\$;DATA PACKET - GET NEXT PACKET	
7794	006614	000404			BR	10\$;END PACKET ERROR DETECTED	
7795	006616	000414			BR	12\$;END PACKET NO ERRORS DETECTED	
7796	006620	104036		9\$:	ERROR+36		;DEVICE ERROR DURING READ	
7797	006622	000137	011230		JMP	EOP	;ABORT TU58 TESTING ON DEVICE FAILURE	
7798	006626	122737	000001	007074	10\$:	CMPB	#1,SUCCESS	;CHECK IF SUCCESS WITH RETRIES
7799	006634	001402			BEQ	11\$;IF YES UPDATE SOFT ERROR AND CONT	
7800	006636	104040			ERROR+40		;REPORT FATAL ERROR FOR THIS DEVICE	
7801	006640	000406			BR	13\$;GO CHECK FOR NEXT DRIVE	
7802	006642	004537	011142	11\$:	JSR	R5,SOFTER	;GO CHECK IF SOFT ERROR TO BE REPORTED	
7803	006646	104042			ERROR+42		;REPORT SOFT ERROR AND CONT IF ALLOWED	
7804	006650	005337	007046	12\$:	DEC	BLKCNT	;DECREMENT BLOCK COUNTER	
7805	006654	001245			BNE	TSTTU	;IF NOT 0 DO NEXT BLOCK	
7806								
7807	006656	005737	007032	13\$:	TST	UNITNM	;CHECK IF UNIT 1	
7808	006662	001035			BNE	15\$;IF UNIT 1 THEN EXIT	
7809	006664	032777	010000	172246	BIT	#BIT12,@SWR	;CHECK IF PERFORMANCE REPORT	
7810	006672	001410			BEQ	14\$;IF NOT THEN SKIP TYPE OUT	
7811	006674	104407			SAVEVT		;SAVE THE CURSOR POSITION	
7812	006676	104401	016457		TYPE	,TU58MS	;TYPE TU58 TAPE UNIT 0 DONE	
7813	006702	104401	016773		TYPE	,DRVO		
7814	006706	104401	017074		TYPE	,DUN		
7815	006712	104410			RESTVT		;RESTORE THE CURSOR POSITION	
7816	006714	032737	001000	001246	14\$:	BIT	#BIT9,\$DEVN	;CHECK IF UNIT 1 SELECTED
7817	006722	001031			BNE	16\$;IF NOT THEN EXIT THE TEST	
7818	006724	005237	007032		INC	UNITNM	;UPDATE THE UNIT NUMBER TO 1	
7819	006730	032737	000040	001246	BIT	#BIT5,\$DEVN	;CHECK IF SEQUENTIAL BLOCKS	
7820	006736	001611			BEQ	TSTT58	;IF 0 THEN RANDOM BLOCKS SELECTED	
7821	006740	162737	000012	007040	SUB	#10.,BLKNUM	;BACK UP BLOCK NUMBER FOR TAPE UNIT 1	

```

7822 006746 042737 177000 007040      BIC      #177000,BLKNUM ;MASK TO BLOCK NUMBER RANGES
7823 006754 000602                    BR       TST158      ;GO DO TEST ON DRIVE 1
7824 006756 032777 010000 172154 15$: BIT      #BIT12,@SWR ;CHECK IF PERFORMANCE REPORT
7825 006764 001410                    BEQ     16$          ;IF NOT THEN EXIT THE TEST
7826 006766 104407                    SAVEVT ;SAVE THE CURSOR POSITION
7827 006770 104401 016457              TYPE    ,TU58MS     ;TYPE TU58 TAPE UNIT 1 DONE
7828 006774 104401 017007              TYPE    ,DRV1
7829 007000 104401 017074              TYPE    ,DUN
7830 007004 104410                    RESTVT ;RESTORE THE CURSOR POSITION
7831 007006 042737 000100 177170 16$: BIC      #1E,TURCSR ;CLEAR RECEIVER INT ENABLE BIT
7832 007014 042737 000100 177174     BIC      #1E,TUXCSR ;CLEAR TRANSMIT INT ENABLE BIT
7833 007022 000137 011230              JMP     EOP         ;GO TO EOP ROUTINE
7834
7835 ;COMMAND PACKET TO TU58
7836 ;
7837 007026 000000      FLAG: .WORD 0 ;HAS PACKET BYTE COUNT AND COMMAND FLAG
7838 007030 000000      OPCODE: .WORD 0 ;HAS OPCODE AND MODIFIER
7839 007032 000000      UNITNM: .WORD 0 ;HAS UNIT NUMBER
7840 007034 000000      SEQNUM: .WORD 0 ;N/A FOR TU58
7841 007036 000000      BYTCNT: .WORD 0 ;BLOCK BYTE COUNT
7842 007040 000000      BLKNUM: .WORD 0 ;HAS BLOCK NUMBER
7843 007042 000000      CKSUM: .WORD 0 ;COMMAND PACKET CHECKSUM WORD
7844
7845 007044 000000      HILOWF: .WORD 0 ;TAPE RANGE INDICATOR
7846 007046 000000      BLKCNT: .WORD 0 ;NUMBER OF TOTAL BLOCKS LEFT TO EXER
7847
7848 007050 000000      TUXPNT: .WORD 0 ;POINTER TO WRITE PACKET DATA
7849 007052 000000      TUXCNT: .WORD 0 ;NUMBER OF BYTES TO BE TRANSMITTED
7850
7851 007054 000000      TURPNT: .WORD 0 ;POINTER TO RECV BUFFER FOR STORING BYTE
7852 007056 000000      TUBUFP: .WORD 0 ;POINTER TO RECV BUFFER FOR GETTING CHAR
7853
7854 007060 000000      PKTPNT: .WORD 0 ;POINTER TO END PACKET IN RECV BUFFER
7855 ;*
7856 ;*****
7857 ;*
7858 007062 000000      PKTERR: .WORD 0 ;END PACKET ERROR WORD BIT DEFINITIONS
7859 ;
7860 ;BIT15=1 READ PACKET 4 FAILED
7861 ;
7862 ;BIT14=1 READ PACKET 3 FAILED
7863 ;BIT13=1 READ PACKET 2 FAILED
7864 ;BIT12=1 READ PACKET 1 FAILED
7865 ;
7866 ;BIT11=1
7867 ;BIT10=1
7868 ;BIT09=1 READ - < 4 DATA PACKETS RECV'D
7869 ;
7870 ;BIT08=1 WRITE - < 4 CONT RECV'D
7871 ;
7872 ; END PACKET ERROR BITS
7873 ;
7874 ;BIT07=1 CHECKSUM ERROR
7875 ;BIT06=1 PACKET BYTE COUNT ERROR
7876 ;
7877 ;BIT05=1 OPCODE ERROR
    
```

7878 :BIT04=1 SUCCESS CODE FAILURE
7879 :BIT03=1 UNIT NUMBER READ INCORRECT
7880
7881 :BIT02=1 SEQUENCE NUMBER NOT 0
7882 :BIT01=1 DATA BYTE COUNT INCORRECT
7883 :BIT00=1 SUMMARY WORD FAILURE
7884
7885
7886
7887

007064 000000

TU58ER: .WORD 0

: DEVICE ERROR WORD BIT DEFINITIONS
:BIT15=1 UNRECOGNIZABLE CHAR RECV'D
:BIT14=1 > 4 DATA PACKETS RECV'D
:BIT13=1 > 4 CONT RECV'D DURING WRITE
:BIT12=1 CONT RECV'D DURING READ
:BIT11=1 DATA RECV'D DURING WRITE
:BIT10=1 INIT RECV'D
:BIT09=1 ERROR INITTING THE TU58
:BIT08=1
:BIT07=1 DEVICE TIMEOUT ERROR
:BIT06=1 RECVR DONE NOT SET ON INTERRUPT
:BIT05=1 RECVR BUFFER OVERFLOWED
:BIT04=1 XMIT RDY NOT SET ON INTERRUPT
:BIT03=1 XMIT INTERRUPT WITH IE CLEARED
:BIT02=1
:BIT01=1
:BIT00=1

7911 :*
7912 :*****
7913 :*
7914 :*

007066 000000

DPKTER: .WORD 0

: ACCUMULATED DATA PAKET ERROR WORD
: DATA PACKET 4 ERROR BITS
:BIT14=1 PACKET CHECKSUM FAILURE
:BIT13=1 DATA CHECKSUM FAILURE
:BIT12=1 PACKET BYTE COUNT INCORRECT
: DATA PACKET 3 ERROR BITS
:BIT11=1 PACKET CHECKSUM FAILURE
:BIT10=1 DATA CHECKSUM FAILURE
:BIT09=1 PACKET BYTE COUNT INCORRECT
: DATA PACKET 2 ERROR BITS
:BIT08=1 PACKET CHECKSUM FAILURE
:BIT07=1 DATA CHECKSUM FAILURE
:BIT06=1 PACKET BYTE COUNT INCORRECT
: DATA PACKET 1 ERROR BITS
:BIT05=1 PACKET CHECKSUM FAILURE
:BIT04=1 DATA CHECKSUM FAILURE

7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933

```
7934                                     ;BIT03=1      PACKET BYTE COUNT INCORRECT
7935                                     :*
7936                                     :*****
7937                                     :*
7938
7939 007070 000000      CONTFL: .WORD 0      ;NUMBER OF CONT FLAGS RECEIVED FROM TUS8
7940 007072 000000      DATAFL: .WORD 0      ;NUMBER OF DATA FLAGS RECEIVED FROM TUS8
7941
7942 007074 000000      SUCCES: .WORD 0      ;DATA PACKET SUCCESS CODE
7943 007076 000000      BYTECT: .WORD 0      ;DATA PACKET DATA WORD COUNT
7944 007100 000000      SUMMARY: .WORD 0      ;DATA PACKET SUMMARY WORD
7945
7946 007102 000000      BRKFLG: .WORD 0      ;SET WHEN INITTING DEVICE
7947 007104 000000      TUCHR: .WORD 0      ;BYTE READ FROM RECEIVE BUFFER
7948
7949 007106 000000      WRKSUM: .WORD 0      ;POINTER TO WRITE CHECKSUM TABLE
7950 007110 000000      WRKERR: .WORD 0      ;POINTER TO DATA PACKET ERROR FLAGS
7951 007112 000000      WRKPNT: .WORD 0      ;POINTER TO DATA PACKET ADDRESS TABLE
7952 007114 000000      DERFLG: .WORD 0      ;USED TO SET PKTERR WHEN DATA PACKET ERR
7953
7954
7955 007116 000000      DATPNT: .WORD 0      ;POINTER TO DATA PACKET 1 STARTING ADD
7956 007120 000000      .WORD 0      ;POINTER TO DATA PACKET 2 STARTING ADD
7957 007122 000000      .WORD 0      ;POINTER TO DATA PACKET 3 STARTING ADD
7958 007124 000000      .WORD 0      ;POINTER TO DATA PACKET 4 STARTING ADD
7959
7960 007126 000000      DERTBL: .WORD 0      ;ERROR WORD FOR DATA PACKET 1
7961 007130 000000      .WORD 0      ;ERROR WORD FOR DATA PACKET 2
7962 007132 000000      .WORD 0      ;ERROR WORD FOR DATA PACKET 3
7963 007134 000000      .WORD 0      ;ERROR WORD FOR DATA PACKET 4
7964
7965
7966                                     ;THE FOLLOWING TABLE CONTAINS POINTERS FOR WRITE PACKETS
7967
7968 007136 007026      WRITBL: .WORD FLAG      ;ADDRESS OF COMMAND PACKET
7969 007140 027170      WRTPNT: .WORD WTPK1      ;ADDRESS OF WRITE PACKET 1
7970 007142 027374      .WORD WTPK2      ;ADDRESS OF WRITE PACKET 2
7971 007144 027600      .WORD WTPK3      ;ADDRESS OF WRITE PACKET 3
7972 007146 030004      .WORD WTPK4      ;ADDRESS OF WRITE PACKET 4
7973 007150 000000      .WORD 0      ;TABLE TERMINATOR
7974
7975 007152 000 000 004  TUIMSG: .BYTE 0,0,TINIT,TINIT
7976 007155 004
7977
7978                                     :*****
7979                                     ;*THE FOLLOWING ROUTINE WILL BUILD THE PACKETS SPECIFIED BY THE
7980                                     ;*FUNCTION SELECTED.
7981                                     :*****
7982 007156 010046      BLDPKT: MOV RO,-(SP)      ;SAVE ROUTINES WORKING REGISTER
7983 007160 022715 000003      CMP #TWRITE,(R5)      ;CHECK IF FUNCTION = WRITE
7984 007164 001024      BNE 1$      ;IF NOT WRITE - BRANCH
7985 007166 032737 000040 001246      BIT #BIT5,$DEV      ;CHECK IF SEQUENTIAL BLOCKS SELECTED
7986 007174 001406      BEQ 5$      ;IF NOT SELECTED GENERATE RANDOM NUMBER
7987 007176 005237 007040      INC BLKNUM      ;INCREMENT BLOCK NUMBER
7988 007202 042737 177000 007040      BIC #177000,BLKNUM      ;MASK TO BLOCK NUMBER RANGE
```

```
7989 007210 000412          BR      1$          ;GO BUILD REST OF PACKET
7990 007212 004737 026336 5$: JSR      PC,$RAND ;GENERATE RANDOM BLOCK NUMBER
7991 007216 013700 026436      MOV     $LONUM,R0 ;GET A RANDOM NUMBER
7992 007222 042700 177100      BIC     #177100,R0 ;MASK TO LOW RANGE BLOCK NUMBER
7993 007226 053700 007044      BIS     HILOWF,R0 ;SET BLOCK TO HIGH OR LOW RANGE
7994 007232 010037 007040      MOV     R0,BLKNUM ;SAVE THE BLOCK NUMBER IN PACKET
7995 007236 112737 000012 007027 1$: MOVB    #TMSG,FLAG+1 ;SETUP PACKET BYTE COUNT
7996 007244 112737 000002 007026      MOVB    #TCMDPK,FLAG ;SETUP COMMAND PACKET INDICATOR
7997 007252 011537 007030      MOV     (R5),OPCODE ;GET THE FUNCTION AND SAVE
7998 007256 032737 000040 001246      BIT     #BITS,$DEVM ;CHECK IF SEQUENTIAL BLOCK TESTING
7999 007264 001010          BNE     2$          ;IF SEQUENTIAL DO NOT GENERATE RANDOM MODIFIER
8000 007266 004737 026336      JSR     PC,$RAND ;GENERATE RANDOM MODIFIER
8001 007272 105737 026436      TSTB   $LONUM ;IF BIT 7=0 THEN NORMAL READ/WRITE
8002 007276 100003          BPL     2$          ;BRANCH IF NORMAL FUNCTION
8003 007300 052737 000400 007030      BIS     #BIT8,OPCODE ;SET MODIFIER
8004 007306 005037 007034 2$: CLR     SEQNUM ;CLEAR THE SEQUENCE NUMBER
8005 007312 012737 001000 007036      MOV     #TWC,BYTCNT ;SET BYTE COUNT TO 512 BYTES (1 BLOCK)
8006 007320 022725 000007      CMP     #TDIAGN,(R5)+ ;CHECK IF FUNCTION WAS DIAGNOSE
8007 007324 001007          BNE     3$          ;IF NOT GO BUILD PACKET CHECKSUM
8008 007326 042737 000400 007030      BIC     #BIT8,OPCODE ;CLEAR MODIFIER FOR FUNCTION
8009 007334 005037 007036      CLR     BYTCNT ;CLEAR DATA BYTE COUNT
8010 007340 005037 007040      CLR     BLKNUM ;CLEAR BLOCK NUMBER NO TAPE MOTION
8011 007344 004537 007356 3$: JSR     R5,BLDCHK ;GO BUILD CHECKSUM FOR COMMAND PASKET
8012 007350 007026          .WORD  FLAG ;ADDRESS OF PACKET TO BE CHECKSUMMED
8013 007352 012600 4$: MOV     (SP)+,R0 ;RESTORE WORKING REGISTER
8014 007354 000205          RTS     R5 ;RETURN BACK TO THE TEST
```

```
8015
8016 ;:*****
8017 ;ROUTINE TO BUILD COMMAND AND DATA PACKET CHECKSUMS
8018 ;:
8019 ;:*****
```

```
8020
8021 007356 010046          BLDCHK: MOV     R0,-(SP) ;SAVE ROUTINE WORKING REGISTERS
8022 007360 010146          MOV     R1,-(SP) ;
8023 007362 010246          MOV     R2,-(SP) ;
8024 007364 012500          MOV     (R5)+,R0 ;GET ADDRESS OF PACKET TO CHECKSUM
8025 007366 011007          MOV     (R0),R2 ;GET FLAG AND PACKET BYTE COUNT
8026 007370 000302          SWAB   R2 ;SWAP FLAG WITH PACKET BYTE COUNT
8027 007372 042702 177400      BIC     #177400,R2 ;DROP FLAG BYTE
8028 007376 006202          ASR    R2 ;DIVIDE BY 2 FOR WORD COUNT
8029 007400 005202          INC    R2 ;ADD IN FLAG AND PACKET BYTE COUNT
8030 007402 005001          CLR    R1 ;USE R1 TO CALCULATE CHECKSUM
8031 007404 062001 1$: ADD     (R0)+,R1 ;BUILD CHECKSUM VALUE
8032 007406 005501          ADC    R1 ;DO END AROUND CARRY
8033 007410 005302          DEC    R2 ;CHECKSUM FINISHED?
8034 007412 001374          BNE    1$ ;IF NOT DO NEXT BYTE
8035 007414 010110          MOV    R1,(R0) ;SAVE CHECKSUM IN PACKET
8036 007416 012602          MOV    (SP)+,R2 ;RESTORE WORKING REGISTERS
8037 007420 012601          MOV    (SP)+,R1 ;
8038 007422 012600          MOV    (SP)+,R0 ;
8039 007424 000205          RTS    R5 ;RETURN BACK TO CALLING ROUTINE
```

```
8040
8041 ;:*****
8042 ;THIS ROUTINE GENERATES RANDOM DATA AND BUILDS WRITE DATA PACKETS
8043 ;:
8044 ;:*****
```

```
8045
8046 007426 010046          WTDPAK: MOV    R0,-(SP)      ;SAVE ROUTINES WORKING REGISTERS
8047 007430 010146          MOV    R1,-(SP)      ;
8048 007432 010246          MOV    R2,-(SP)      ;
8049 007434 010346          MOV    R3,-(SP)      ;
8050 007436 010446          MOV    R4,-(SP)      ;
8051 007440 012500          MOV    (R5)+,R0      ;GET ADDRESS OF PACKET TO BUILD
8052 007442 012702 000041  MOV    #TMSIZ/4+1,R2  ;SETUP PACKET WC/2 +1 FOR FLAG WORD
8053 007446 012710 100001  MOV    #100001,(R0)   ;SAVE DATA FLAG AND PACKET BYTE COUNT
8054 007452 005001          CLR    R1             ;CLEAR WORKING CHECKSUM
8055 007454 012704 000001  MOV    #1,R4          ;SETUP TO ADD FLAG AND BYTE COUNT TO CKSUM
8056 007460 000407          BR     3$             ;GO ADD FLAG AND BYTE COUNT TO CKSUM
8057 007462 012703 026440  1$:  MOV    #LONUM+2,R3  ;SETUP ADDRESS OF RANDOM NUMBER +2
8058 007466 012704 000002  MOV    #2,R4          ;SETUP WORKING COUNTER FOR 2 RANDOM #'S
8059 007472 004737 026336  JSR    PC,$RAND       ;GENERATE TWO RANDOM NUMBERS
8060 007476 014310          2$:  MOV    -(R3),(R0)    ;SAVE RANDOM NUMBER IN MEMORY
8061 007500 062001          3$:  ADD    (R0)+,R1      ;ADD NUMBER TO CHECKSUM
8062 007502 005501          ADC    R1             ;DO END AROUND CARRY
8063 007504 005304          DEC    R4             ;DECREMENT RANDOM WORKING COUNTER
8064 007506 001373          BNE    2$             ;IF NOT 0 GET NEXT RANDOM NUMBER
8065 007510 005302          DEC    R2             ;DECREMENT PACKET WC/2 +1
8066 007512 001363          BNE    1$             ;GO GET NEXT TWO WORDS
8067 007514 010120          MOV    R1,(R0)+      ;SAVE THE CHECKSUM IN MEMORY
8068 007516 012604          MOV    (SP)+,R4      ;RESTORE THE WORKING REGISTERS
8069 007520 012603          MOV    (SP)+,R3      ;
8070 007522 012602          MOV    (SP)+,R2      ;
8071 007524 012601          MOV    (SP)+,R1      ;
8072 007526 012600          MOV    (SP)+,R0      ;
8073 007530 000205          RTS    R5             ;RETURN BACK AND WAIT FOR FLAG BYTE
```

```
8074
8075
8076
8077
8078
8079
8080
8081
8082
8083
8084
8085
8086
8087
8088
8089
8090
8091
8092
8093
8094
8095
8096
8097
8098
8099
8100
```

*THE FOLLOWING ROUTINE WILL SETUP POINTERS TO THE COMMAND OR DATA
*PACKET TO BE SENT. CALL PLUS 2 CONTAINS THE POINTER TO THE PACKET TO
*BE SENT. THE TUS8 XMIT INTERRUPT ENABLE BIT WILL BE SET ON EXIT.

```
SNDPKT: MOV    R0,-(SP)      ;SAVE WORKING REGISTER
MOV    @R5,R0          ;GET FLAG AND BYTE COUNT WORD
SWAB   R0              ;EXCHANGE BYTES
BIC    #177400,R0     ;CLEAR THE FLAG BYTE
ADD    #4,R0           ;ADD FLAG AND CHECKSUM WORDS TO BYTE COUNT
MOV    R0,TUXCNT      ;SAVE THE TOTAL PACKET BYTE COUNT
MOV    (R5)+,TUXPNT   ;SETUP THE TRANSMIT POINTER TO PACKET
MOV    (SP)+,R0       ;RESTORE THE WORKING REGISTER
TST    BRKFLG         ;CHECK IF BREAK BEING PERFORMED
BEQ    1$             ;IF NOT THEN SET IF BIT
BIS    #BREAK,TUXCSR  ;SET THE BREAK BIT IN XMIT CSR REGISTER
TST    TURXDB         ;CLEAR RECEIVE FLAG IF ANY SET
BIS    #IE,TURCSR     ;SET RECEIVER INT ENABLE BIT
1$:  BIS    #IE,TUXCSR  ;SET TRANSMIT INTERRUPT ENABLE
2$:  RTS    R5         ;RETURN BACK TO THE TEST
```

*THE FOLLOWING ROUTINE WILL GET A CHARACTER FROM THE TUS8 READ MEMORY
*BUFFER AND SAVE IT IN LOCATION TUCHR. IF THE ROUTINE DETECTS A FAILURE

```

8101      ;*IN THE RECEIVE/TRANSMIT INTERRUPT SERVICE ROUTINES, THE ROUTINE WILL
8102      ;*RETURN TO CALL+2, OTHERWISE, THE RETURN IS TO CALL+4.  IF A TIMEOUT
8103      ;*ERROR, RETURN IS ALSO TO CALL+2.
8104      ;:*****
8105
8106 007622 010046      GETCHR: MOV     R0,-(SP)      ;SAVE THE WORKING REGISTER
8107 007624 010146      MOV     R1,-(SP)
8108 007626 010246      MOV     R2,-(SP)
8109 007630 013700 007056  MOV     TUBUFP,R0      ;GET ADDRESS OF READ BUFFER POINTER
8110 007634 005001      CLR     R1              ;SETUP TIMEOUT REGISTERS
8111 007636 012702 000021  MOV     #17.,R2
8112 007642 005737 007064  1$:   TST     TU5BER      ;CHECK IF ANY ERRORS DETECTED BY INT SRV
8113 007646 001021      BNE
8114 007650 023700 007054  CMP     TURPNT,R0      ;CHECK IF CHARACTER RECEIVED
8115 007654 101405      BLOS   2$              ;IF NOT GO CHECK FOR DEVICE TIMEOUT
8116 007656 112037 007104  MOVB   (R0)+,TUCHR     ;GET THE CHARACTER FROM MEMORY
8117 007662 010037 007056  MOV     R0,TUBUFP     ;UPDATE MEMORY BUFFER POINTER
8118 007666 000416      BR     4$              ;RETURN CALL+4 - CHAR RECEIVED
8119 007670 005301      2$:   DEC     R1              ;DECREMENT 1ST TIMEOUT COUNTER
8120 007672 001363      BNE    1$              ;WAIT AGAIN IF NOT 0
8121 007674 005237 001200  INC     $DEVCT         ;INDICATE TO APT PROGRAM STILL RUNNING
8122 007700 005302      DEC     R2              ;CHECK IF 2ND COUNTER OVERFLOWED
8123 007702 001357      BNE    1$              ;IF NOT GO WAIT AGAIN
8124 007704 052737 000200 007064  BIS    #BIT7,TU5BER    ;SET TIMEOUT ERROR BIT
8125 007712 005037 177170  3$:   CLR     TURCSR      ;CLEAR BOTH RECV AND XMIT CSR REGISTERS
8126 007716 005037 177174  CLR     TUXCSR
8127 007722 000403      BR     5$              ;RESTORE WORKING REGISTERS RETURN CALL+2
8128 007724 005725      4$:   TST     (R5)+      ;BUMP POINTER PAST DEVICE FAILURE ADDRESS
8129 007726 005237 001200  INC     $DEVCT         ;INDICATE TO APT PROGRAM STILL RUNNING
8130 007732 012602      5$:   MOV     (SP)+,R2     ;RESTORE THE WORKING REGISTERS
8131 007734 012601      MOV     (SP)+,R1
8132 007736 012600      MOV     (SP)+,R0
8133 007740 000205      RTS     R5              ;RETURN BACK TO PROGRAM
    
```

```

8134
8135      ;:*****
8136      ;*THE FOLLOWING ROUTINE WILL READ A CHARACTER AND DETERMINE WHAT
8137      ;*FUNCTION IS TO BE PERFORMED. THE PROGRAM WILL THEN GO TO THE FUNCTION
8138      ;*ROUTINE AND SERVICE IT. THE RETURNS ARE LISTED BELOW FOR THE FUNCTION.
8139      ;*
8140      ;*CALL+2      DEVICE ERROR
8141      ;*CALL+4      (READ)DATA PACKET/(WRITE)CONT
8142      ;*CALL+6      END PACKET ERROR DETECTED
8143      ;*CALL+10     END PACKET NO ERRORS DETECTED
8144      ;:*****
    
```

```

8146 007742 010046      GETPKT: MOV     R0,-(SP)      ;SAVE THE WORKING REGISTER
8147 007744 004537 007622  JSR     R5,GETCHR     ;GO GET A CHARACTER FROM BUFFER
8148 007750 000461      BR     6$              ;DEVICE ERROR RETURN CALL+2
8149 007752 113700 007104  MOVB   TUCHR,R0      ;GET THE CHARACTER READ
8150 007756 122700 000020  CMPB   #TCONT,R0     ;CHECK IF IT WAS A CONTINUE
8151 007762 001022      BNE    2$              ;IF NOT CHECK IF IT WAS AN INIT
8152 007764 122737 000003 007030  CMPB   #TWRITE,OPCODE ;CHECK IF FUNCTION WAS WRITE
8153 007772 001404      BEQ    1$              ;IF WRITE - COUNT THE NUMBER OF CONT'S
8154 007774 052737 010000 007064  BIS    #BIT12,TU5BER  ;SET BIT INDICATING CONT RECV'D DURING READ
8155 010002 000444      BR     6$              ;DEVICE ERROR RETURN CALL+2
8156 010004 005237 007070  1$:   INC     CONTFL      ;INCREMENT CONTINUE FLAG COUNTER
    
```

```

8157 010010 122737 000004 007070      CMPB    #4,CONTFL      ;CHECK IF MORE THEN 4 CONT'S
8158 010016 103045                BHIS    9$              ;IF 4 OR LESS RETURN TO CALL+4
8159 010020 052737 020000 007064      BIS     #BIT13,TU58ER  ;SET BIT INDICATING MORE THEN 4 CONT'S
8160 010026 000432                BR      6$              ;DEVICE ERROR RETURN CALL+2
8161 010030 122700 000004      2$:    CMPB    #TINIT,RO  ;CHECK IF INIT CODE
8162 010034 001004                BNE     3$              ;IF NOT GO CHECK TO SEE IF DATA PACKET
8163 010036 052737 002000 007064      BIS     #BIT10,TU58ER  ;SET BIT INDICATING INIT RECEIVED
8164 010044 000423                BR      6$              ;DEVICE ERROR RETURN CALL+2
8165 010046 122700 000001      3$:    CMPB    #TDATA,RO  ;CHECK IF DATA PACKET
8166 010052 001004                BNE     4$              ;IF NOT GO CHECK FOR END PACKET
8167 010054 004537 010140      JSR     R5,DATPKT      ;GO READ IN THE PACKET
8168 010060 000415                BR      6$              ;DEVICE ERROR RETURN CALL+2
8169 010062 000423                BR      9$              ;NO DEVICE FAILURE RETURN CALL+4
8170 010064 122700 000002      4$:    CMPB    #TCMDPK,RO  ;CHECK IF END PACKET
8171 010070 001404                BEQ     5$              ;IF END PACKET GO READ IT
8172 010072 052737 100000 007064      BIS     #BIT15,TU58ER  ;SET BIT INDICATING UNRECOGNIZABLE CHAR
8173 010100 000405                BR      6$              ;DEVICE FAILURE RETURN CALL+2
8174 010102 004537 010512      5$:    JSR     R5,ENDPKT    ;GO READ IN THE END PACKET
8175 010106 000402                BR      6$              ;DEVICE FAILURE RETURN CALL+2
8176 010110 000407                BR      8$              ;PACKET ERROR RETRETURN CALL+6
8177 010112 000405                BR      7$              ;NO ERRORS RETURN CALL+10
8178 010114 005037 177170      6$:    CLR     TURCSR      ;CLEAR CSR REGISTERS
8179 010120 005037 177174      CLR     TUXCSR
8180 010124 000403                BR      10$             ;DEVICE ERROR RETURN CALL+2
8181 010126 005725                7$:    TST     (R5)+       ;END PACKET RETURN NO ERRORS CALL+10
8182 010130 005725                8$:    TST     (R5)+       ;END PACKET ERROR DETECTED RETURN CALL+6
8183 010132 005725                9$:    TST     (R5)+       ;DATA PACKET OR CONT RETURN CALL+4
8184 010134 012600                10$:   MOV     (SP)+,RO      ;RESTORE THE WORKING REGISTER
8185 010136 000205                RTS     R5              ;RETURN BACK TO PROGRAM
    
```

```

8186
8187
8188
8189
8190
8191
8192
8193
8194
8195
8196
8197
8198
8199
8200
8201
8202
8203
8204
8205
    ;*****
    ;*THE FOLLOWING ROUTINE WILL READ AND CHECK DATA PACKETS RECEIVED FROM
    ;*TU58. IF AN ERROR IS DETECTED, LOCATION PKTERR WILL INDICATE A DATA
    ;*PACKET FAILURE FOR THE PACKET RECEIVED. TABLE DERTBL WILL CONTAIN THE
    ;*ERROR BITS FOR THE FAILING PACKET. TABLE DATPNT WILL CONTAIN THE
    ;*STARTING ADDRESSES OF THE DATA PACKETS AS STORED IN THE MEMORY BUFFER.
    ;*BIT DEFINITIONS OF DATA PACKET ERROR FLAG AS STORED IN DERTBL ARE:
    ;*
    ;*BIT14=1      PACKET CHECKSUM FAILURE
    ;*BIT13=1      DATA CHECKSUM FAILURE (READ NOT EQUAL WRITE CHECKSUM)
    ;*BIT12=1      PACKET BYTE COUNT READ NOT EQUAL EXPECTED
    ;*
    ;*RETURNS
    ;*
    ;*CALL+2      DEVICE ERROR
    ;*CALL+4      GOOD RETURN (PACKET ERRORS MAY BE PENDING)
    ;*
    ;*****
    
```

```

8206 010140 010046      DATPKT: MOV     R0,-(SP)      ;SAVE THE WORKING REGISTERS
8207 010142 010146      MOV     R1,-(SP)
8208 010144 010246      MOV     R2,-(SP)
8209 010146 010346      MOV     R3,-(SP)
8210 010150 010446      MOV     R4,-(SP)
8211 010152 122737 000002 007030      CMPB    #TREAD,OPCODE  ;CHECK IF FUNCTION WAS READ
8212 010160 001404                BEQ     1$              ;IF FUNCTION IS READ THEN CONTINUE
    
```

8213	010162	052737	004000	007064	BIS	#BIT11,TU58R	:SET BIT INDICATING DATA PACKET DURING WRITE
8214	010170	000542			BR	12\$:RETURN CALL+2 DEVICE ERROR
8215	010172	005737	007072		1\$: TST	DATAFL	:CHECK IF FIRST PACKET
8216	010176	001025			BNE	2\$:IF NOT FIRST TIME CONTINUE
8217	010200	012737	007140	007106	MOV	#WRTPNT,WRKSUM	:TABLE POINTER TO WRITE PACKET ADDRESSES
8218	010206	012737	007126	007110	MOV	#DERTBL,WRKERR	:TABLE POINTER TO THIS PACKETS ERROR
8219	010214	012737	007116	007112	MOV	#DATPNT,WRKPNT	:TABLE POINTER TO PACKETS STARTING ADDRESS
8220	010222	012737	010000	007114	MOV	#BIT12,DERFLG	:SET PACKET ERROR FLAG TO 1ST PACKET
8221	010230	012700	007126		MOV	#DERTBL,RO	:SETUP TO CLEAR DATA PACKET ERROR TABLE
8222	010234	012701	000004		MOV	#4,R1	:
8223	010240	005020			20\$: CLR	(R0)+	:CLEAR THE LOCATION IN TABLE
8224	010242	005301			DEC	R1	:DECREMENT TABLE COUNTER
8225	010244	001375			BNE	20\$:NO NEXT LOCATION IF NOT 0
8226	010246	005037	007066		CLR	DPKTER	:CLEAR ACCUMULATED PACKET ERROR WORD
8227	010252	005237	007072		2\$: INC	DATAFL	:INCREMENT DATA PACKET COUNTER
8228	010256	022737	000004	007072	CMP	#4,DATAFL	:CHECK IF 4 DATA PACKETS OR LESS
8229	010264	103004			BHIS	3\$:CONTINUE IF 4 OR LESS
8230	010266	052737	040000	007064	BIS	#BIT14,TU58R	:MORE THEN 4 DATA PACKETS RECEIVED
8231	010274	000500			BR	12\$:RETURN CALL+2 DEVICE ERROR
8232	010276	013700	007056		3\$: MOV	TUBUFP,RO	:GET UPDATED BUFFER POINTER
8233	010302	005300			DEC	RO	:BACK UP TO START OF DATA PACKET
8234	010304	010077	176602		MOV	RO,@WRKPNT	:SAVE DATA PACKET POINTER IN TABLE
8235	010310	005003			CLR	R3	:CLEAR WORKING WORD
8236	010312	005004			CLR	R4	:SET CHECKSUM INITIALLY TO 0
8237	010314	012701	000101		MOV	#TMSIZ+2/2,R1	:SETUP PACKET EXPECTED WC -1
8238	010320	012702	000002		MOV	#2,R2	:SETUP BYTE COUNT FOR READING A WORD
8239	010324	000406			BR	6\$:GO ADD CHARACTER READ TO CHECKSUM
8240	010326	012702	000002		4\$: MOV	#2,R2	:SETUP BYTE COUNT FOR READING A WORD
8241	010332	005003			CLR	R3	:CLEAR THE WORKING WORD
8242	010334	004537	007622		5\$: JSR	R5,GETCHR	:GO GET A CHARACTER FROM THE BUFFER
8243	010340	000456			BR	12\$:DEVICE ERROR RETURN CALL+2
8244	010342	152003			6\$: BISB	(R0)+,R3	:ADD BYTE TO WORKING WORD
8245	010344	000303			SWAB	R3	:EXCHANGE BYTES
8246	010346	005302			DEC	R2	:DECREMENT WORD BYTE COUNT
8247	010350	001371			BNE	5\$:IF NOT WORD GET NEXT BYTE
8248	010352	005301			DEC	R1	:DECREMENT PACKET WC
8249	010354	100403			BMI	7\$:IF MINUS THEN CHECKSUM READ IS IN R3
8250	010356	060304			ADD	R3,R4	:ADD WORD TO CHECKSUM
8251	010360	005504			ADC	R4	:ADD END AROUND CARRY
8252	010362	000761			BR	4\$:GO READ NEXT WORD
8253	010364	005001			7\$: CLR	R1	:CLEAR WORKING ERROR REGISTER
8254	010366	020304			CMP	R3,R4	:COMPARE CHECKSUM CALCULATED WITH READ
8255	010370	001402			BEQ	8\$:IF EQUAL THEN CONTINUE
8256	010372	052701	040000		BIS	#BIT14,R1	:SET PACKET CHECKSUM ERROR BIT
8257	010376	017702	176504		8\$: MOV	@WRKSUM,R2	:GET ADDRESS OF WRITE PACKET
8258	010402	026203	000202		CMP	TMSIZ+2(R2),R3	:CHECK WRITE CHECKSUM WITH ONE READ
8259	010406	001402			BEQ	9\$:IF EQUAL THEN CONTINUE
8260	010410	052701	020000		BIS	#BIT13,R1	:SET DATA CHECKSUM FAILURE
8261	010414	017700	176472		9\$: MOV	@WRKPNT,RO	:GET STARTING ADDRESS OF DATA PACKET
8262	010420	105720			TSTB	(R0)+	:BUMP POINTER TO PACKET BYTE COUNT
8263	010422	122710	000200		CMPB	#TMSIZ,(R0)	:CHECK DATA PACKET BYTE COUNT
8264	010426	001402			BEQ	10\$:IF EQUAL THEN CONTINUE
8265	010430	052701	010000		BIS	#BIT12,R1	:SET PACKET BYTE COUNT ERROR BIT
8266	010434	010177	176450		10\$: MOV	R1,@WRKERR	:SAVE ERROR WORD IN ERROR TABLE
8267	010440	001403			BEQ	11\$:IF NO ERRORS DO NOT SET PKTERR
8268	010442	053737	007114	007062	BIS	DERFLG,PKTERR	:SET ERROR INDICATOR FOR THIS PACKET

```
8269 010450 012702 000002      11$:  MOV      #2,R2          ;SETUP POINTER UPDATE COUNTER
8270 010454 060237 007106      ADD      R2,WRKSUM        ;UPDATE WRITE PACKET TABLE POINTER
8271 010460 060237 007110      ADD      R2,WRKERR        ;UPDATE DATA PACKET ERROR POINTER
8272 010464 060237 007112      ADD      R2,WRKPNT        ;UPDATE DATA PACKET ADDRESS TABLE POINTER
8273 010470 006337 007114      ASL      DERFLG          ;UPDATE PACKET ERROR FLAG
8274 010474 005725              TST      (R5)+            ;BUMP POINTER PAST DEVICE ERROR RETURN
8275 010476 012604      12$:  MOV      (SP)+,R4        ;RESTORE THE ROUTINES WORKING REGISTERS
8276 010500 012603      MOV      (SP)+,R3
8277 010502 012602      MOV      (SP)+,R2
8278 010504 012601      MOV      (SP)+,R1
8279 010506 012600      MOV      (SP)+,R0
8280 010510 000205      RTS      R5              ;RETURN BACK TO PROGRAM
```

```
8281
8282
8283 :*****
8284 :*THE FOLLOWING ROUTINE WILL RECEIVE AND CHECK THE END PACKET RECEIVED
8285 :*FROM THE TU58. LOCATION PKTERR WILL CONTAIN PACKET ERROR INFORMATION
8286 :*ON EXIT FROM THIS ROUTINE.
8287 :*
8288 :*RETURN
8289 :*
8290 :*CALL+2      DEVICE ERROR
8291 :*CALL+4      END PACKET ERROR DETECTED
8292 :*CALL+6      END PACKET NO ERROR DETECTED
8293 :*
8294 :*LOCATION PKTERR BIT DEFINITIONS
8295 :*
8296 :*BIT15=1     READ PACKET 4 FAILED
8297 :*BIT14=1     READ PACKET 3 FAILED
8298 :*BIT13=1     READ PACKET 2 FAILED
8299 :*BIT12=1     READ PACKET 1 FAILED
8300 :*BIT11=1     NOT USED
8301 :*BIT10=1     NOT USED
8302 :*BIT09=1     READ OPERATION - 4 DATA PACKETS NOT RECEIVED
8303 :*BIT08=1     WRITE OPERATION - 4 CONTINUES NOT RECEIVED
8304 :*
8305 :*          END PACKET ERROR BITS
8306 :*
8307 :*BIT07=1     CHECKSUM ERROR
8308 :*BIT06=1     PACKET BYTE COUNT ERROR
8309 :*BIT05=1     OPCODE ERROR
8310 :*BIT04=1     SUCCESS CODE FAILURE
8311 :*BIT03=1     UNIT NUMBER READ INCORRECT
8312 :*BIT02=1     SEQUENCE NUMBER NOT 0
8313 :*BIT01=1     DATA BYTE COUNT WORD INCORRECT
8314 :*BIT00=1     SUMMARY WORD FAILURE
8315 :*
8316 :*****
```

```
8317 010512 010046      ENDPKT: MOV      R0,-(SP)      ;SAVE ALL WORKING REGISTERS
8318 010514 010146      MOV      R1,-(SP)
8319 010516 010246      MOV      R2,-(SP)
8320 010520 010346      MOV      R3,-(SP)
8321 010522 010446      MOV      R4,-(SP)
8322 010524 013700 007056      MOV      TUBUFP,R0        ;GET UPDATED BUFFER POINTER
8323 010530 005300      DEC      R0              ;BACK UP TO START OF END PACKET
8324 010532 010037 007060      MOV      R0,PKTPNT       ;SAVE END PACKET POINTER
```

8325	010536	005003		CLR	R3	:CLEAR WORKING WORD
8326	010540	005004		CLR	R4	:SET CHECKSUM INITIALLY TO 0
8327	010542	012701	000006	MOV	#TMSG+2/2,R1	:SETUP PACKET EXPECTED WC -1
8328	010546	012702	000002	MOV	#2,R2	:SETUP BYTE COUNT FOR READING A WORD
8329	010552	000406		BR	3\$:GO ADD THIS CHARACTER TO CHECKSUM
8330	010554	012702	000002	1\$: MOV	#2,R2	:SETUP BYTE COUNT FOR READING A WORD
8331	010560	005003		CLR	R3	:CLEAR THE WORKING WORD
8332	010562	004537	007622	2\$: JSR	R5,GETCHR	:GO GET A BYTE FROM MEMORY
8333	010566	000557		BR	17\$:DEVICE ERROR RETURN CALL+2
8334	010570	152003		3\$: BISB	(R0)+,R3	:ADD BYTE TO WORKING WORD
8335	010572	000303		SWAB	R3	:EXCHANGE IT
8336	010574	005302		DEC	R2	:DECREMENT WORD BYTE COUNT
8337	010576	001371		BNE	2\$:GO GET 2ND BYTE OF WORD
8338	010600	005301		DEC	R1	:DECREMENT PACKET WORD COUNT
8339	010602	100403		BMI	4\$:IF MINUS THE WC - CHECKSUM IN R3
8340	010604	060304		ADD	R3,R4	:ADD WORD READ TO CHECKSUM
8341	010606	005504		ADC	R4	:ADD END AROUND CARRY
8342	010610	000761		BR	1\$:GO GET NEXT TWO BYTES FROM MEMORY
8343	010612	005001		4\$: CLR	R1	:CLEAR PACKET ERROR REGISTER
8344	010614	020304		CMP	R3,R4	:CHECK FOR CHECKSUM ERROR
8345	010616	001402		BEQ	5\$:IF GOOD THEN CONTINUE
8346	010620	052701	000200	BIS	#BIT7,R1	:SET PACKET CHECKSUM ERROR BIT
8347	010624	013700	007060	5\$: MOV	PKTPT,R0	:GET POINTER TO PACKET
8348	010630	105720		TSTB	(R0)+	:BUMP POINTER TO PACKET BYTE COUNTER
8349	010632	123720	007027	CMPB	FLAG+1,(R0)+	:CHECK PACKET BYTE COUNT
8350	010636	001402		BEQ	6\$:IF OK THEN CONTINUE
8351	010640	052701	000100	BIS	#BIT6,R1	:SET PACKET BYTE COUNT ERROR BIT
8352	010644	122720	000100	6\$: CMPB	#TENDPK,(R0)+	:CHECK OPCODE TO BF END PACKET
8353	010650	001402		BEQ	7\$:IF IT IS THEN CONTINUE
8354	010652	052701	000040	BIS	#BIT5,R1	:SET NOT END PACKET OPCODE ERROR BIT
8355	010656	112037	007074	7\$: MOVB	(R0)+,SUCCES	:CHECK SUCCESS CODE BYTE TO BE 0
8356	010662	001402		BEQ	8\$:IF SUCCESS CODE = 0 THEN CONTINUE
8357	010664	052701	000020	BIS	#BIT4,R1	:SET SUCCESS CODE ERROR BIT
8358	010670	123720	007032	8\$: CMPB	UNITNM,(R0)+	:CHECK UNIT NUMBER TO BE =
8359	010674	001402		BEQ	9\$:IF SAME UNIT NUMBER THEN CONT
8360	010676	052701	000010	BIS	#BIT3,R1	:SET UNIT NUMBER ERROR BIT
8361	010702	105720		9\$: TSTB	(R0)+	:BUMP POINTER TO SEQUENCE NUMBER
8362	010704	105720		TSTB	(R0)+	:CHECK LOW BYTE TO BE 0
8363	010706	001402		BEQ	10\$:IF SEQ NUMBER = 0 THEN BRANCH
8364	010710	052701	000004	BIS	#BIT2,R1	:SET SEQUENCE NUMBER ERROR BIT
8365	010714	105720		10\$: TSTB	(R0)+	:CHECK HIGH BYTE OF SEQUENCE NUMBER
8366	010716	001402		BEQ	11\$:IF 0 THEN CONTINUE
8367	010720	052701	000004	BIS	#BIT2,R1	:SET SEQUENCE NUMBER ERROR BIT
8368	010724	111037	007076	11\$: MOVB	(R0),BYTECT	:SAVE LOW BYTE OF DATA BYTE COUNT
8369	010730	123720	007036	CMPB	BYTCNT,(R0)+	:CHECK LOW BYTE OF BYTE COUNT
8370	010734	001402		BEQ	12\$:IF GOOD THEN CONTINUE
8371	010736	052701	000002	BIS	#BIT1,R1	:SET DATA BYTE COUNT ERROR BIT
8372	010742	111037	007077	12\$: MOVB	(R0),BYTECT+1	:SAVE HIGH BYTE OF DATA BYTE COUNT
8373	010746	123720	007037	CMPB	BYTCNT+1,(R0)+	:CHECK HIGH BYE OF DATA BYTE COUNT
8374	010752	001402		BEQ	13\$:IF OK THEN CONTINUE
8375	010754	052701	000002	BIS	#BIT1,R1	:SET DATA BYTE COUNT ERROR BIT
8376	010760	112037	007100	13\$: MOVB	(R0)+,SUMARY	:SAVE LOW BYTE OF SUMMARY WORD
8377	010764	112037	007101	MOVB	(R0)+,SUMARY+1	:SAVE HIGH BYTE OF SUMMARY WORD
8378	010770	032737	170000 007100	BIT	#170000,SUMARY	:CHECK SUMMARY ERROR BITS TO BE 0
8379	010776	001402		BEQ	14\$:IF NO ERROR BITS SET CONTINUE
8380	011000	052701	000001	BIS	#BIT0,R1	:SET SUMMARY ERROR BIT

```

8381 011004 122737 000003 007030 14$: CMPB #TWRITE,OPCODE ;CHECK FUNCTION TO BE WRITE
8382 011012 001006 BNE 15$ ;IF NCT WRITE THEN EXIT
8383 011014 022737 000004 007070 CMP #4,CONTFL ;CHECK IF 4 CONTINUES WERE RECEIVED
8384 011022 001402 BEQ 15$ ;IF 4 CONTINUES THEN EXIT
8385 011024 052701 000400 BIS #BIT8,R1 ;SET BIT INDICATING INCORRECT # OF CONT
8386 011030 122737 000002 007030 15$: CMPB #TREAD,OPCODE ;CHECK IF FUNCTION WAS READ PACKET
8387 011036 001006 BNE 16$ ;IF NOT THEN SET ERROR BITS IF ANY
8388 011040 022737 000004 007072 CMP #4,DATAFL ;CHECK IF 4 DATA PACKETS RECEIVED
8389 011046 001402 BEQ 16$ ;IF YES THEN CONTINUE
8390 011050 052701 001000 BIS #BIT9,R1 ;SET BIT INDICATING DATA PACKET INCOMPLETE
8391 011054 050137 007062 16$: BIS R1,PKTERR ;SAVE END PACKET ERROR INFORMATION
8392 011060 005725 TST (R5)+ ;BUMP POINTER TO END PACKET ERROR RETURN
8393 011062 012700 007126 MOV #DERTBL,R0 ;GET ADDRESS OF DATA PACKET ERROR TABLE
8394 011066 012702 000004 MOV #4,R2 ;SETUP TO BUILD DATA PACKET ERROR WORD
8395 011072 005001 CLR R1 ;CLEAR WORKING ERROR WORD
8396 011074 052001 18$: BIS (R0)+,R1 ;SET THE ERROR BITS FROM THIS PACKET
8397 011076 005302 DEC R2 ;CHECK IF ALL 4 PACKETS BUILT
8398 011100 001404 BEQ 19$ ;IF YES THEN CHECK TO SEE IF ERRORS
8399 011102 006201 ASR R1 ;MOVE THE PACKET ERROR BITS TO POSITION
8400 011104 006201 ASR R1 ;
8401 011106 006201 ASR R1 ;
8402 011110 000771 BR 18$ ;GO GET NEXT PACKET ERROR BITS
8403 011112 010137 007066 19$: MOV R1,DPKTER ;SAVE THE ACCUMULATED DATA PACKET ERROR BITS
8404 011116 005737 007062 TST PKTERR ;CHECK IF ANY ERRORS DETECTED
8405 011122 001001 BNE 17$ ;IF YES THEN RETURN CALL+4
8406 011124 005725 TST (R5)+ ;RETURN CALL+6 NO ERROR DETECTED
8407 011126 012604 17$: MOV (SP)+,R4 ;RESTORE WORKING REGISTERS
8408 011130 012603 MOV (SP)+,R3 ;
8409 011132 012602 MOV (SP)+,R2 ;
8410 011134 012601 MOV (SP)+,R1 ;
8411 011136 012600 MOV (SP)+,R0 ;
8412 011140 000205 RTS R5 ;RETURN BACK TO THE TEST

```

```

;*****
;*THIS ROUTINE WILL DETERMINE IF A TUSB SOFT ERROR IS TO BE REPORTED.
;*LOCATION $USWR CONTAINS THE ALLOWABLE NUMBER OF SOFT ERRORS (DEFAULT
;*VALUE EQUALS 000000). THE PROGRAM WILL RETURN TO
;*CALL+2 ERROR RETURN - SOFT ERRORS ON THIS DRIVE EXCEEDS ALLOWED #
;*CALL+4 SOFT ERROR IS ALLOWED - CONT WITHOUT REPORTING THE ERROR
;*****

```

```

8421
8422 011142 005237 001256 SOFTER: INC SERTTL ;INCREMENT TOTAL SOFT ERROR COUNTER
8423 011146 005237 026334 INC PSERR ;INCREMENT EOP SOFT ERROR COUNTER
8424 011152 005737 007032 TST UNITNM ;CHECK TO SEE WHICH UNIT ERROR OCCURED
8425 011156 001011 BNE 1$ ;BR IF UNIT 1
8426 011160 005237 001260 INC SERUNO ;INCREMENT UNIT 0'S SOFT ERROR COUNTER
8427 011164 023737 001214 001260 CMP $USWR,SERUNO ;CHECK IF ALLOWED SOFT ERROR CNT EXCEEDED
8428 011172 103014 BHS 2$ ;IF $USWR >= SERUNO DON'T REPORT ERROR
8429 011174 005037 001260 CLR SERUNO ;CLEAR UNIT 0'S SOFT ERROR COUNTER
8430 011200 000412 BR 3$ ;RETURN AND REPORT ERROR
8431 011202 005237 001262 1$: INC SERUN1 ;INCREMENT UNIT 1'S COFT ERROR COUNTER
8432 011206 023737 001214 001262 CMP $USWR,SERUN1 ;CHECK IF SOFT ERROR COUNT EXCEEDED
8433 011214 103003 BHS 2$ ;IF $USWR >= SERUN1 DON'T REPORT ERROR
8434 011216 005037 001262 CLR SERUN1 ;CLEAR UNIT 1'S SOFT ERROR COUNTER
8435 011222 000401 BR 3$ ;RETURN AND REPORT ERROR
8436 011224 005725 2$: TST (R5)+ ;SOFT ERROR ALLOWED - DON'T REPORT ERROR

```

```

8437 011226 000205      3$:      RTS      R5      ;RETURN BACK TO CALL +2 OR +4
8438
8439
8440
8441      ;:.....
8442      ;      END OF PASS
8443      ;:.....
8444
8445 011230 005237 001264      EOP:      INC      KILFLG      ;SET DEVICE KILL FLAG
8446 011234 004537 013156      JSR      R5,TIMER      ;GO WAIT FOR ALL DEVICES TO SHUT DOWN
8447 011240 001266      DEVACT      ;DEVICE ACTIVE FLAG
8448 011242 000000      0      ;SHOULD GO TO 0
8449 011244 000240      NOP      ;ERROR RETURN
8450 011246 000401      BR      EOPERR      ;GO REPORT THE HUNG DEVICE
8451 011250 000443      BR      EOP1      ;NO ERROR - CHECK VIDEO PROCESSER
8452 011252 032737 000200 001266      EOPERR:  BIT      #BIT7,DEVACT      ;CHECK IF VT100 HUNG
8453 011260 001401      BEQ      1$      ;
8454 011262 104034      ERROR+34      ;VT100 HUNG
8455 011264 032737 000100 001266      1$:      BIT      #BIT6,DEVACT      ;CHECK IF CLOCK HUNG
8456 011272 001401      BEQ      2$      ;
8457 011274 104015      ERROR+15      ;CLOCK HUNG
8458 011276 032737 000040 001266      2$:      BIT      #BIT5,DEVACT      ;CHECK IF PRINTER HUNG
8459 011304 001401      BEQ      3$      ;
8460 011306 104016      ERROR+16      ;PRINTER HUNG
8461 011310 032737 000020 001266      3$:      BIT      #BIT4,DEVACT      ;CHECK IF ASYNC COMM HUNG
8462 011316 001401      BEQ      4$      ;
8463 011320 104023      ERROR+23      ;ASYNC COMM HUNG
8464 011322 032737 000010 001266      4$:      BIT      #BIT3,DEVACT      ;CHECK IF CLUSTER TERMINAL 1 HUNG
8465 011330 001401      BEQ      5$      ;
8466 011332 104017      ERROR+17      ;CLUSTER TERMINAL 1 HUNG
8467 011334 032737 000004 001266      5$:      BIT      #BIT2,DEVACT      ;CHECK IF CLUSTER TERMINAL 2 HUNG
8468 011342 001401      BEQ      6$      ;
8469 011344 104020      ERROR+20      ;CLUSTER TERMINAL 2 HUNG
8470 011346 032737 000002 001266      6$:      BIT      #BIT1,DEVACT      ;CHECK IF CLUSTER TERMINAL 3 HUNG
8471 011354 001401      BEQ      EOP1      ;IF NOT ERRORS ARE DONE FOR THIS PASS
8472 011356 104021      ERROR+21      ;CLUSTER TERMINAL 3 HUNG
8473 011360 004737 013270      EOP1:      JSR      PC,TIMER1      ;DELAY FOR OPERATOR TO SEE DISPLAY
8474 011364 004737 013270      JSR      PC,TIMER1      ;
8475 011370 005037 001264      CLR      KILFLG      ;SET KILL FLAG TO 0
8476 011374 005037 001266      CLR      DEVACT      ;CLEAR DEVICE ACTIVE FLAG
8477 011400 105737 001250      TSTB      TYPFLG      ;DOING VT100 DISPLAY TESTING?
8478 011404 001001      BNE      1$      ;BRANCH IF NOT - DECLARE END OF PASS
8479 011406 104407      SAVEVT      ;POSITION AT LINE 13 AND KEEP PRIORITY HIGH
8480
8481      1$:      ;RAISE PRIORITY IN CASE CAME FROM ABOVE
8482 011410      MOV      #PR7,-(SP)      ;;PUT NEW PS ON STACK
      (1) 011414 012746 000340      MOV      #64$,-(SP)      ;;PUT NEW PC ON STACK
      (1) 011420 000002      RTI      ;;POP NEW PC AND PS
      (1) 011422
8483 011422 104401 016236      64$:      TYPE      ,SAVCON      ;SAVE THE CURSOR POSITION AND ATTRIBUTES
8484 011426 005237 001176      INC      $PASS      ;BUMP PASS CTR
8485 011432 104401      TYPE      ;POSITION CURSOR OVER PASS COUNT (LINE 1) AND
8486 011434 016175      PASPOS      ; SET REVERSE VIDEO
8487 011436 013746 001176      MOV      $PASS,-(SP)      ;PUSH PASS COUNT ONTO STACK
8488 011442 104405      TYPDS      ;TYPE PASS COUNT IN DECIMAL
8489 011444 104401 011577      TYPE      ,RESCUR      ;RESTORE CURSOR POSITION
    
```

```

8490 011450 122737 000001 001210      CMPB  #APTENV,$ENV      ;RUNNING UNDER APT ?
8491 011456 001003                    BNE   3$               ;IF NOT ERASE SCREEN AND REPORT EOP
8492 011460 104401 016302              TYPE  ,ENDAPT         ;APT, LEAVE LAST DISPLAY PATTERN ON SCREEN
8493 011464 000402                    BR    4$               ;CONTINUE THE EOP PRINTOUT
8494 011466 104401 016277              3$:  TYPE  ,ENDPAS     ;ERASE TO END OF SCREEN AND TYPE END PASS MESSAGE
8495
8496 011472 104401 017103              4$:  TYPE  ,MSG3
8497 011476 013746 026332              MOV   PERR,-(SP)      ;;SAVE PERR FOR TYPEOUT
(1) 011502 104405                    TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
8498 011504 104401 017135              TYPE  ,MSG4
8499 011510 013746 026334              MOV   PSERR,-(SP)    ;;SAVE PSERR FOR TYPEOUT
(1) 011514 104405                    TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
8500 011516 104401 001165              TYPE  ,$CRLF         ;CARRIAGE RETURN-LINE FEED
8501 011522 104401 011574              TYPE  ,NULL          ;NULL CHAR
8502 011526 004737 013376              JSR   PC,GFLCHK      ;CHECK IF ^G TYPED DURING TYPEOUT
8503 011532 122737 000001 001210      CMPB  #APTENV,$ENV      ;RUNNING UNDER APT?
8504 011540 001003                    BNE   $GET42         ;IF NOT CHECK FOR XXDP CHAIN MODE
8505 011542 152737 000040 001211      BISB  #APTCSUP,$ENVM  ;UNCONDITIONALLY SET CONSOLE SUPPRESS BIT
8506
8507 011550 013700 000042              $GET42: MOV  @#42,R0   ;INSURE R0 CONTAINS THE MONITOR
8508 011554 001405                    BEQ   $DOAGN         ;BRANCH IF NO MONITOR
8509 011556 000005                    RESET ;CLEAR THE WORLD
8510 011560 004710                    $ENDAD: JSR  PC,(R0)  ;GO TO MONITOR
8511 011562 000240                    NOP   ;SAVE ROOM
8512 011564 000240                    NOP   ;FOR
8513 011566 000240                    NOP   ;ACT11
8514 011570 000137                    $DOAGN: JMP  @PC)+    ;RETURN
8515 011572 002374                    $RTRN: .WORD  LOOP
8516
8517 011574      377      377      000  NULL: .BYTE  -1,-1,0  ;NULL CHAR STRING
8518 011577      033      070      000  RESCUR: .BYTE  ESC,70,0 ;RESTORE CURSOR (<ESC>8)
8519
8520      .SBTTL PROGRAM SUBROUTINES
8521
8522 011602 012737 013420 000004  SIZE:  MOV   #INTSRV,ERRVEC ;SETUP TIMEOUT TRAP ADDR
8523 011610 012737 000340 000006      MOV   #PR7,ERRVEC+2 ;LOCKOUT INTERR
8524 011616 005037 013460              CLR   INTFLG
8525 011622 012700 017776              MOV   #17776,R0      ;SIZE MEM. START WITH 4K
8526 011626 005001                    CLR   R1              ;MEM MAP, 1=4K, 2=8K, ETC
8527 011630 005037 013460              CLR   INTFLG
8528 011634 005710                    16$:  TST  (R0)
8529 011636 005737 013460              TST  INTFLG          ;GOT TIMEOUT INTERR?
8530 011642 001007                    BNE   17$            ;BR IF YES
8531 011644 005201                    INC   R1
8532 011646 020027 157776              CMP   R0,#157776     ;DONE 28K MEM?
8533 011652 001405                    BEQ   10$            ;BRANCH IF YES
8534 011654 062700 020000              ADD   #20000,R0      ;GO TO NEXT 4K
8535 011660 000765                    BR    16$
8536 011662 162700 020000              17$:  SUB   #20000,R0    ;RESET R0 BACK TO HIGHEST NON-TIMED OUT ADDR
8537 011666 010037 012640              10$:  MOV   R0,MAXMEM  ;SAVE
8538 011672 006301                    ASL   R1              ;MULT BY 2
8539 011674 016137 012620 011704      MOV   MEMTBL-2(R1),18$
8540 011702 104401
8541 011704 000000                    18$:  .WORD  0           ;MEM SIZE
8542 011706 104401 016655              TYPE  ,MEM
8543 011712 004737 013376              JSR   PC,GFLCHK      ;CHECK IF ^G TYPED DURING TYPEOUT
  
```

```

8544
8545 ;CHECK TO SEE IF CLUSTER OPTION PRESENT BY ADDRESSING CSR REGISTER.
8546 ;IF ONE CSR ADDRESS NOT PRESENT, THEN CLUSTER OPTION NOT PRESENT.
8547
8548 011716 005037 012612 CLR CLPRES ;CLEAR FLAGS INDICATING CLUSTER NOT PRESENT
8549 011722 005000 CLR RO ;
8550 011724 005037 013460 CLR INTFLG ;CLEAR PROGRAM INTERRUPT FLAG
8551 011730 012700 176500 MOV #TK1S,RO ;SETUP 1ST CLUSTER CSR ADDRESS
8552 011734 005710 1$: TST (RO) ;TEST CLUSTER CSR REGISTER
8553 011736 005737 013460 TST INTFLG ;CHECK TO SEE IF TIMEOUT
8554 011742 001006 BNE 2$ ;IF YES THEN OPTION NOT PRESENT
8555 011744 020027 176520 CMP RO,#TK3S ;CHECK TO SEE IF DONE ALL 3
8556 011750 001412 BEQ 3$ ;IF YES THEN CLUSTER PRESENT
8557 011752 062700 000010 ADD #10,RO ;UPDATE CSR ADDRESS TO NEXT CLUSTER CSR
8558 011756 000766 BR 1$ ;GO TRY THIS CSR ADDRESS
8559 011760 104401 016433 2$: TYPE ,CLOPT ;TYPE CLUSTER OPTION NOT PRESENT
8560 011764 104401 016676 TYPE ,NOTPRES ;
8561 011770 004737 013376 JSR PC,GFLCHK ;CHECK IF CONTROL G TYPED
8562 011774 000402 BR 4$ ;GO CHECK TO SEE IF TUS8 PRESENT
8563 011776 005237 012612 3$: INC CLPRES ;INDICATE CLUSTER OPTION PRESENT
8564
8565 ;THE FOLLOWING CODE WILL CHECK TO SEE IF THE TUS8 IS PRESENT
8566
8567 012002 005037 012620 4$: CLR TUPRES ;CLEAR FLAG INDICATING TUS8 NOT PRESENT
8568 012006 005037 013460 CLR INTFLG ;CLEAR PROGRAM INTERRUPT FLAG
8569 012012 042737 000100 177170 BIC #IE,TURCSR ;CLEAR RECEIVER INTERRUPT ENABLE
8570 012020 005737 013460 TST INTFLG ;CHECK TO SEE IF TIMEOUT OCCURED
8571 012024 001407 BEQ 8$ ;IF NOT THEN OPTION PRESENT
8572 012026 104401 016452 TYPE ,TUS8M ;TYPE TUS8 NOT PRESENT
8573 012032 104401 016676 TYPE ,NOTPRES ;
8574 012036 004737 013376 JSR PC,GFLCHK ;CHECK IF CONTROL G TYPED
8575 012042 000402 BR 9$ ;GO RESET TIMEOUT VECTOR
8576 012044 005237 012620 8$: INC TUPRES ;SET TUS8 PRESENT FLAG
8577 012050 012737 000006 000004 9$: MOV #ERRVEC+2,ERRVEC ;RESTORE TIMEOUT VECTOR
8578 012056 005037 000006 CLR ERRVEC+2 ;
8579
8580 ;THE FOLLOWING CODE WILL CHECK TO SEE IF THE PRINTER OPTION IS PRESENT.
8581 ;AND SET DTR IF ASYNC COMM IS TO BE IN EXTERNAL LOOPBACK
8582
8583 012062 005037 176610 CLR AMRC ;CLEAR COMM CSR REGISTER
8584 012066 052737 000016 176610 BIS #<SXMIT!RTS!DTR>,AMRC ;TOGGLE BITS TO CLEAR COMM
8585 012074 042737 000016 176610 BIC #<SXMIT!RTS!DTR>,AMRC ;
8586 012102 005737 176610 TST AMRC ;READ TO CLEAR BITS THAT MAY SET
8587 012106 012737 037014 177420 MOV #<AMOD!B9600!CHAR8>,PARAM ;SETUP COMM PARAMETER REG
8588 012114 005037 012614 CLR PRPRES ;CLEAR PRINTER PRESENT FLAG
8589 012120 005037 012616 CLR PRPORT ;CLEAR PRINTER EXT LOOPBACK FLAG
8590 012124 005737 177514 TST PRS ;CHECK FOR PRINTER ERROR
8591 012130 100402 BMI 6$ ;IF ERROR - CHECK EXTERNAL LOOPBACK MODE
8592 012132 005237 012614 INC PRPRES ;SET PRINTER PRESENT FLAG
8593 012136 032737 000010 001246 6$: BIT #BIT3,$DEVN ;CHECK IF COMM EXTERNAL LOOPBACK
8594 012144 001407 BEQ 7$ ;IF YES THEN DO FURTHER CHECKS
8595 012146 052737 000006 176610 BIS #<RTS!DTR>,AMRC ;SET RTS AND DTR FOR INTERNAL LOOPBACK
8596 012154 005737 012614 TST PRPRES ;WAS THE PRINTER PRESENT
8597 012160 001027 BNE 15$ ;IF YES THEN CHECK DROP DEVICE FLAGS
8598 012162 000420 BR 5$ ;GO PRINT PRINTER NOT PRESENT
8599 012164 012737 000002 176610 7$: MOV #DTR,AMRC ;SET COMM DTR TO SET EXTERNAL LOOPBACK

```

8600	012172	005737	176610			TST	AMRC		:TO CLEAR ANY BITS THAT MAY SET
8601	012176	005737	012614			TST	PRPRES		:CHECK TO SEE IF PRINTER ALREADY PRESENT
8602	012202	001016				BNE	15\$:IF YES THEN CHECK DROP DEVICE BITS
8603	012204	005737	177514			TST	PRS		:CHECK PRINTER AGAIN FOR ERROR
8604	012210	100405				BMI	5\$:IF ERROR THEN PRINTER NOT PRESENT
8605	012212	005237	012616			INC	PRPORT		:SET PRINTER EXT LOOPBACK FLAG
8606	012216	005237	012614			INC	PRPRES		:SET PRINTER PRESENT FLAG
8607	012222	000406				BR	15\$:GO CHECK DROP DEVICE BITS
8608	012224	104401	016561	5\$:		TYPE	,PRINT		:TYPE PRINTER NOT PRESENT
8609	012230	104401	016676			TYPE	,NOTPRES		:
8610	012234	004737	013376			JSR	PC,GFLCHK		:CHECK IF CONTROL G TYPED WHILE PRINTING
8611									
8612	012240	032737	100000	001246	15\$:	BIT	#BIT15,\$DEV		:DROP PRINTER TEST?
8613	012246	001406				BEQ	19\$:BR IF NO
8614	012250	104401	016561			TYPE	,PRINT		
8615	012254	104401	017023			TYPE	,DROP		
8616	012260	004737	013376			JSR	PC,GFLCHK		:CHECK IF ^G TYPED DURING TYPEOUT
8617									
8618	012264	032737	010000	001246	19\$:	BIT	#BIT12,\$DEV		:DROP CLUSTER TERM #1?
8619	012272	001410				BEQ	20\$:BR IF NO
8620	012274	104401	016465			TYPE	,CLTERM		
8621	012300	104401	016510			TYPE	,T1		
8622	012304	104401	017023			TYPE	,DROP		
8623	012310	004737	013376			JSR	PC,GFLCHK		:CHECK IF ^G TYPED DURING TYPEOUT
8624									
8625	012314	032737	020000	001246	20\$:	BIT	#BIT13,\$DEV		:DROP CLUSTER TERM #2?
8626	012322	001410				BEQ	21\$:BR IF NO
8627	012324	104401	016465			TYPE	,CLTERM		
8628	012330	104401	016512			TYPE	,T2		
8629	012334	104401	017023			TYPE	,DROP		
8630	012340	004737	013376			JSR	PC,GFLCHK		:CHECK IF ^G TYPED DURING TYPEOUT
8631									
8632	012344	032737	040000	001246	21\$:	BIT	#BIT14,\$DEV		:DROP CLUSTER TERM #3?
8633	012352	001410				BEQ	22\$:BR IF NO
8634	012354	104401	016465			TYPE	,CLTERM		
8635	012360	104401	016514			TYPE	,T3		
8636	012364	104401	017023			TYPE	,DROP		
8637	012370	004737	013376			JSR	PC,GFLCHK		:CHECK IF ^G TYPED DURING TYPEOUT
8638									
8639	012374	032737	004000	001246	22\$:	BIT	#BIT11,\$DEV		:DROP ASYNC COMM TEST?
8640	012402	001406				BEQ	23\$:BR IF NO
8641	012404	104401	016546			TYPE	,ACOM		
8642	012410	104401	017023			TYPE	,DROP		
8643	012414	004737	013376			JSR	PC,GFLCHK		:CHECK IF ^G TYPED DURING TYPEOUT
8644									
8645	012420	032737	002000	001246	23\$:	BIT	#BIT10,\$DEV		:DROP SYNC COMM TEST?
8646	012426	001406				BEQ	24\$:BR IF NO
8647	012430	104401	016534			TYPE	,SCOM		
8648	012434	104401	017023			TYPE	,DROP		
8649	012440	004737	013376			JSR	PC,GFLCHK		:CHECK IF ^G TYPED DURING TYPEOUT
8650									
8651	012444	032737	000400	001246	24\$:	BIT	#BIT8,\$DEV		:DROP TUSB UNIT 0 TESTS?
8652	012452	001406				BEQ	25\$:BR IF NO
8653	012454	104401	016773			TYPE	,DRVO		
8654	012460	104401	017023			TYPE	,DROP		
8655	012464	004737	013376			JSR	PC,GFLCHK		:CHECK IF ^G TYPED DURING TYPEOUT

```

8656
8657 012470 032737 001000 001246 25$: BIT #BIT9,$DEV  ;DROP TU58 UNIT 1 TESTS?
8658 012476 001406 BEQ 26$ ;BR IF NO
8659 012500 104401 017007 TYPE ,DRV1
8660 012504 104401 017023 TYPE ,DROP
8661 012510 004737 013376 JSR PC,GFLCHK ;CHECK IF ^G TYPED DURING TYPEOUT
8662
8663 012514 032737 000200 001246 26$: BIT #BIT7,$DEV  ;DROP CLOCK TESTS?
8664 012522 001406 BEQ 27$ ;BR IF NO
8665 012524 104401 016571 TYPE ,CLOCK
8666 012530 104401 017023 TYPE ,DROP
8667 012534 004737 013376 JSR PC,GFLCHK ;CHECK IF ^G TYPED DURING TYPEOUT
8668
8669 012540 032737 000100 001246 27$: BIT #BIT6,$DEV  ;DROP VT00 TEST ?
8670 012546 001406 BEQ 28$ ;IF NOT THEN EXIT
8671 012550 104401 016577 TYPE ,VT100 ;TYPE VT100 DROPPED
8672 012554 104401 017023 TYPE ,DROP
8673 012560 004737 013376 JSR PC,GFLCHK ;CHECK IF CONTROL G TYPED
8674 012564 032737 000020 001246 28$: BIT #BIT4,$DEV  ;CHECK IF MEMORY TEST TO BE DROPPED
8675 012572 001406 BEQ 29$ ;IF NOT THEN EXIT AND START TEST
8676 012574 104401 017057 TYPE ,MEMORY ;TYPE MEMORY TESTS TESTING DROPPED
8677 012600 104401 017023 TYPE ,DROP
8678 012604 004737 013376 JSR PC,GFLCHK ;CHECK IF CONTROL G TYPED
8679 012610 000207 29$: RTS PC ;RETURN BACK TO PROGRAM
8680
8681 012612 000000 CLPRES: 0 ;CLUSTER PRESENT = NON-ZERO
8682 012614 000000 PRPRES: 0 ;PRINTER PRESENT = NON-ZERO
8683 012616 000000 PRPORT: 0 ;1=PRINTER ENABLED THROUGH LOOPBACK
8684 012620 000000 TUPRES: 0 ;TU58 PRESENT = NON-ZERO
8685 012622 016623 MEMTBL: M4K ;MSG ADDRESSES
8686 012624 016626 M8K
8687 012626 016631 M12K
8688 012630 016635 M16K
8689 012632 016641 M20K
8690 012634 016645 M24K
8691 012636 016651 M30K
8692 012640 000000 MAXMEM: 0 ;MAX MEMORY
    
```

```

8694
8695
8696
8697
8698
8699
8700
8701
8702
8703 012642 010046
8704 012644 012500
8705 012646 012520
8706 012650 012720 000340
8707 012654 012520
8708 012656 012710 000340
8709 012662 012600
8710 012664 000205
8711
8712
8713
8714
8715
8716
8717
8718
8719
8720
8721
8722
8723
8724
8725
8726
8727 012666 012537 014346
8728 012672 012537 014016
8729 012676 105037 014012
8730 012702 005037 014112
8731 012706 005037 014014
8732 012712 105037 014013
8733 012716 005037 001270
8734 012722 004537 012642
8735 012726 000060
8736 012730 013622
8737 012732 014020
8738 012734 052777 000100 166206
8739 012742 004537 013156
8740 012746 014112
8741 012750 000001
8742 012752 104034
8743 012754 000423
8744
8745 012756 004537 013156
8746 012762 014014
8747 012764 000001
8748 012766 000405
8749 012770 000240

```

```

*****
; ROUTINE TO SET 2 CONSECUTIVE INTERRUPT VECTORS
; 3 ARGUMENTS ARE PASSED THRU R5:
; VECTOR ADDRESS
; INTERRUPT SERVICE ROUTINE ADDRESS FOR RECVR
; INTERRUPT SERVICE ROUTINE ADDRESS FOR XMITTER.
; PRIORITY WILL BE SET TO DISABLE FURTHER INTERR.
*****
SETVEC: MOV R0,-(SP) ;SAVE
MOV (R5)+,R0 ;GET VECTOR ADDR
MOV (R5)+,(R0)+ ;PUT SERV ROUTINE ADDR THERE
MOV #PR7,(R0)+ ;DISABLE INTERR
MOV (R5)+,(R0)+ ;GET XMIT SERV ADDR NEXT
MOV #PR7,(R0)
MOV (SP)+,R0 ;RESTORE
RTS R5

```

```

*****
; THIS SUBROUTINE IS USED TO SEND AND ACCEPT AN ESCAPE SEQUENCE FORM THE VT100.
; IT ACCEPTS TWO ARGUMENTS: 1) POINTER TO TEXT TO SEND, AND 2) POINTER TO TEXT TO
; EXPECT BACK FROM VT100. THE EXPECTED RECEIVED TEXT MAY CONTAIN A NEGATIVE
; BYTE (BIT 7 SET) TO FLAG THAT ANY CHARACTER IS ACCEPTABLE AS A MATCH IN THAT
; POSITION. THE ACTUAL CHARACTER RECEIVED IS STORED IN 'OPTCHR'. (NOTE THAT ONLY
; THE LAST SUCH CHARACTER WILL BE SAVED.) THREE RETURNS ARE USED, AS FOLLOWS:
; JSR R5,VTALK
; ARG1 ; POINTER TO TEXT TO SEND
; ARG2 ; POINTER TO TEXT TO EXPECT BACK
; RETURN1 ; XMITTER HUNG (ERROR HAS BEEN REPORTED)
; RETURN2 ; RECEIVER HUNG OR UNRECOGNIZED RESPONSE (ERROR REPORTED)
; RETURN3 ; EVERYTHING OK
*****
VTALK: MOV (R5)+,TXTPTR ;SET POINTER TO SEND TEXT
MOV (R5)+,RCVPTR ;SET POINTER TO EXPECTED RECEIVED TEXT
CLRB OPTCHR ;CLEAR OPTIONAL CHARACTER
CLR XMTDUN ;CLEAR DONE FLAG FOR XMITTER
CLR RCVDUN ; AND FOR RECEIVER
CLRB ESEQFL ;NO ESCAPE SEQUENCE RECEIVED YET
CLR DELFLG ;CLEAR PAUSE COUNT FLAG
JSR R5,SETVEC ;SET UP INTERRUPT VECTORS
TKVEC
TKSRV2
TPSRV2
BIS #IE,@$TIPS ;SET XMIT INTERRUPT ENABLE (RCVR ALREADY SET)
JSR R5,TIMER ;WAIT FOR XMITTER INT SERV TO FLAG IT IS DONE
XMTDUN
1
ERROR+34 ;VT100 NOT GETTING XMIT INTERRUPTS
BR 5$
JSR R5,TIMER ;WAIT FOR RECEIVER TO FINISH RECEIVING
RCVDUN
1
BR 1$ ;ERROR RETURN
NOP ;NEED TWO WORDS FOR GOOD RETURN

```

8750								
8751	012772	105737	014015		TSTB	RCVDUN+1		;CHECK IF RESPONSE WAS UNRECOGNIZED
8752	012776	001410			BEQ	38		;BRANCH IF NOT -- ALL OK
8753	013000	000405			BR	28		;UNRECOGNIZED ESCAPE SEQ
8754								
8755	013002	105737	014013	18:	TSTB	ESEQFL		;CHECK IF ANY CHARACTER WERE RECEIVED
8756	013006	001002			BNE	28		;BR IF YES--REPORT UNRECOGNIZED RESPONSE
8757	013010	104025			ERROR+25			;NO RESPONSE FROM VT100
8758	013012	000403			BR	48		
8759	013014	104032		28:	ERROR+32			;UNRECOGNIZED RESPONSE FROM VT100
8760	013016	000401			BR	48		
8761	013020	005725		38:	TST	(R5)+		;BUMP RETURN TWICE FOR OK RETURN
8762	013022	005725		48:	TST	(R5)+		;BUMP ONLY ONCE FOR RCVR HUNG OR BAD RESPONSE
8763	013024	000205		58:	RTS	R5		

```

8765
8766
8767
8768
8769
8770
8771
8772
8773
8774 013026 113737 001250 013102 T.SAVT: MOVB TYPFLG,SAVFLG ;SAVE THE STATE OF TYPFLG; DON'T NEED TO DO
8775 013034 001021 BNE 2$ ; ANYTHING IF DISPLAY PROCESS NOT ACTIVE
8776 013036 032777 000200 166104 1$: BIT #RDY,@STPS ;WAIT FOR READY BEFORE CLEARING INT ENA
8777 013044 001774 BEQ 1$ ;
8778 013046 042777 000100 166074 BIC #IE,@STPS ;CLEAR XMIT INTERRUPT ENABLE
8779 013054 016637 000002 013104 MOV 2(SP),SAVPSW ;SAVE THE CALLING PRIORITY
8780 013062 052766 000340 000002 BIS #PR7,2(SP) ;FORCE HIGH PRIORITY ON RTI
8781 013070 105237 001250 INCB TYPFLG ;INSURE NO SAVE OR RESTORE CURSOR ISSUED BY "TYPE"
8782 013074 104401 016241 TYPE ,SAVCUR ;SAVE CURSOR AND ATTRIBUTES
8783 013100 000002 2$: RTI ;RETURN
8784
8785 013102 000000 SAVFLG: .WORD 0
8786 013104 000000 SAVPSW: .WORD 0
8787
8788 013106 004737 013376 T.RSVT: JSR PC,GFLCHK ;CHECK IF ^G TYPED DURING TYPEOUT
8789 013112 105737 013102 TSTB SAVFLG ;CHECK IF ANYTHING WAS DONE ON SAVEVT CALL
8790 013116 001016 BNE 2$ ;BRANCH IF NOT--DON'T DO ANYTHING NOW EITHER
8791 013120 104401 016264 TYPE ,RESDSP ;RESTORE CURSOR AND ATTRIBUTES AND DISPLAY AREA
8792 013124 032777 000200 166016 1$: BIT #RDY,@STPS ;CHECK TRANSMIT READY BIT
8793 013132 001774 BEQ 1$ ;WAIT FOR IT TO BECOME READY BEFORE SETTING INT ENA
8794 013134 052777 000100 166006 BIS #IE,@STPS ;RESTORE XMIT IE
8795 013142 105337 001250 DECB TYPFLG ;RESTORE FLAG TO PREVIOUS VALUE
8796 013146 013766 013104 000002 MOV SAVPSW,2(SP) ;RESTORE ORIGINAL SAVEVT CALLER'S PRIORITY
8797 013154 000002 2$: RTI
8798
8799
8800
8801
8802
8803
8804
8805
8806 013156 012537 013264 TIMER: MOV (R5)+,FLGHLD ;GET FLAG
8807 013162 012537 013266 MOV (R5)+,CTHLD ;GET COUNT THAT FLAG SHOULD GO TO
8808
8809 013166 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
8810 (2) 013170 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
8811 013172 012701 000005 MOV #5,R1
8812 013176 005000 4$: CLR R0
8813 013200 005737 001264 1$: TST KILFLG ;IS IT EOP ?
8814 013204 001405 BEQ 5$ ;IF NOT DO NORMAL TIMER
8815 013206 027737 000052 013266 CMP @FLGHLD,CTHLD ;ALL DEVICES DEAD YET ?
8816 013214 001414 BEQ 2$ ;IF YES THEN EXIT
8817 013216 000404 BR 6$ ;OTHERWISE INCREMENT TIMER
8818 013220 027737 000040 013266 5$: CMP @FLGHLD,CTHLD ;RESPONDING?
8819 013226 103007 BHIS 2$ ;BR IF YES
8819 013230 005300 6$: DEC R0 ;ELSE DEC COUNTER
    
```

```

8820 013232 001362          BNE      1$          ;BR IF NOT TIMED OUT
8821 013234 005237 001200    INC      $DEVCT      ;FOR APT TO INDICATE PRG RUNNING
8822 013240 005301          DEC      R1
8823 013242 001355          BNE      4$
8824 013244 000404          BR       3$          ;ELSE TIMED OUT
8825
8826 013246 062705 000004    2$:     ADD      #4,R5          ;JUMP OVER ERROR ON RETURN
8827 013252 005237 001200    INC      $DEVCT      ;FOR APT
8828 013256          3$:
(2) 013256 012601          MOV      (SP)+,R1      ;;POP STACK INTO R1
(2) 013260 012600          MOV      (SP)+,R0      ;;POP STACK INTO R0
8829 013262 000205          RTS      R5
8830
8831 013264 000000          FLGHLD: 0
8832 013266 000000          CTHLD:  0
8833
8834          ;;*****
8835          :     GENERAL TIMER
8836          ;;*****
8837
8838 013270 010046          TIMER1: MOV     RO,-(SP)      ;SAVE RO ON STACK
8839 013272 005000          CLR      RO
8840 013274 005300          DEC      RO
8841 013276 001376          BNE      -2
8842 013300 012600          MOV      (SP)+,RO      ;RESTORE RO
8843 013302 000207          RTS      PC
8844
8845
8846          ;;*****
8847          ;WATCHDOG TIMER TO PREVENT DEVICES FROM ENTERING AN ENDLESS WAIT LOOP
8848          ;RETURN      IF TIMED OUT
8849          ;RETURN +4 IF NOT TIMED OUT & JUMP OVER ERROR
8850          ;;*****
8851
8852 013304 012537 013372          TIMER2: MOV     (R5)+,REGHLD  ;GET REG
8853 013310 012537 013374          MOV     (R5)+,BITHLD      ;GET BIT TO BE TESTED
8854
8855 013314 010046          MOV     RO,-(SP)          ;;PUSH RO ON STACK
(2) 013316 010146          MOV     R1,-(SP)          ;;PUSH R1 ON STACK
8856 013320 012701 000002          MOV     #2,R1
8857 013324 005000          4$:     CLR      RO
8858 013326 033777 013374 000036 1$:     BIT     BITHLD,@REGHLD    ;BIT THERE?
8859 013334 001007          BNE     2$              ;BR IF YES
8860 013336 005300          DEC     RO              ;ELSE DEC COUNTER
8861 013340 001372          BNE     1$              ;BR IF NOT TIMED OUT
8862 013342 005237 001200          INC     $DEVCT          ;FOR APT TO INDICATE PRG RUNNING
8863 013346 005301          DEC     R1
8864 013350 001365          BNE     4$
8865 013352 000404          BR      3$              ;ELSE TIMED OUT
8866
8867 013354 062705 000004    2$:     ADD     #4,R5          ;JUMP OVER ERROR ON RETURN
8868 013360 005237 001200    INC     $DEVCT          ;FOR APT
8869 013364          3$:
(2) 013364 012601          MOV     (SP)+,R1        ;;POP STACK INTO R1
(2) 013366 012600          MOV     (SP)+,R0        ;;POP STACK INTO R0
8870 013370 000205          RTS      R5
    
```

8871
8872 013372 000000
8873 013374 000000

REGHLD: 0
BITHLD: 0

8874
8875
8876
8877
8878
8879
8880

; THIS SUBROUTINE MUST BE CALLED WITH THE DISPLAY PROCESS DISABLED--
; I.E. WITH THE CURSOR IN NORMAL MESSAGE POSITION. (E.G. EITHER BEFORE
; IT IS STARTED, FROM RESTVT TRAP CALL, OR AFTER IT HAS BEEN STOPPED (EOP).)
;*****

8881
8882 013376 105737 001252
8883 013402 001405
8884 013404 104401 025257
8885 013410 104406
8886 013412 105037 001252
8887 013416 000207

GFLCHK: TSTB GFLAG ;CHECK IF CONTROL-G RECEIVED WHILE TYPING
BEQ 1\$;BRANCH IF NOT
TYPE ,SCNTLG
GTSWR
CLR B GFLAG
1\$: RTS PC

```

8889          .SBTTL  INTERRUPT HANDLERS
8890
8891          ;*****
8892          ; GENERAL SERVICE ROUTINE TO BUMP 'INTFLG'
8893          ; CALLING ROUTINE WILL KNOW WHAT TO DO
8894          ;*****
8895
8896 013420 005237 013460      INTSRV: INC      INTFLG
8897 013424 005737 001264      TST      KILFLG      ;IS IT TIME TO KILL THE CLOCK
8898 013430 001412              BEQ      1$          ;IF 0 THEN NOT TIME YET
8899 013432 032737 000200 001246  BIT      #BIT7,$DEVM  ;CHECK IF CLOCK WAS ACTIVE
8900 013440 001006              BNE      1$          ;IF NOT THEN EXIT
8901 013442 042737 000100 177546  BIC      #IE,CLK      ;CLEAR THE CLOCKS INTERRUPT ENABLE BIT
8902 013450 042737 000100 001266  BIC      #BIT6,DEVACT ;SET THE DEVICE TO INACTIVE STATE
8903 013456 000002      1$:      RTI          ;RETURN BACK TO PROGRAM
8904
8905 013460 000000      INTFLG: 0
8906
8907          ;*****
8908          ; STANDARD CONSOLE RECEIVER INTERRUPT SERVICE ROUTINE
8909          ; HANDLES XON, XOFF, AND CONTROL-G
8910          ;*****
8911
8912 013462 017746 165460      TKSRV1: MOV      @STKB,-(SP) ;SAVE CHARACTER ON STACK
8913 013466 042716 177600      BIC      #^C177,(SP) ;STRIP ALL BUT ASCII
8914 013472 021627 000021      CMP      (SP),#XON  ;IS IT AN XON?
8915 013476 001013              BNE      1$          ;BRANCH IF NOT
8916 013500 105037 001253      CLRB     XFLAG      ;ALLOW TYPEOUTS
8917 013504 105737 014344      TSTB     XIEFLG     ;REQUESTED TO SET XMIT INTERRUPT ENABLE?
8918 013510 001442              BEQ      TKSIEN     ;BRANCH IF NOT
8919 013512 052777 000100 165430  BIS      #IE,@STPS
8920 013520 105037 014344      CLRB     XIEFLG     ;AND TURN OFF FLAG
8921 013524 000434              BR       TKSIEN     ;POP CHARACTER AND EXIT
8922 013526 022716 000023      1$:      CMP      #XOFF,(SP) ;CHECK IF CHARACTER WAS XOFF
8923 013532 001004              BNE      2$          ;BRANCH IF NOT
8924 013534 112737 000001 001253  MOVB     #1,XFLAG   ;DO NOT ALLOW OUTPUT
8925 013542 000425              BR       TKSIEN     ;POP CHARACTER AND EXIT
8926 013544 122737 000001 001134  2$:      CMPB     #1,$AUTOB ;AUTO MODE?
8927 013552 001421              BEQ      TKSIEN     ;BRANCH IF YES--IGNORE ALL OTHER INPUT
8928 013554 021627 000007      CMP      (SP),#7    ;WAS CHARACTER A CONTROL-G?
8929 013560 001407              BEQ      3$          ;IF YES THEN TAKE CARE OF IT
8930 013562 021627 000003      CMP      (SP),#3    ;CHECK IF CONTROL C
8931 013566 001013              BNE      TKSIEN     ;IF NOT THEN EXIT
8932 013570 004737 013270      JSR      PC,TIMER1 ;DELAY TO ALLOW DEVICES TO SHUTUP
8933 013574 000137 001716      JMP      @#START    ;RESTART THE PROGRAM
8934 013600 104407      3$:      SAVEVT   ;SAVE VT100 STATE
8935 013602 104401 025257      TYPE     ,SCNTLG    ;TYPE '^G'
8936 013606 104406              GTSWR      ;ALLOW SWITCH REGISTER CHANGE
8937 013610 105037 001252      CLRB     GFLAG     ;IN CASE ^G TYPED WHILE TYPING SWR MSG
8938 013614 104410              RESTVT    ;RESTORE VT100 STATE
8939 013616 005726      TKSIEN: TST      (SP)+ ;POP CHARACTER OFF STACK
8940 013620 000002      RTI
    
```



```

8998 014042 117746 000300      1$:   MOVB   @TXTPTR,-(SP)   ;GET THE CHARACTER TO TYPE
8999 014046 001410              BEQ    2$                ;BRANCH IF TERMINATOR
9000 014050 112737 000001 001251  MOVB   #1,ESCFLG        ;FLAG AN ESCAPE SEQUENCE
9001 014056 012677 165070      MOV    (SP)+,@$TPB      ;TYPE CHARACTER
9002 014062 005237 014346      INC    TXTPTR           ;BUMP POINTER TO NEXT CHARACTER
9003 014066 000002              RTI
9004
9005 014070 042777 000100 165052 2$:   BIC    #IE,$$TPS       ;TURN OFF INTERRUPTS
9006 014076 005237 014112      INC    XMTDUN           ;FLAG THAT I AM DONE
9007 014102 105037 001251      CLRB  ESCFLG           ;DONE SENDING SEQUENCE
9008 014106 005726              TST   (SP)+            ;POP TERMINATOR OFF STACK
9009 014110 000002              RTI
9010
9011 014112 000000              XMTDUN: .WORD 0
9012
9013      ;*****
9014      ; VT100 TRANSMIT INTERRUPT SERVICE ROUTINE. THIS DISPLAY PROCESS
9015      ; LOOKS FOR SPECIAL CHARACTERS FOLLOWING A 0 TERMINATOR. IF THE
9016      ; HIGH ORDER BIT OF THE BYTE IS SET, THEN IT IS TAKEN AS A NEGATIVE
9017      ; COUNT FOR A "PAUSE" TO GIVE TIME TO VIEW THE DISPLAY. TWO 0
9018      ; BYTES IN A ROW TERMINATE THE ENTIRE DISPLAY TEXT.
9019      ;*****
9020
9021 014114 032777 000200 165026  TPSRV: BIT    #RDY,$$TPS   ;CHECK TO SEE IF READY SET - NEEDED FOR
9022                                     ;FALSE INTERRUPTS
9023 014122 001431              BEQ    9$                ;IF NOT THEN RETURN BACK TO PROGRAM
9024 014124 105737 001253      TSTB  XFLAG            ;OK TO XMIT?
9025 014130 001406              BEQ    1$                ;BRANCH IF YES
9026 014132 105237 014344      INCB  XIEFLG           ;REQUEST RECEIVER INT SERV TO TURN IE BACK ON WHEN XON R
9027 014136 042777 000100 165004  BIC    #IE,$$TPS       ;TURN OFF INT ENABL
9028 014144 000002              RTI                       ;EXIT--RETURN WHEN XON RECEIVED
9029 014146 005737 001270      1$:   TST   DELFLG        ;CHECK IF PAUSE COUNT IN PROGRESS
9030 014152 001063              BNE   6$                ;IF IT IS SEND DELETE CHARACTER
9031 014154 117746 000166      MOVB  @TXTPTR,-(SP)     ;GET THE CHARACTER
9032 014160 001414              BEQ    4$                ;BRANCH IF TERMINATOR
9033 014162 122716 000033      2$:   CMPB  #ESC,(SP)    ;IS IT AN ESCAPE?
9034 014166 001002              BNE   3$                ;BRANCH IF NOT
9035 014170 105237 001251      INCB  ESCFLG           ;SET ESCAPE SEQUENCE FLAG
9036 014174 012677 164752      3$:   MOV    (SP)+,$$TPB  ;TYPE THE CHARACTER
9037 014200 005237 014346      INC    TXTPTR           ;BUMP POINTER TO THE NEXT CHARACTER
9038 014204 000002              8$:   RTI                       ;RETURN TO INTERRUPTED PROCESS
9039 014206 000240              9$:   NOP
9040 014210 000002              RTI                       ;FALSE INTERRUPT RETURN BACK TO PROGRAM
9041
9042 014212 005237 014346      4$:   INC    TXTPTR        ;SKIP OVER TERMINATOR
9043 014216 105037 001251      CLRB  ESCFLG           ;FLAG THAT ESCAPE SEQUENCE IS FINISHED
9044 014222 117716 000120      MOVB  @TXTPTR,(SP)     ;REPLACE TERMINATOR ON STACK WITH NEXT CHARACTER
9045 014226 100424              BMI   5$                ;BRANCH IF A PAUSE COUNT
9046 014230 001354              BNE   2$                ;BRANCH IF NOT 2ND TERMINATOR (TYPE CHARACTER)
9047 014232 012737 021467 014346  MOV    #DSPTXT-1,TXTPTR ;RESET TEXTPOINTER TO DISPLAY TEXT
9048 014240 012737 000001 001254  MOV    #1,VTDONE       ;FLAG THAT PROCESS HAS COMPLETED A PASS
9049 014246 112716 177777      MOVB  #-1,(SP)         ;REPLACE TERMINATOR ON STACK WITH A PAUSE COUNT
9050 014252 005737 001264      TST   KILFLG           ;IS IT EOP TIME YET
9051 014256 001410              BEQ    5$                ;IF NOT REPEAT TEST
9052 014260 042737 000200 001266  BIC    #BIT7,DEVACT     ;SET VT100 INACTIVE
9053 014266 105726              TSTB  (SP)+            ;POP TERMINATOR OFF THE STACK
    
```

```

9054 014270 042777 000100 164652      BIC      #IE,@$TPS      ;CLEAR TRANSMITTER INTERRUPT ENABLE
9055 014276 000421                BR        7$           ;RETURN BACK TO THE PROGRAM
9056
9057 014300 105726                5$:      TSTB      (SP)+      ;POP TERMINATOR OFF THE STACK
9058 014302 005237 014346                INC      TXTPTR      ;BUMP POINTER PAST PAUSE CONTROL
9059 014306 012737 177777 001270                MOV      #-1,DELFLG   ;SET PAUSE CONTROL FLAG
9060 014314 012737 001750 001272                MOV      #1000.,DELCNT ;SETUP DELAY COUNTER FOR ABOUT 1 SEC
9061 014322 112777 000177 164622 6$:      MOVB     #177,@$TPS   ;SEND DELETE TO VT100
9062 014330 005337 001272                DEC      DELCNT      ;DECREMENT DELAY COUNTER
9063 014334 001002                BNE     7$           ;IF NOT 0 THEN EXIT
9064 014336 005037 001270                CLR     DELFLG      ;CLEAR PAUSE CONTROL FLAG
9065 014342 000002                7$:      RTI          ;RETURN FROM INTERRUPT
9066
9067 014344      000                XIEFLG: .BYTE 0      ;FLAG FOR RECEIVER INT SERV TO TURN ON XMIT IE ON XON
9068      014346                .EVEN
9069 014346 000000                TXTPTR: .WORD 0     ;POINTER TO TEXT BEING TYPED
9070
  
```

```

9072
9073
9074
9075
9076 014350 013737 177514 014516 PRSRV: MOV PRS,SPRS ;STORE
9077 014356 100002 BPL 1$ ;BR IF ERROR BIT NOT SET
9078 014360 1040C7 ERROR+7 ;PRINTER STATUS ERROR
9079 014362 000002 RTI
9080 014364 013737 014512 177516 1$: MOV PRCHR,PRB ;ELSE PRINT CHAR
9081 014372 023727 014512 000015 CMP PRCHR,#CR ;JUST DID A CARRIAGE RETURN?
9082 014400 001413 BEQ 2$ ;BRANCH IF YES
9083 014402 023727 014512 000012 CMP PRCHR,#LF ;JUST DID LF?
9084 014410 001413 BEQ 3$ ;BR IF YES
9085 014412 023727 014512 000176 CMP PRCHR,#176 ;DID LAST CHAR?
9086 014420 001413 BEQ 4$ ;BR IF YES
9087 014422 005237 014512 INC PRCHR ;BUMP CHAR
9088 014426 000430 BR 5$
9089
9090 014430 012737 000012 014512 2$: MOV #LF,PRCHR ;DO LINE FEED NEXT
9091 014436 000424 BR 5$
9092 014440 012737 000040 014512 3$: MOV #40,PRCHR ;START ON LOOP VALUES NEXT
9093 014446 000420 BR 5$
9094 014450 012737 000015 014512 4$: MOV #CR,PRCHR ;DO CR AND START OVER
9095 014456 005237 014514 INC LINCT ;COUNT THE LINE
9096 014462 005237 001200 INC $DEVCT ;COUNTER FOR APT INDICATES RUNNING
9097 014466 005737 001264 TST KILFLG ;TIME TO KILL THE PRINTER
9098 014472 001406 BEQ 5$ ;IF NOT THEN EXIT FOR NEXT INTERRUPT
9099 014474 042737 000100 177514 BIC #IE,PRS ;CLEAR PRINTER INTERRUPT ENABLE BIT
9100 014502 042737 000040 001266 BIC #BITS,DEVACT ;SET PRINTER INACTIVE FLAG
9101 014510 000002 5$: RTI
9102
9103 014512 000000 PRCHR: 0 ;CHAR TO BE PRINTED
9104 014514 000000 LINCT: 0 ;LINE CTR
9105 014516 000000 SPRS: 0 ;STORE PRINTER CSR
9106
9107
9108
9109
9110
9111 014520 042737 000100 176504 TP1SRV: BIC #IE,TP1S ;DISABLE INTER, RECV'R WILL TURN BACK ON
9112 014526 113737 014630 176506 MOVB T1CHR,TP1B ;PRINT CHAR
9113 014534 000002 RTI
9114
9115 014536 113737 176502 014632 TK1SRV: MOVB TK1B,T1HLD ;STORE CHAR
9116 014544 123737 014632 014630 CMPB T1HLD,T1CHR ;OK?
9117 014552 001401 BEQ 1$ ;BR IF YES
9118 014554 104010 ERROR+10 ;CHAR COMP ERROR
9119
9120 014556 105237 014630 1$: INCB T1CHR ;UPDATE THE CHARACTER
9121 014562 001016 BNE 3$ ;IF NOT DONE RETURN AND PRINT NEXT CHAR
9122 014564 005237 014634 INC T1CNT ;INCREMENT LOOP COUNT
9123 014570 005237 001200 INC $DEVCT ;COUNTER FOR APT INDICATES RUNNING
9124 014574 005737 001264 TST KILFLG ;TIME TO KILL CLUSTER
9125 014600 001407 BEQ 3$ ;NO GO DO NEXT CHARACTER
9126 014602 042737 000100 176500 BIC #IE,TK1S ;CLEAR RECEIVER INTERRUPT ENABLE
9127 014610 042737 000010 001266 BIC #BIT3,DEVACT ;CLEAR CLUSTER 1 ACTIVE FLAG
    
```

 :PRINTER INTERRUPT HANDLER: VALUES FROM 40 THRU 176.

 :CLUSTER TERMINAL #1 INTERRUPT HANDLER: LOOP VALUES 0 THRU 377.

9128	014616	000403				BR	48		:RETURN BACK TO PROGRAM
9129	014620	052737	000100	176504	38:	BIS	#IE,TP1S		:ALLOW PRINTER INTERR
9130	014626	000002			48:	RTI			
9131									
9132	014630	000000			T1CHR:	0			:CHAR TO XMITT
9133	014632	000000			T1HLD:	0			:REC'D CHAR
9134	014634	000000			T1CNT:	.WORD	0		
9135	014636	000000			CTPARM:	.WORD	0		:CONTAINS BAUD RATE, 8 CHAR BIT, AND MAINT BIT

```

9137
9138
9139
9140
9141 014640 042737 000100 176514 TP2SRV: BIC #IE,TP2S ;DISABLE INTER, RECV'R WILL TURN BACK ON
9142 014646 113737 014750 176516 MOVB T2CHR,TP2B ;PRINT CHAR
9143 014654 0000C2 RTI
9144
9145 014656 113737 176512 014752 TK2SRV: MOVB TK2B,T2HLD ;STORE CHAR
9146 014664 123737 014752 014750 CMPB T2HLD,T2CHR ;OK?
9147 014672 001401 BEQ 1$ ;BR IF YES
9148 014674 104011 ERROR+11 ;CHAR COMP ERROR
9149
9150 014676 105237 014750 1$: INCB T2CHR ;UPDATE THE CHARACTER
9151 014702 001016 BNE 3$ ;IF NOT DONE SETUP TO DO NEXT CHAR
9152 014704 005237 014754 INC T2CNT ;INCREMENT LOOP COUNT
9153 014710 005237 001200 INC $DEVCT ;INCREMENT COUNTER FOR APT
9154 014714 005737 001264 TST KILFLG ;TIME TO KILL CLUSTER
9155 014720 001407 BEQ 3$ ;NO GO DO NEXT CHARACTER
9156 014722 042737 000100 176510 BIC #IE,TK2S ;CLEAR RECEIVER INTERRUPT ENABLE
9157 014730 042737 000004 001266 BIC #BIT2,DEVACT ;CLEAR CLUSTER 2 ACTIVE FLAG
9158 014736 000403 BR 4$ ;RETURN BACK TO PROGRAM
9159 014740 052737 000100 176514 3$: BIS #IE,TP2S ;ALLOW PRINTER INTERR
9160 014746 000002 4$: RTI
9161
9162 014750 000000 T2CHR: 0 ;CHAR TO XMITT
9163 014752 000000 T2HLD: 0 ;REC'D CHAR
9164 014754 000000 T2CNT: .WORD 0
9165
9166
9167
9168
9169
9170 014756 042737 000100 176524 TP3SRV: BIC #IE,TP3S ;DISABLE INTER, RECV'R WILL TURN BACK ON
9171 014764 113737 015066 176526 MOVB T3CHR,TP3B ;PRINT CHAR
9172 014772 000002 RTI
9173
9174 014774 113737 176522 015070 TK3SRV: MOVB TK3B,T3HLD ;STORE CHAR
9175 015002 123737 015070 015066 CMPB T3HLD,T3CHR ;OK?
9176 015010 001401 BEQ 1$ ;BR IF YES
9177 015012 104012 ERROR+12 ;CHAR COMP ERROR
9178
9179 015014 105237 015066 1$: INCB T3CHR ;UPDATE THE CHARACTER
9180 015020 001016 BNE 3$ ;IF NOT DONE GO TRANSMIT THIS CHAR
9181 015022 005237 015072 INC T3CNT ;INCREMENT LOOP COUNT
9182 015026 005237 001200 INC $DEVCT ;INCREMENT COUNTER FOR APT
9183 015032 005737 001264 TST KILFLG ;TIME TO KILL THE DEVICE
9184 015036 001407 BEQ 3$ ;NO GO DO NEXT CHARACTER
9185 015040 042737 000100 176520 BIC #IE,TK3S ;CLEAR RECEIVER INTERRUPT ENABLE
9186 015046 042737 000002 001266 BIC #BIT1,DEVACT ;CLEAR CLUSTER 3 ACTIVE FLAG
9187 015054 000403 BR 4$ ;RETRUN BACK TO PROGRAM
9188 015056 052737 000100 176524 3$: BIS #IE,TP3S ;ALLOW PRINTER INTERR
9189 015064 000002 4$: RTI
9190
9191 015066 000000 T3CHR: 0 ;CHAR TO XMITT
9192 015070 000000 T3HLD: 0 ;REC'D CHAR

```

CVKDBAO PDT-11 110/130 SYS EX. MACY11 30A(1052) 09-APR-79 16:30 F 7 PAGE 80-12
CVKDBA.P11 09-APR-79 08:20 INTERRUPT HANDLERS

SEQ 0083

9193 015072 000000

T3CNT: .WORD 0

```

9195
9196
9197
9198
9199 015074 042737 000100 176614 AMXSRV: BIC #IE,AMXC ;DISABLE INTER, RECV'R WILL TURN BACK ON
9200 015102 113737 015244 176616 MOV COMCHR,AMXB ;XMITT CHAR
9201 015110 0000C2 RTI
9202
9203 015112 113737 176612 015246 AMRSRV: MOVB AMRB,COMHLD ;STORE CHAR
9204 015120 123737 015246 015244 CMPB COMHLD,COMCHR ;OK?
9205 015126 001401 BEQ 1$ ;BR IF YES
9206 015130 104013 ERROR+13 ;CHAR COMPARE ERROR
9207
9208 015132 105237 015244 1$: INCB COMCHR ;UPDATE THE CHARACTER
9209 015136 001036 BNE 3$ ;GO TRANSMIT THIS CHAR IF NOT LAST
9210 015140 005237 015250 INC COMCNT ;INCREMENT LOOP COUNTER
9211 015144 005237 001200 INC $DEVCT ;COUNTER FOR APT INDICATING DEVICE RUNNING
9212 015150 005737 001264 TST KILFLG ;TIME TO KILL ASYN COMM
9213 015154 001427 BEQ 3$ ;IF NOT ALLOW NEXT XMIT INTERRUPT
9214 015156 042737 000020 001266 BIC #BIT4,DEVACT ;CLEAR ASYNC COMM ACTIVE FLAG
9215 015164 042737 000100 176614 BIC #IE,AMXC ;DISABLE XMIT INTERRUPTS
9216 015172 042737 000100 176610 BIC #IE,AMRC ;DISABLE RECEIVER INTERRUPTS
9217 015200 012737 037014 177420 MOV #<AMOD!B9600!CHAR8>,PARAM ;TAKE OUT OF INTERNAL LOOPBACK
9218 ;IF IT WAS SELECTED
9219 015206 032737 000010 001246 BIT #BIT3,$DEVN ;CHECK FOR EXT LOOPBACK UNDER APT
9220 015214 001012 BNE 4$ ;IF NOT EXTERNAL EXIT
9221 015216 052737 000006 176610 BIS #<RTS!DTR>,AMRC ;TOGGLE RTS TO CLEAR EXT LOOPBACK
9222 015224 042737 000004 176610 BIC #RTS,AMRC ;
9223 015232 000403 BR 4$ ;
9224 015234 052737 000100 176614 3$: BIS #IE,AMXC ;ALLOW XMIT INTERR
9225 015242 000002 4$: RTI
9226
9227 015244 000000 COMCHR: 0 ;XMIT CHAR
9228 015246 000000 COMHLD: 0 ;REC'D CHAR
9229 015250 000000 COMCNT: .WORD 0
9230
9231
9232
9233
9234
9235 015252 042737 000100 176624 SMXSRV: BIC #IE,SMXC ;DISABLE INTER, RECV'R WILL TURN BACK ON
9236 015260 013737 015244 176626 MOV COMCHR,SMXB ;XMITT CHAR
9237 015266 023727 015244 000377 CMP COMCHR,#377 ;LAST CHAR?
9238 015274 001021 BNE 2$ ;BR IF NO
9239 015276 005737 015342 TST LSTCHR ;WAS DUMMY PAD CHARACTER TRANSMITTED?
9240 015302 001006 BNE 1$ ;BRANCH IF YES
9241 015304 005237 015342 INC LSTCHR ;NO--FLAG THAT IS JS ABOUT TO BE
9242 015310 052737 000100 176624 BIS #IE,SMXC ;ALLOW INTERRUPT SO CAN SHUT DOWN FAST
9243 015316 000002 RTI
9244
9245 015320 042737 000020 176624 1$: BIC #SEND,SMXC ;NO MORE TO SEND
9246 015326 042737 000004 176620 BIC #RTS,SMRC ;DROP REQUEST TO SEND
9247 015334 005237 001200 INC $DEVCT ;COUNTER FOR APT TO INDICATE DEVICE RUNNING
9248 015340 000002 2$: RTI
9249
9250 015342 000000 LSTCHR. .WORD 0
    
```

 :ASYNC COMM PORT INTERRUPT HANDLER: LOOP VALUES 0 THRU 377.

 :SYNC COMM PORT INTERRUPT HANDLER: LOOP VALUES 27 THRU 377.

```

9251
9252
9253 015344 013737 176622 015246 SMRSRV: MOV SMRB,COMHLD ;STORE CHAR
9254 015352 123727 015246 000026 CMPB COMHLD,#026 ;IGNORE SYNC CHARS
9255 015360 001430 BEQ 4$
9256 015362 023737 015246 015244 CMP COMHLD,COMCHR ;OK?
9257 015370 001401 BEQ 1$ ;BR IF YES
9258 015372 104014 ERROR+14 ;CHAR COMPARE ERROR
9259
9260 015374 023727 015244 000377 1$: CMP COMCHR,#377 ;LAST CHAR?
9261 015402 001403 BEQ 2$ ;BR IF YES
9262 015404 005237 015244 INC COMCHR ;ELSE BUMP
9263 015410 000411 BR 3$
9264
9265 015412 012737 177777 015244 2$: MOV #-1,COMCHR ;INDICATE DONE
9266 015420 042737 000100 176620 BIC #IE,SMRC ;ALL DONE
9267 015426 005237 001200 INC $DEVCT ;COUNTER FOR APT INDICTES DEVICE RUNNING
9268 015432 000403 BR 4$
9269 015434 052737 000100 176624 3$: BIS #IE,SMXC ;ALLOW XMIT INTERR
9270 015442 000002 4$: RTI
9271
9272
9273
9274 ;*****
9275 ;*TU58 TRANSMIT INTERRUPT SERVICE ROUTINE
9276 ;*LOCATIONS TUXPNT AND TUXCNT MUST BE SETUP PRIOR TO TURNING THE INTERRUPT
9277 ;*ENABLE BIT ON IN THE TU58 XCSR REGISTER. WHEN TUXCNT GOES TO ZERO OR
9278 ;*UPON AN ERROR CONDITION, THE INTERRUPT ENABLE BIT WILL BE CLEARED.
9279 ;*LOCATION TU58ER WILL CONTAIN HARDWARE ERROR STATUS. BIT 4 SET INDICATES
9280 ;*AN INTERRUPT OCCURED WITH RDY BIT CLEARED. BIT 3 SET INDICATES AN
9281 ;*INTERRUPT OCCURED WHEN INTERRUPT ENABLE WAS CLEARED.
9282 ;*****
9283 015444 010046 TUXMIT: MOV R0,-(SP) ;SAVE THE WORKING REGISTER
9284 015446 032737 000200 177174 BIT #RDY,TUXCSR ;CHECK TO SEE IF XMIT RDY SET
9285 015454 001004 BNE 1$ ;IF SET THEN CONTINUE
9286 015456 052737 000020 007064 BIS #BIT4,TU58ER ;SET ERROR BIT INDICATING RDY NOT SET
9287 015464 000440 BR 3$ ;EXIT INT SRV WITH INT ENABLE CLEARED
9288 015466 005737 007052 1$: TST TUXCNT ;CHECK TO SEE IF INTERRUPT EXPECTED
9289 015472 001004 BNE 2$ ;IF NOT 0 THEN EXPECTED INTERRUPT
9290 015474 052737 000010 007064 BIS #BIT3,TU58ER ;SET BIT INDICATING INTERRUPT NOT EXPECTED
9291 015502 000431 BR 3$ ;EXIT INT SRV WITH INT ENABLE CLEARED
9292 015504 005337 007052 2$: DEC TUXCNT ;DECREMENT BYTE COUNTER
9293 015510 001003 BNE 5$ ;IF NOT 0 TRANSMIT NEXT BYTE
9294 015512 042737 000100 177174 BIC #IE,TUXCSR
9295 015520 013700 007050 5$: MOV TUXPNT,R0 ;GET POINTER TO TRANSMIT BUFFER
9296 015524 112037 177172 MOVB (R0)+,TURXDB ;WRITE THE BYTE TO THE TU58
9297 015530 010037 007050 MOV R0,TUXPNT ;RESTORE THE UPDATED BUFFER POINTER
9298 015534 005737 007102 TST BRKFLG ;CHECK IF BREAK IN PROGRESS
9299 015540 001415 BEQ 4$ ;IF NOT BREAK THEN EXIT
9300 015542 022737 000002 007052 CMP #2,TUXCNT ;CHECK IF 2ND NULL CHARACTER
9301 015550 001011 BNE 4$ ;IF NOT 2ND NULL THEN EXIT
9302 015552 042737 000001 177174 BIC #BREAK,TUXCSR ;CLEAR BREAK AFTER 2ND NULL TRANSMITTED
9303 015560 005037 007102 CLR BRKFLG ;CLEAR THE BREAK SOFTWARE FLAG
9304 015564 000403 BR 4$ ;IF NOT 0 THEN EXIT
9305 015566 042737 000100 177174 3$: BIC #IE,TUXCSR ;CLEAR IE ON DONE OR ERROR CONDITION
9306 015574 012600 4$: MOV (SP)+,R0 ;RESTORE WORKING REGISTER
  
```

```

9307 015576 000002          RTI          ;RETURN BACK TO PROGRAM
9308
9309
9310          ;*****
9311          ;*TUS8 RECEIVER INTERRUPT SERVICE ROUTINE
9312          ;*THIS ROUTINE WILL READ CHARACTERS FROM THE TUS8 AND STORE THEM INTO A
9313          ;*READ MEMORY BUFFER. LOCATION TURPNT MUST BE SETUP PRIOR TO TURNING
9314          ;*THE INTERRUPT ENABLE BIT ON IN THE TUS8 RCSR REGISTER. UPON AN ERROR
9315          ;*CONDITION, THE INTERRUPT ENABLE BIT WILL BE CLEARED. LOCATION TUS8ER
9316          ;*WILL CONTAIN THE HARDWARE ERROR STATUS. BIT 6 INDICATES AN INTERRUPT
9317          ;*OCCURED WITH RECEIVER READY BIT A ZERO. BIT 5 INDICATES THE TUS8 IS
9318          ;*CONTINUOUSLY SENDING DATA AND THE RECEIVE MEMORY BUFFER IS FILLED.
9319          ;*****
9320 015600 010046          TURCVR: MOV      RO,-(SP)          ;SAVE THE WORKING REGISTER
9321 015602 032737 000200 177170      BIT      #RDONE,TURCSR      ;CHECK TO SEE IF RECEIVER DONE BIT SET
9322 015610 001004          BNE      1$                ;IF SET THEN CONTINUE
9323 015612 052737 000100 007064      BIS      #BIT6,TUS8ER      ;SET BIT INDICATING RECEIVER DONE NOT SET
9324 015620 000416          BR       3$                ;EXIT WITH INTERRUPT ENABLE CLEARED
9325 015622 013700 007054          1$:  MOV      TURPNT,RO      ;GET READ BUFFER POINTER
9326 015626 020027 031664          CMP      RO,#TURDLA        ;CHECK TO SEE IF LESS THEN LAST ADDRESS
9327 015632 101404          BLOS    2$                ;IF LESS THEN GO PUT DATA INTO MEMORY BUFFER
9328 015634 052737 000040 007064      BIS      #BIT5,TUS8ER      ;SET BUFFER FULL INDICATOR
9329 015642 000405          BR       3$                ;EXIT WITH INTERRUPT ENABLE CLEARED
9330 015644 113720 177172          2$:  MOVB    TURXDB,(RO)+    ;SAVE BYTE INTO READ BUFFER
9331 015650 010037 007054          MOV      RO,TURPNT        ;UPDATE READ BUFFER POINTER
9332 015654 000403          BR       4$                ;EXIT BACK TO PROGRAM
9333 015656 042737 000100 177170      3$:  BIC      #IE,TURCSR      ;ERROR - CLEAR INT ENA BIT
9334 015664 012600          4$:  MOV      (SP)+,RO      ;RESTORE THE WORKING REGISTER
9335 015666 000002          RTI          ;RETURN BACK TO THE PROGRAM
9336
  
```

```

9338 .SBTTL PROGRAM MESSAGES
9339
9340 ;:*****
9341 ;: MESSAGES & ERROR FORMATS
9342 ;:*****
9343
9344 .ENABL LC ;LOWER CASE MUST BE ALLOWED IN ESCAPE SEQUENCES
9345
9346 015670 036033 055433 045062 TITLE: .ASCII <ESC>/</<ESC>/[2J/<ESC>/[?4h/ ;ANSI MODE;ERASE SCREEN;SET SMOOTH SCROL
015676 055433 032077 150
9347 015703 033 037533 035461 .ASCII <ESC>/[?1;3;5;6;9l/ ;80 COLUMN, ABSOLUTE ORG, NORMAL VIDEO
015710 035463 035465 035466
015716 066071
9348 015720 055433 063463 055433 .ASCII <ESC>/[3g/<ESC>/[;9H/<ESC>/H/<ESC>/[;17H/<ESC>/H/ ;CLEAR ALL TABS.
015726 034473 015510 015510
015734 035533 033461 015510
015742 110
9349 015743 033 035533 032462 .ASCII <ESC>/[;25H/<ESC>/H/<ESC>/[;33H/<ESC>/H/ ;SET TABS AT COLUMNS 25.
015750 015510 015510 035533
015756 031463 015510 110
9350 015763 033 035533 030464 .ASCII <ESC>/[;41H/<ESC>/H/<ESC>/[;49H/<ESC>/H/ ;TABS AT COLS 41,49
015770 015510 015510 035533
015776 034464 015510 110
9351 016003 033 035533 033465 .ASCII <ESC>/[;57H/<ESC>/H/<ESC>/[;65H/<ESC>/H/<ESC>/[;73H/<ESC>/H/ ;TABS
016010 015510 015510 035533
016016 032466 015510 015510
016024 035533 031467 015510
016032 110
9352 016033 033 031133 030473 .ASCII <ESC>/[2;13r/<ESC>/[0;7m/ ;SCROLLING REGION LINES 2-13; REVERSE VI
016040 071063 055433 035460
016046 066467
9353 016050 053103 042113 040502 .ASCII *CVKDBAO PDT-11 110/130 SYS EX. 0 PASSES*
016056 020060 042120 026524
016064 030461 030440 030061
016072 030457 030063 051440
016100 051531 042440 027130
016106 020040 020040 020040
016114 020040 020060 040520
016122 051523 051505
9354 016126 020040 020040 020040 .ASCIIZ / 0 TOTAL ERRORS ( 0 SOFT)/<CRLF><ESC>/[m/ ;RETURN TO NORMAL V
016134 020060 047524 040524
016142 020114 051105 047522
016150 051522 024040 020040
016156 020040 030040 051440
016164 043117 024524 015600
016172 066533 000
9355
9356 ;PASPOS, ERRPOS, AND SERPOS ARE FOR DIRECT CURSOR ADDRESSING OF THE PASS AND ERROR
9357 ;COUNTS ON LINE 1. IF THE TITLE MESSAGE IS CHANGED, THEY MAY HAVE TO BE CHANGED ALSO.
9358 016175 033 035533 032063 PASPOS: .ASCIIZ <ESC>/[;34H/<ESC>/[0;7m/ ;DIRECT CURSOR ADDRESS TO PASS NUMBER, REVERSE
016202 015510 030133 033473
016210 000155
9359 016212 055433 032073 044070 ERRPOS: .ASCIIZ <ESC>/[;48H/<ESC>/[0;7m/ ;DIRECT CURSOR ADDRESS TO ERROR NUMBER.
016220 055433 035460 066467
016226 000
9360 016227 033 035533 034466 SERPOS: .ASCIIZ <ESC>/[;69H/ ;DIRECT CURSOR ADDRESS TO SOFT ERROR COUNT
  
```

```
016234 000110
9361
9362 016236 033433 000 SAVCON: .ASCIZ <ESC>/7/ ;SAVE CURSOR POSITION ONLY
9363 016241 033 067 SAVCUR: .ASCII <ESC>/7/ ;SAVE CURSOR AND POSITION AT LINE 13
9364 016243 033 031133 030473 LINE13: .ASCIZ <ESC>/[2;13r/<ESC>/[m/<ESC>/[13;H/ ;SCROLL 2-13;ALL ATTRIBUTES OFF;LINE
016250 071063 055433 015555
016256 030533 035463 000110
9365 016264 055433 032061 031073 RESDSP: .ASCIZ <ESC>/[14;24r/<ESC>/8/ ;SCROLL 14-24 AND RESTORE CURSOR
016272 071064 034033 000
9366 016277 033 045133
9367 016302 055433 032477 042554 ENDPAS: .ASCII <ESC>/[J/
016310 042116 050040 051501 ENDAPT: .ASCIZ <ESC>/[?5LEND PASS/ ;ERASE TO END OF SCREEN; NORMAL VIDEO; TYPE END
016316 000123
9368 016320 055433 035460 035461 BLKBLD: .ASCIZ <ESC>/[O;1;5m/ ;BLINKING AND BOLD
016326 066465 000
9369 016331 033 015474 032533 STATRQ: .ASCIZ <ESC>/</<ESC>/[5n/ ;REQUEST STATUS REPORT
016336 000156
9370 016340 055433 000143 IDENRQ: .ASCIZ <ESC>/[c/ ;REQUEST TERMINAL IDENTIFY
9371 ;*****
9372 ;RCVR INT SERV TEXT INCLUDES WILDCARD MATCH CHARACTERS OF NEGATIVE BYTES
9373 016344 055433 067377 000 DSTATR: .ASCIZ <ESC>/[<377>/n/ ;DEVICE STATUS REPORT
9374 016351 033 037533 035461 TIDNTF: .ASCIZ <ESC>/[?1;/<377>/c/ ;TERMINAL IDENTIFY (VT100)
016356 061777 000
9375 016361 033 037533 035464 VIDNTF: .ASCIZ <ESC>/[?4;/<377>/c/ ;TERMIANL IDENTIFY (VT132)
016366 061777 000
9376 ;*****
9377
9378 016371 126 030524 030060 OPTPRE: .ASCII /VT100 REPORTS OPTION PRESENT = /
016376 051040 050105 051117
016404 051524 047440 052120
016412 047511 020116 051120
016420 051505 047105 020124
016426 020075
9379 016430 000 OPCHTX: .BYTE 0 ;OPTION PRESENT PARAMETER RECEIVED PLACED HERF
9380 016431 200 000 .BYTE CRLF,0 ;<CR><Lr> AND TERMINATOR
9381 016433 103 052514 052123 CLOPT: .ASCIZ /CLUSTER OPTION/
016440 051105 047440 052120
016446 047511 000116
9382 016452 052524 034065 000 TU58M: .ASCIZ /TU58/
9383 016457 124 032525 020070 TU58MS: .ASCIZ /TU58 /
016464 000
9384 016465 103 052514 052123 CLTERM: .ASCIZ /CLUSTER TERMINAL #/
016472 051105 052040 051105
016500 044515 040516 020114
016506 000043
9385 016510 000061 T1: .ASCIZ /1/
9386 016512 000062 T2: .ASCIZ /2/
9387 016514 000063 T3: .ASCIZ /3/
9388 016516 044504 050123 040514 DSPTST: .ASCIZ /DISPLAY TESTS/
016524 020131 042524 052123
016532 000123
9389 016534 054523 041516 041440 SCOM: .ASCIZ /SYNC COMM/
016542 046517 000115
9390 016546 051501 047131 020103 ACOM: .ASCIZ /ASYN COMM/
016554 047503 046515 000
9391 016561 120 044522 052116 PRINT: .ASCIZ /PRINTER/
```

9392	016566	051105	000				
	016571	103	047514	045503	CLOCK:	.ASCIZ	/CLOCK/
	016576	000					
9393	016577	126	030524	030060	VT100:	.ASCIZ	/VT100/
	016604	000					
9394	016605	040	047520	052122	PORT:	.ASCIZ	/ PORT TESTED/<CRLF>
	016612	052040	051505	042524			
	016620	100104	000				
9395	016623	064	000113		M4K:	.ASCIZ	/4K/
9396	016626	045470	000		M8K:	.ASCIZ	/8K/
9397	016631	061	045462	000	M12K:	.ASCIZ	/12K/
9398	016635	061	045466	000	M16K:	.ASCIZ	/16K/
9399	016641	062	045460	000	M20K:	.ASCIZ	/20K/
9400	016645	062	045464	000	M24K:	.ASCIZ	/24K/
9401	016651	063	045460	000	M30K:	.ASCIZ	/30K/
9402	016655	040	042515	047515	MEM:	.ASCIZ	/ MEMORY PRESENT/<CRLF>
	016662	054522	050040	042522			
	016670	042523	052116	000200			
9403	016676	047040	052117	050040	NOTPRES:	.ASCIZ	/ NOT PRESENT/<CRLF>
	016704	042522	042523	052116			
	016712	000200					
9404	016714	044600	051516	051105	WARNING:	.ASCIZ	<CRLF>/INSERT SCRATCH CASSETTES, TYPE 'P' TO PROCEED/
	016722	020124	041523	040522			
	016730	041524	020110	040503			
	016736	051523	052105	042524			
	016744	026123	052040	050131			
	016752	020105	050047	020047			
	016760	047524	050040	047522			
	016766	042503	042105	000			
9405	016773	124	050101	020105	DRV0:	.ASCIZ	/TAPE UNIT 0/
	017000	047125	052111	030040			
	017006	000					
9406	017007	124	050101	020105	DRV1:	.ASCIZ	/TAPE UNIT 1/
	017014	047125	052111	030440			
	017022	000					
9407	017023	040	042524	052123	DROP:	.ASCIZ	/ TESTING DROPPED/<CRLF>
	017030	047111	020107	051104			
	017036	050117	042520	100104			
	017044	000					
9408	017045	040	052522	047116	RUN:	.ASCIZ	/ RUNNING/<CRLF>
	017052	047111	100107	000			
9409	017057	115	046505	051117	MEMORY:	.ASCIZ	/MEMORY TESTS/
	017064	020131	042524	052123			
	017072	000123					
9410	017074	042040	047117	100105	DUN:	.ASCIZ	/ DONE/<CRLF>
	017102	000					
411	017103	011	052011	052117	MSG3:	.ASCIZ	<HT><HT>/TOTAL ERRORS THIS PASS:/
	017110	046101	042440	051122			
	017116	051117	020123	044124			
	017124	051511	050040	051501			
	017132	035123	000				
9412	017135	200	004411	051411	MSG4:	.ASCIZ	<CRLF><HT><HT><HT>/SOFT ERRORS THIS PASS: /
	017142	043117	020124	051105			
	017150	047522	051522	052040			
	017156	044510	020123	040520			
	017164	051523	020072	000			

9413	017171	115	046505	051117	EM1:	.ASCIZ /MEMORY TIMEOUT/
	017176	020131	044524	042515		
	017204	052517	000124			
9414	017210	042515	047515	054522	EM2:	.ASCIZ /MEMORY DATA COMPARE ERROR/
	017216	042040	052101	020101		
	017224	047503	050115	051101		
	017232	020105	051105	047522		
	017240	000122				
9415	017242	054523	041516	041440	EM3:	.ASCIZ /SYNC COMM DID NOT RECEIVE SYNC CHAR/
	017250	046517	020115	044504		
	017256	020104	047516	020124		
	017264	042522	042503	053111		
	017272	020105	054523	041516		
	017300	041440	040510	000122		
9416	017306	051120	047111	042524	EM4:	.ASCIZ /PRINTER STATUS ERROR/
	017314	020122	052123	052101		
	017322	051525	042440	051122		
	017330	051117	000			
9417	017333	103	052514	052123	EM5:	.ASCIZ /CLUSTER TERM #1 DATA COMP ERR/
	017340	051105	052040	051105		
	017346	020115	030443	042040		
	017354	052101	020101	047503		
	017362	050115	042440	051122		
	017370	000				
9418	017371	103	052514	052123	EM6:	.ASCIZ /CLUSTER TERM #2 DATA COMP ERR/
	017376	051105	052040	051105		
	017404	020115	031043	042040		
	017412	052101	020101	047503		
	017420	050115	042440	051122		
	017426	000				
9419	017427	103	052514	052123	EM7:	.ASCIZ /CLUSTER TERM #3 DATA COMP ERR/
	017434	051105	052040	051105		
	017442	020115	031443	042040		
	017450	052101	020101	047503		
	017456	050115	042440	051122		
	017464	000				
9420	017465	101	054523	041516	EM8:	.ASCIZ /ASYNCH COMM DATA COMP ERR/
	017472	041440	046517	020115		
	017500	040504	040524	041440		
	017506	046517	020120	051105		
	017514	000122				
9421	017516	054523	041516	041440	EM9:	.ASCIZ /SYNC COMM DATA COMP ERR/
	017524	046517	020115	040504		
	017532	040524	041440	046517		
	017540	020120	051105	000122		
9422	017546	046103	020113	052510	EM23:	.ASCIZ /CLK HUNG/
	017554	043516	000			
9423	017557	120	044522	052116	EM24:	.ASCIZ /PRINTER HUNG/
	017564	051105	044040	047125		
	017572	000107				
9424	017574	042524	046522	021440	EM25:	.ASCIZ /TERM #1 HUNG/
	017602	020061	052510	043516		
	017610	000				
9425	017611	124	051105	020115	EM26:	.ASCIZ /TERM #2 HUNG/
	017616	031043	044040	047125		
	017624	000107				

9426	017626	042524	046522	021440	EM27:	.ASCIZ /TERM #3 HUNG/
	017634	020063	052510	043516		
	017642	000				
9427	017643	123	047131	020103	EM28:	.ASCIZ /SYNC COMM HUNG/
	017650	047503	046515	044040		
	017656	047125	000107			
9428	017662	051501	047131	020103	EM29:	.ASCIZ /ASync COMM HUNG/
	017670	047503	046515	044040		
	017676	047125	000107			
9429	017702	054523	041516	041440	EM32:	.ASCIZ /SYNC COMM - DATA NOT AVAIL NOT SET/
	017710	046517	020115	020055		
	017716	040504	040524	047040		
	017724	052117	040440	040526		
	017732	046111	047040	052117		
	017740	051440	052105	000		
9430	017745	123	047131	020103	EM33:	.ASCIZ /SYNC COMM - RECVR ACTIVE NOT SET/
	017752	047503	046515	026440		
	017760	051040	041505	051126		
	017766	040440	052103	053111		
	017774	020105	047516	020124		
	020002	042523	000124			
9431	020006	054523	041516	041440	EM34:	.ASCIZ /SYNC COMM - RECVR DONE NOT SET/
	020014	046517	020115	020055		
	020022	042522	053103	020122		
	020030	047504	042516	047040		
	020036	052117	051440	052105		
	020044	000				
9432	020045	122	046517	052040	EM127:	.ASCIZ /ROM TIMED OUT/
	020052	046511	042105	047440		
	020060	052125	000			
9433	020063	126	030524	030060	EM134:	.ASCIZ /VT100 HUNG/
	020070	044040	047125	000107		
9434	020076	047516	051040	051505	EM125:	.ASCIZ /NO RESPONSE FROM VT100/
	020104	047520	051516	020105		
	020112	051106	046517	053040		
	020120	030524	030060	000		
9435	020125	126	030524	030060	EM130:	.ASCIZ /VT100 REPORTED SELF-TEST FAILURE/
	020132	051040	050105	051117		
	020140	042524	020104	042523		
	020146	043114	052055	051505		
	020154	020124	040506	046111		
	020162	051125	000105			
9436	020166	050117	044524	047117	EM131:	.ASCIZ /OPTION PRESENT (STP) NOT DETECTED BY VT100/
	020174	050040	042522	042523		
	020202	052116	024040	052123		
	020210	024520	047040	052117		
	020216	042040	052105	041505		
	020224	042524	020104	054502		
	020232	053040	030524	030060		
	020240	000				
9437	020241	125	051116	041505	EM132:	.ASCIZ /UNRECOGNIZED RESPONSE FROM VT100/
	020246	043517	044516	042532		
	020254	020104	042522	050123		
	020262	047117	042523	043040		
	020270	047522	020115	052126		
	020276	030061	000060			

9438	020302	054523	041516	041440	EM133:	.ASCIZ	/SYNC COMM MODEM LOOPBACK SIGNALS NOT SET IN CSR/
	020310	046517	020115	047515			
	020316	042504	020115	047514			
	020324	050117	040502	045503			
	020332	051440	043511	040516			
	020340	051514	047040	052117			
	020346	051440	052105	044440			
	020354	020116	051503	000122			
9439	020362	051501	047131	020103	EM140:	.ASCIZ	/ASYNC COMM MODEM LOOPBACK SIGNALS NOT SET IN CSR/
	020370	047503	046515	046440			
	020376	042117	046505	046040			
	020404	047517	041120	041501			
	020412	020113	044523	047107			
	020420	046101	020123	047516			
	020426	020124	042523	020124			
	020434	047111	041440	051123			
	020442	000					
9440	020443	124	032525	020070	EM135:	.ASCIZ	/TU58 DEVICE ERROR/
	020450	042504	044526	042503			
	020456	042440	051122	051117			
	020464	000					
9441	020465	124	032525	020070	EM136:	.ASCIZ	/TU58 WRITE ERROR/
	020472	051127	052111	020105			
	020500	051105	047522	000122			
9442	020506	052524	034065	051040	EM137:	.ASCIZ	/TU58 READ ERROR/
	020514	040505	020104	051105			
	020522	047522	000122				
9443	020526	051105	047522	020122	DH1:	.ASCIZ	/ERROR # ERR PC/
	020534	004443	051105	020122			
	020542	041520	000				
9444	020545	105	051122	051117	DH2:	.ASCIZ	/ERROR # ERR PC ADDR/
	020552	021440	042411	051122			
	020560	050040	004503	042101			
	020566	051104	000				
9445	020571	105	051122	051117	DH3:	.ASCIZ	/ERROR # ERR PC ADDR EXPECT RECVD/
	020576	021440	042411	051122			
	020604	050040	004503	042101			
	020612	051104	042411	050130			
	020620	041505	020124	051040			
	020626	041505	042126	000			
9446	020633	105	051122	051117	DH4:	.ASCIZ	/ERROR # ERR PC PR STATUS/
	020640	021440	042411	051122			
	020646	050040	004503	051120			
	020654	051440	040524	052524			
	020662	000123					
9447	020664	051105	047522	020122	DH5:	.ASCIZ	/ERROR # ERR PC EXPECT RECVD/
	020672	004443	051105	020122			
	020700	041520	042411	050130			
	020706	041505	020124	051040			
	020714	041505	042126	000			
9448	020721	105	051122	051117	DH31:	.ASCIZ	/ERROR # ERR PC OPTPRES/
	020726	021440	042411	051122			
	020734	050040	004503	050117			
	020742	050124	042522	000123			
9449	020750	051105	047522	020122	DH32:	.ASCIZ	/ERROR # ERR PC ERR WRD OP-CODE UNIT # /
	020756	004443	051105	020122			

	021424	007032	007040	007062	
	021432	007074	007076	007100	
	021440	000000			
9467	021442	026330	001116	007030	DT36: .WORD ERRNUM,\$ERRPC,OPCODE,UNITNM,BLKNUM,PKTERR,DPKTER,SUCCESS,BYTECT,SUMARY,0
	021450	007032	007040	007062	
	021456	007066	007074	007076	
	021464	007100	000000		
9468					
9469					
9470					::*****
9471					:DISPLAY PROCESS TEXT
9472					:FOR THIS TEXT TO BE COMPATIBLE WITH THE TYPE ROUTINE, ALL ESCAPE
9473					:SEQUENCES SHOULD BE FOLLOWED BY A ZERO BYTE. ANY ZERO BYTE MAY, IF DESIRED,
9474					:BE FOLLOWED BY A NEGATIVE BYTE, WHICH IS A NEGATIVE 'PAUSE COUNT'. THIS WILL
9475					:PAUSE THE DISPLAY PROCESS (ALLOWING THE DISPLAY TO BE READ).
9476					:THE ENTIRE DISPLAY TEXT IS TERMINATED BY TWO ZERO BYTES IN A ROW
9477					:*****
9478	021470	055433	032061	031073	DSPTXT: .ASCIZ <ESC>/[14;24r/ ;SCROLL 14-24
	021476	071064	000		
9479	021501	033	031133	035464	.ASCIZ <ESC>/[24;1H/ ;CURSOR POSITION
	021506	044061	000		
9480	021511	033	030133	030473	.ASCIZ <ESC>/[0;1q/ ;LED 1 ON
	021516	000161			
9481	021520	051525	040440	041523	.ASCII /US ASCII CHARACTER SET/<CR><LF>
	021526	044511	041440	040510	
	021534	040522	052103	051105	
	021542	051440	052105	005015	
9482	021550	040	041	042	.BYTE 40,41,42,43,44,45,46,47,50,51,52,53,54,55,56,57,60,61,62,63,64,65,66,67.
	021553	043	044	045	
	021556	046	047	050	
	021561	051	052	053	
	021564	054	055	056	
	021567	057	060	061	
	021572	062	063	064	
	021575	065	066	067	
	021600	070	071	072	
	021603	073	074	075	
	021606	076	077		
9483	021610	100	101	102	.BYTE 100,101,102,103,104,105,106,107,110,111,112,113,114,115,116,117,CR,LF
	021613	103	104	105	
	021616	106	107	110	
	021621	111	112	113	
	021624	114	115	116	
	021627	117	015	012	
9484	021632	120	121	122	.BYTE 120,121,122,123,124,125,126,127,130,131,132,135,134,135,136,137,140,141.
	021635	123	124	125	
	021640	126	127	130	
	021643	131	132	133	
	021646	134	135	136	
	021651	137	140	141	
	021654	142	143	144	
	021657	145	146	147	
9485	021662	150	151	152	.BYTE 150,151,152,153,154,155,156,157,160,161,162,163,164,165,166,167,170,171.
	021665	153	154	155	
	021670	156	157	160	
	021673	161	162	163	

	021676	164	165	166		
	021701	167	170	171		
	021704	172	173	174		
	021707	175	176	015		
	021712	012	012			
9486	021714	051107	050101	044510	.ASCIZ	/GRAPHIC CHARACTER SET/<CR><LF><ESC>/(</
	021722	020103	044103	051101		;SELECT GRAPHIC CHAR SET
	021730	041501	042524	020122		
	021736	042523	006524	015412		
	021744	030050	000			
9487	021747	040	041	042	.BYTE	40,41,42,43,44,45,46,47,50,51,52,53,54,55,56,57,60,61,62,63,64,65,66,67.
	021752	043	044	045		
	021755	046	047	050		
	021760	051	052	053		
	021763	054	055	056		
	021766	057	060	061		
	021771	062	063	064		
	021774	065	066	067		
	021777	070	071	072		
	022002	073	074	075		
	022005	076	077			
9488	022007	100	101	102	.BYTE	100,101,102,103,104,105,106,107,110,111,112,113,114,115,116,117,CR,LF
	022012	103	104	105		
	022015	106	107	110		
	022020	111	112	113		
	022023	114	115	116		
	022026	117	015	012		
9489	022031	120	121	122	.BYTE	120,121,122,123,124,125,126,127,130,131,132,133,134,135,136,137,140,141.
	022034	123	124	125		
	022037	126	127	130		
	022042	131	132	133		
	022045	134	135	136		
	022050	137	140	141		
	022053	142	143	144		
	022056	145	146	147		
9490	022061	150	151	152	.BYTE	150,151,152,153,154,155,156,157,160,161,162,163,164,165,166,167,170,171.
	022064	153	154	155		
	022067	156	157	160		
	022072	161	162	163		
	022075	164	165	166		
	022100	167	170	171		
	022103	172	173	174		
	022106	175	176	015		
	022111	012	012			
9491	022113	033	041050	000	.ASCIZ	<ESC>/(</
9492	022117	374			.BYTE	374
9493	022120	005015	044514	042516	.ASCII	<CR><LF>/LINE SIZE TEST/<CR><LF>
	022126	051440	055111	020105		
	022134	042524	052123	005015		
9494	022142	005015	021433	000063	.ASCIZ	<CR><LF><ESC><43><63>
9495	022150	044124	051511	046040	.ASCII	/THIS LINE IS DOUBLE-HEIGHT DOUBLE-WIDTH/
	022156	047111	020105	051511		
	022164	042040	052517	046102		
	022172	026505	042510	043511		
	022200	052110	042040	052517		
	022206	046102	026505	044527		

9496	022214	052104	110		
	022217	015	015412	032043	.ASCIZ <CR><LF><ESC><43><64>
	022224	000			
9497	022225	124	044510	020123	.ASCII /THIS LINE IS DOUBLE-HEIGHT DOUBLE-WIDTH/<CR><LF>
	022232	044514	042516	044440	
	022240	020123	047504	041125	
	022246	042514	044055	044505	
	022254	044107	020124	047504	
	022262	041125	042514	053455	
	022270	042111	044124	005015	
9498	022276	005015	021433	000066	.ASCIZ <CR><LF><ESC><43><66>
9499	022304	044124	051511	046040	.ASCII /THIS LINE IS SINGLE-HEIGHT DOUBLE-WIDTH/<CR><LF>
	022312	047111	020105	051511	
	022320	051440	047111	046107	
	022326	026505	042510	043511	
	022334	052110	042040	052517	
	022342	046102	026505	044527	
	022350	052104	006510	012	
9500	022355	015	015412	032443	.ASCIZ <CR><LF><ESC><43><65>
	022362	000			
9501	022363	124	044510	020123	.ASCIZ /THIS LINE IS SINGLE-HEIGHT SINGLE-WIDTH/<CR><LF>
	022370	044514	042516	044440	
	022376	020123	044523	043516	
	022404	042514	044055	044505	
	022412	044107	020124	044523	
	022420	043516	042514	053455	
	022426	042111	044124	005015	
	022434	000			
9502	022435	376			.BYTE 376
9503	022436	055433	035460	070462	.ASCIZ <ESC>/[O;2q/ ;LFD 2
	022444	000			
9504	022445	015	040412	052124	.ASCII <CR><LF>/ATTRIBUTE DISPLAY TEST/<CR><LF><LF>
	022452	044522	052502	042524	
	022460	042040	051511	046120	
	022466	054501	052040	051505	
	022474	006524	005012		
9505	022500	055433	066460	000	.ASCIZ <ESC><133><60><155>
9506	022505	124	044510	020123	.ASCII /THIS LINE IS NORMAL - NO ATTRIBUTES/<CR><LF>
	022512	044514	042516	044440	
	022520	020123	047516	046522	
	022526	046101	026440	047040	
	022534	020117	052101	051124	
	022542	041111	052125	051505	
	022550	005015			
9507	022552	055433	035460	066461	.ASCIZ <ESC><133><60><73><61><155>
	022560	000			
9508	022561	124	044510	020123	.ASCII /THIS LINE IS BOLD/<CR><LF>
	022566	044514	042516	044440	
	022574	020123	047502	042114	
	022602	005015			
9509	022604	055433	035460	066464	.ASCIZ <ESC><133><60><73><64><155>
	022612	000			
9510	022613	124	044510	020123	.ASCII /THIS LINE IS UNDERLINED/<CR><LF>
	022620	044514	042516	044440	
	022626	020123	047125	042504	
	022634	046122	047111	042105	

9511	022642	005015				
	022644	055433	035460	035461	.ASCIIZ	<ESC><133><60><73><61><73><64><155>
	022652	066464	000			
9512	022655	124	044510	020123	.ASCII	/THIS LINE IS BOLD AND UNDERLINED/<CR><LF>
	022662	044514	042516	044440		
	022670	020123	047502	042114		
	022676	040440	042116	052440		
	022704	042116	051105	044514		
	022712	042516	006504	012		
9513	022717	033	030133	032473	.ASCIIZ	<ESC><133><60><73><65><155>
	022724	000155				
9514	022726	044124	051511	046040	.ASCII	/THIS LINE IS BLINKING/<CR><LF>
	022734	047111	020105	051511		
	022742	041040	044514	045516		
	022750	047111	006507	012		
9515	022755	033	030133	032473	.ASCIIZ	<ESC><133><60><73><65><73><61><155>
	022762	030473	000155			
9516	022766	055433	035460	070463	.ASCIIZ	<ESC>/[O;3q/ ;LED 3
	022774	000				
9517	022775	124	044510	020123	.ASCII	/THIS LINE IS BLINKING AND BOLD/<CR><LF>
	023002	044514	042516	044440		
	023010	020123	046102	047111		
	023016	044513	043516	040440		
	023024	042116	041040	046117		
	023032	006504	012			
9518	023035	033	030133	032473	.ASCIIZ	<ESC><133><60><73><65><73><64><155>
	023042	032073	000155			
9519	023046	044124	051511	046040	.ASCII	/THIS LINE IS BLINKING AND UNDERLINED/<CR><LF>
	023054	047111	020105	051511		
	023062	041040	044514	045516		
	023070	047111	020107	047101		
	023076	020104	047125	042504		
	023104	046122	047111	042105		
	023112	005015				
9520	023114	055433	035460	035465	.ASCIIZ	<ESC><133><60><73><65><73><61><73><64><155>
	023122	035461	066464	000		
9521	023127	124	044510	020123	.ASCII	/THIS LINE IS BLINKING, UNDERLINED AND BOLD/<CR><LF>
	023134	044514	042516	044440		
	023142	020123	046102	047111		
	023150	044513	043516	020054		
	023156	047125	042504	046122		
	023164	047111	042105	040440		
	023172	042116	041040	046117		
	023200	006504	012			
9522	023203	033	030133	033473	.ASCIIZ	<ESC><133><60><73><67><155>
	023210	000155				
9523	023212	376			.BYTE	376
9524	023213	015	052012	044510	.ASCII	<CR><LF>/THIS LINE IS REVERSE VIDEO/<CR><LF>
	023220	020123	044514	042516		
	023226	044440	020123	042522		
	023234	042526	051522	020105		
	023242	044526	042504	006517		
	023250	012				
9525	023251	033	030133	033473	.ASCIIZ	<ESC><133><60><73><67><73><61><155>
	023256	030473	000155			
9526	023262	044124	051511	046040	.ASCII	/THIS LINE IS REVERSE VIDEO AND BOLD/<CR><LF>

	023270	047111	020105	051511	
	023276	051040	053105	051105	
	023304	042523	053040	042111	
	023312	047505	040440	042116	
	023320	041040	046117	006504	
	023326	012			
9527	023327	033	030133	033473	.ASCIZ <ESC><133><60><73><67><73><64><155>
	023334	032073	000155		
9528	023340	044124	051511	046040	.ASCII /THIS LINE IS REVERSE VIDEO AND UNDERLINED/<CR><LF>
	023346	047111	020105	051511	
	023354	051040	053105	051105	
	023362	042523	053040	042111	
	023370	047505	040440	042116	
	023376	052440	042116	051105	
	023404	044514	042516	006504	
	023412	012			
9529	023413	033	030133	032073	.ASCIZ <ESC>/[O;4q/ ;LEU 4
	023420	000161			
9530	023422	055433	035460	035467	.ASCIZ <ESC><133><60><73><67><73><61><73><64><155>
	023430	035461	066464	000	
9531	023435	124	044510	020123	.ASCII /THIS LINE IS REVERSE VIDEO, BOLD AND UNDERLINED/<CR><LF>
	023442	044514	042516	044440	
	023450	020123	042522	042526	
	023456	051522	020105	044526	
	023464	042504	026117	041040	
	023472	046117	020104	047101	
	023500	020104	047125	042504	
	023506	046122	047111	042105	
	023514	005015			
9532	023516	055433	035460	035467	.ASCIZ <ESC><133><60><73><67><73><65><155>
	023524	066465	000		
9533	023527	124	044510	020123	.ASCII /THIS LINE IS REVERSE VIDEO AND BLINKING/<CR><LF>
	023534	044514	042516	044440	
	023542	020123	042522	042526	
	023550	051522	020105	044526	
	023556	042504	020117	047101	
	023564	020104	046102	047111	
	023572	044513	043516	005015	
9534	023600	055433	035460	035467	.ASCIZ <ESC><133><60><73><67><73><65><73><61><155>
	023606	035465	066461	000	
9535	023613	124	044510	020123	.ASCII /THIS LINE IS REVERSE VIDEO, BLINKING AND BOLD/<CR><LF>
	023620	044514	042516	044440	
	023626	020123	042522	042526	
	023634	051522	020105	044526	
	023642	042504	026117	041040	
	023650	044514	045516	047111	
	023656	020107	047101	020104	
	023664	047502	042114	005015	
9536	023672	055433	035460	035467	.ASCIZ <ESC><133><60><73><67><73><65><73><64><155>
	023700	035465	066464	000	
9537	023705	124	044510	020123	.ASCII /THIS LINE IS REVERSE VIDEO, BLINKING AND UNDERLINED/<CR><LF>
	023712	044514	042516	044440	
	023720	020123	042522	042526	
	023726	051522	020105	044526	
	023734	042504	026117	041040	
	023742	044514	045516	047111	

023750 020107 047101 020104
 023756 047125 042504 046122
 023764 047111 042105 005015
 9538 023772 055433 035460 035467
 024000 035465 035464 066461
 024006 000
 9539 024007 124 044510 020123
 024014 044514 042516 044440
 024022 020123 042522 042526
 024030 051522 020105 044526
 024036 042504 026117 041040
 024044 044514 045516 047111
 024052 026107 052440 042116
 024060 051105 044514 042516
 024066 020104 047101 020104
 024074 047502 042114 005015
 9540 024102 055433 066460 000
 9541 024107 376
 9542 024110 055433 000161
 9543 024114 005015 044103 047101
 024122 044507 043516 051440
 024130 051103 042505 020116
 024136 047524 041040 040514
 024144 045503 047440 020116
 024152 044127 052111 015505
 024160 037533 064065 000
 9544 024165 376
 9545 024166 041415 040510 043516
 024174 047111 020107 041523
 024202 042522 047105 052040
 024210 020117 044127 052111
 024216 020105 047117 041040
 024224 040514 045503 055433
 024232 032477 000154
 9546 024236 005015
 9547 024240 000 000

.ASCIZ <ESC><133><60><73><67><73><65><73><64><73><61><155>
 .ASCII /THIS LINE IS REVERSE VIDEO, BLINKING, UNDERLINED AND BOLD/<CR><LF>
 ATTOFF: .ASCIZ <ESC><133><60><155> ;ALL ATTRIBUTES OFF
 .BYTE 376
 .ASCIZ <ESC>/[q/ ;ALL LED'S OFF
 .ASCIZ <CR><LF>/CHANGING SCREEN TO BLACK ON WHITE/<ESC><133><77><65><150>
 .BYTE 376 ;PAUSE WHILE SCREEN STAYS REVERSED
 .ASCIZ <CR>/CHANGING SCREEN TO WHITE ON BLACK/<ESC><133><77><65><154>
 .ASCII <CR><LF>
 .BYTE 0,0 ;FINAL TERMINATOR

9548
 9549
 9550 .EVEN
 9551 .SBTTL TYPE ROUTINE

```

;*****
;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A ZERO BYTE.
;THIS ROUTINE IS FUNCTIONALLY THE SAME AS SYSMAC'S ".$TYPE" AND IS USED IN THE
;SAME WAY. IT WAS COPIED AND EDITED FROM SYSMAC. HORIZONTAL TAB PROCESSOR WAS REMOVED.
;THE SYSMAC TYPE ROUTINE COULD NOT BE USED FOR 3 REASONS:
;1) IT DID NOT SUPPORT XON, XOFF (THOUGH FUTURE VERSIONS MAY).
;2) IN ORDER TO OPERATE ALONG WITH VT100 DISPLAY PROCESS, THE TYPE ROUTINE
;   MUST FIRST ISSUE SAVE CURSOR COMMAND AND POSITION THE CURSOR AND RESTORE THE
;   CURSOR AT THE END.  THUS, THE FLOW TO TYPE A MESSAGE IS:
;       1) FIRST TYPE SAVE CURSOR MESSAGE TO SAVE CURSOR AND ATTRIBUTES AND TO
;         SET UP THE SCREEN PROPERLY;
;       2) TYPE THE MESSAGE (FROM THE ORIGINAL TYPE STATEMENT)
;       3) TYPE RESTORE CURSOR MESSAGE TO TO RETURN SCREEN TO PREVIOUS STATE.
;   TYPING THE SAVECURSOR AND RESTORE CURSOR COMMANDS IS DONE WITH A TYPE CALL
;   AND SOME FLAGS FOR HOUSEKEEPING.  THE FLOWCHARTS DEALING WITH THIS PORTION OF
;   THE CODE FOLLOWS.
  
```

```

9580
9581      ;[$TYPE]
9582      ;[1$:]   IF TYPFLG = 0
9583                THEN SET TYPFLG = -1
9584                TYPE, SAVCUR           ;TYPE THE SAVE CURSOR MESSAGE
9585      ;[20$:ETC] PROCESS (TYPE) THE MESSAGE FORM THE CALL
9586      ;[100$:] IF TYPFLG = 0
9587                THEN SET TYPFLG = -1
9588                TYPE, RESDSP           ;TYPE THE RESTORE CURSOR MESSAGE
9589                ELSE IF TYPFLG < 0 (= -1) THEN SET TYPFLG = 0
9590      ;[101$:] RETURN
9591
9592      ;THE SYMBOL TYPFLG CONTROLS THE SAVE AND RESTORE CURSOR PROCESS.
9593      ;IF TYPFLG > 0 (AND NOT= 0), THEN ESCAPE SEQUENCES TO SAVE AND RESTORE THE
9594      ;CURSOR POSITION WILL NOT BE SENT TO THE VT100. THIS WILL BE THE CASE IF VT100
9595      ;DISPLAY TESTS WERE NEVER STARTED, WHICH WILL SPEED UP TYPEOUTS. IT ALSO
9596      ;*! MUST !* BE THE CASE IF IT IS DESIRED TO TYPE OUT PORTIONS OF A LINE
9597      ;WITHOUT A CARRIAGE RETURN AT THE END. (E.G. IN THE PAUSE BETWEEN INDIVIDUAL
9598      ;DATA TYPEOUTS ON THE SAME LINE IN THE ERROR ROUTINE.) THIS CAN BE TAKEN
9599      ;CARE OF AUTOMATICALLY WITH THE "SAVEVT" AND "RESTVT" TRAP CALLS.
9600
9601      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
9602      ;*NOTE1:   $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
9603      ;*NOTE2:   $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
9604      ;*NOTE3:   $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
9605
9606      ;*CALL:
9607      ;*1) USING A TRAP INSTRUCTION
9608      ;*   TYPE   ,MESADR           ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
9609      ;*OR
9610      ;*   TYPE
9611      ;*   MESADR
9612
9613
9614      024242 105737 001251      $TYPE:  TSTB   ESCFLG           ;WAS AN ESCAPE SEQUENCE BEING TYPED BY DISPLAY PROCESS?
9615      024246 001402                BEQ     1$           ;BRANCH IF NOT
9616      024250 004737 024672                JSR     PC,FINESC   ;FINISH TYPING THE ESCAPE SEQUENCE
9617      024254 105737 001250      1$:    TSTB   TYPFLG           ;SHOULD I SAVE THE CURSOR POSITION?
9618      024260 001007                BNE     20$         ;BRANCH IF NOT
9619      024262 042777 000100 154660        BIC     #IE,@STPS   ;CLEAR XMIT IE
9620      024270 105137 001250                COMB   TYPFLG       ;SET FLAG TO -1
9621      024274 104401 016241                TYPE,  SAVCUR       ;SAVE CURSOR POSITION AND ATTRIBUTES
9622      024300 042777 000100 154636 20$:   BIC     #IE,@STKS   ;TURN OFF KEYBD IE TO AVOID SPURIOUS 8085 INTERRUPTS
9623      024306 010046                MOV     R0,-(SP)    ;;SAVE R0
9624      024310 017600 000002                MOV     @2(SP),R0   ;;GET ADDRESS OF ASCIZ STRING
9625      024314 122737 000001 001210        CMPB   #APTENV,$ENV ;RUNNING IN APT MODE
9626      024322 001011                BNE     62$         ;;NO,GO CHECK FOR APT CONSOLE
9627      024324 132737 000100 001211        BITB   #APTSPOOL,$ENVM ;SPOOL MESSAGE TO APT
9628      024332 001405                BEQ     62$         ;;NO,GO CHECK FOR CONSOLE
9629      024334 010037 024344                MOV     R0,61$     ;;SETUP MESSAGE ADDRESS FOR APT
9630      024340 004737 025530                JSR     PC,$ATY3    ;;SPOOL MESSAGE TO APT
9631      024344 000000                .WORD  0           ;;MESSAGE ADDRESS
9632      024346 132737 000040 001211 61$:   BITB   #APTCSUP,$ENVM ;APT CONSOLE SUPPRESSED
9633      024354 001003                BNE     60$         ;;YES,SKIP TYPE OUT
9634      024356 112046                2$:    MOVB   (R0)+,-(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
9635      024360 001031                BNE     4$           ;;BR IF IT ISN'T THE TERMINATOR
  
```

9636	024362	005726				TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK
9637	024364	012600			60\$:	MOV	(SP)+,R0	:: RESTORE R0
9638	024366	062716	000002		3\$:	ADD	#2,(SP)	:: ADJUST RETURN PC
9639	024372	105737	001250		100\$:	TSTB	TYPFLG	:: SHOULD I RESTORE THE CURSOR?
9640	024376	001010				BNE	101\$:: BRANCH IF NO
9641	024400	105137	001250			COMB	TYPFLG	:: SET FLAG TO -1
9642	024404	104401	016264			TYPE,	RESDSP	:: RESTORE THE CURSOR AND ATTRIBUTES AND DISPLAY AREA
9643	024410	052777	000100	154532		BIS	#IE,@STPS	:: RESTORE XMIT IE
9644	024416	000403				BR	102\$	
9645	024420	100002			101\$:	BPL	102\$:: BRANCH IF THE TYPFLG WAS NON-ZERO AND POSITIVE
9646	024422	105037	001250			CLRB	TYPFLG	:: SET THE TYPFLG TO 0
9647	024426	105737	001135		102\$:	TSTB	\$INTAG	:: CHECK INT MODE INDICATOR
9648	024432	001403				BEQ	103\$:: IF SET DO NOT SET KBD INT ENA BIT
9649	024434	052777	000100	154502		BIS	#IE,@STKS	:: RESTORE IE FOR KEYBD
9650	024442	000002			103\$:	RTI		:: RETURN
9651	024444	122716	000200		4\$:	CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
9652	024450	001010				BNE	5\$	
9653	024452	005726				TST	(SP)+	:: POP <CR><LF> EQUIV
9654	024454	105237	001250			INCB	TYPFLG	:: INSURE NO CURSOR SAVING
9655	024460	104401				TYPE		:: TYPE A CR AND LF
9656	024462	001165				\$CRLF		
9657	024464	105337	001250			DECB	TYPFLG	:: RESTORE THE FLAG
9658	024470	000732				BR	2\$:: GET NEXT CHARACTER
9659	024472	004737	024524		5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
9660	024476	123726	001156		6\$:	CMPB	\$FILLC,(SP)+	:: IS IT TIME FOR FILLER CHARS.?
9661	024502	001325				BNE	2\$:: IF NO GO GET NEXT CHAR.
9662	024504	013746	001154			MOV	\$NULL,-(SP)	:: GET # OF FILLER CHARS. NEEDED
9663								:: AND THE NULL CHAR.
9664	024510	105366	000001		7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
9665	024514	002770				BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK
9666	024516	004737	024524			JSR	PC,\$TYPEC	:: GO TYPE A NULL
9667	024522	000772				BR	7\$:: LOOP
9668								
9669	024524	032777	000200	154416	\$TYPEC:	BIT	#RDY,@STPS	:: WAIT UNTIL PRINTER IS READY
9670	024532	001774				BEQ	\$TYPEC	
9671	024534	032777	000200	154402	i\$:	BIT	#RDY,@STKS	:: CHECK IF CHARACTER TO BE READ
9672	024542	001444				BEQ	5\$:: BRANCH IF NOT
9673	024544	017746	154376			MOV	@STKB,-(SP)	:: READ CHARACTER
9674	024550	042716	177600			BIC	#^C177,(SP)	:: MASK ALL BUT ASCII
9675	024554	022716	000023			CMP	#XOFF,(SP)	:: XOFF?
9676	024560	001003				BNE	2\$:: BR IF NO
9677	024562	112737	000001	001253		MOVB	#1,XFLAG	:: FLAG XOFF STATUS
9678	024570	022716	000021		2\$:	CMP	#XON,(SP)	:: XON?
9679	024574	001004				BNE	3\$:: BR IF NO
9680	024576	105037	001253			CLRB	XFLAG	:: FLAG XON STATUS
9681	024602	105037	014344			CLRB	XIEFLG	:: CLEAR FLAG THAT IN ENABLE DESIRED (WILL BE
9682								:: SET ON RTI OR BY CALL & RESTORE SEQUENCE
9683								
9684	024606	122737	000001	001134	3\$:	CMPB	#1,\$AUTOB	:: AUTO MODE?
9685	024614	001002				BNE	33\$:: IF NOT CHECK FOR CONTROL G OR C
9686	024616	005726				TST	(SP)+	:: POP CHARACTER OFF STACK
9687	024620	000415				BR	5\$:: EXIT IF XOF FLAG NOT SET
9688	024622	022716	000007		33\$:	CMP	#7,(SP)	:: CONTROL-G?
9689	024626	001003				BNE	4\$:: BR IF NOT
9690	024630	112737	000001	001252		MOVB	#1,GFLAG	:: FLAG THAT CONTROL-G NEEDS HANDLING
9691	024636	022726	000003		4\$:	CMP	#3,(SP)+	:: CHECK IF A CONTROL C WAS TYPED

```

9692 024642 001004          BNE      5$          ;IF NOT THEN CONTINUE
9693 024644 004737 013270    JSR      PC,TIMER1    ;DELAY TO ALLOW DEVICES TO SHUTUP
9694 024650 000137 001716    JMP      @#START      ;RESTART THE PROGRAM
9695 024654 105737 001253    5$:     TSTB     XFLAG    ;WAIT FOR XON STATUS
9696 024660 001325          BNE      1$
9697 024662 016677 000002 154262  MOV     2(SP),@STPB   ;;LOAD CHAR TO BE TYPED INTO DATA REG.
9698 024670 000207    $TYPEX: RTS      PC
9699
9700
9701
9702
9703
9704 024672 105037 001251    FINESC: CLRB     ESCFLG  ;CLEAR THE ESCAPE FLAG
9705 024676 013737 014346 024724  MOV     TXTPTR,1$    ;SAVE THE POINTER TO THE UNFINISHED ESCAPE SEQUENCE
9706 024704 105237 001250          INCB     TYPFLG     ;MAKE SURE TYPE DOES NOT SAVE AND RESTORE CURSOR
9707 024710 017746 154234          MOV     @STPS,-(SP)  ;SAVE STATE OF XMIT IE
9708 024714 042777 000100 154226  BIC     #IE,@STPS    ;AND THEN CLEAR IT
9709 024722 104401          TYPE
9710 024724 000000    1$:     .WORD    0    ;POINTER TO UNFINISHED ESC SEQ
9711 024726 012677 154216    MOV     (SP)+,@STPS  ;RESTORE STATE OF XMIT IE
9712 024732 105337 001250          DECB     TYPFLG     ;RESTORE FLAG FOR CURSOR TO PREVIOUS VALUE
9713 024736 105777 167404    2$:     TSTB     @TXTPTR ;MODIFY TXTPTR TO POINT AFTER ESCAPE SEQUENCE
9714 024742 001403          BEQ     3$          ;BRANCH IF POINTING TO TERMINATOR
9715 024744 005237 014346          INC     TXTPTR      ;POINT IT TO NEXT BYTE
9716 024750 000772          BR      2$          ;GO CHECK IF AT TERMINATOR
9717 024752 000207    3$:     RTS      PC
  
```

 ;THIS SUBROUTINE FINISHES TYPING AN ESCAPE SEQUENCE STARTED BY THE XMIT INTERRUPT
 ;SERVICE ROUTINE (VT100 TESTING).

```

9719          .SBTTL  SOFTWARE SWITCH REGISTER CHANGE ROUTINE
9720          ;*****
9721          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
9722          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
9723          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
9724          ;*WHEN OPERATING IN TTY FLAG MODE.
9725
9726 024754 017737 154170 025250 $GTSWR: MOV @STPS,GSAVIE ;SAVE THE STATE OF XMIT IE
9727 024762 042777 000100 154160 BIC #IE,@STPS ;AND THEN CLEAR XMIT IE
9728 024770 005737 001274 TST DEVCHG ;CHECK IF TO BE DEVICE MAP
9729 024774 001406 BEQ 1$ ;IF 0 THEN SWITCH REGISTER CHANGE
9730 024776 104401 025306 TYPE ,MDEVN ;TYPE CURRENT CONTENTS
9731 025002 013746 001246 MOV $DEVN,-(SP) ;SAVE $DEVN FOR TYPEOUT
9732 (1) 025006 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
9733 025010 000405 BR 2$ ;GO TYPE THE NEW QUETION
9734 025012 104401 025264 1$: TYPE ,SMSWR ;TYPE CURRENT CONTENTS
9735 025016 013746 000176 MOV SWREG,-(SP) ;SAVE SWREG FOR TYPEOUT
9736 (1) 025022 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
9737 025024 104401 025275 2$: TYPE ,SMNEW ;PROMPT FOR NEW SWR
9738 025030 005046 19$: CLR -(SP) ;CLEAR COUNTER
9739 025032 005046 CLR -(SP) ;THE NEW SWR
9740 025034 105777 154104 7$: TSTB @STKS ;CHAR THERE?
9741 025040 100375 BPL 7$ ;IF NOT TRY AGAIN
9742
9743 025042 117746 154100 MOVB @STKB,-(SP) ;PICK UP CHAR
9744 025046 042716 177600 BIC #^C177,(SP) ;MAKE IT 7-BIT ASCII
9745 025052 021627 000025 9$: CMP (SP),#25 ;IS IT A CONTROL-U?
9746 025056 001005 BNE 10$ ;BRANCH IF NOT
9747 025060 104401 025252 TYPE ,%CNTLU ;YES, ECHO CONTROL-U (^U)
9748 025064 062706 000006 20$: ADD #6,SP ;IGNORE PREVIOUS INPUT
9749 025070 000757 BR 19$ ;LET'S TRY IT AGAIN
9750
9751 025072 021627 000015 10$: CMP (SP),#15 ;IS IT A <CR>?
9752 025076 001034 BNE 16$ ;BRANCH IF NO
9753 025100 005766 000004 TST 4(SP) ;YES, IS IT THE FIRST CHAR?
9754 025104 001412 BEQ 11$ ;BRANCH IF YES
9755 025106 005737 001274 TST DEVCHG ;CHECK IF DEVICE MAP CHANGE
9756 025112 001404 BEQ 3$ ;IF 0 THEN SWITCH REGISTER CHANGE
9757 025114 016637 000002 001246 MOV 2(SP),$DEVN ;SAVE NEW $DEVN
9758 025122 000403 BR 11$ ;GET READY TO EXIT
9759 025124 016677 000002 154006 3$: MOV 2(SP),@SWR ;SAVE NEW SWR
9760 025132 062706 000006 11$: ADD #6,SP ;CLEAR UP STACK
9761 025136 104401 001165 14$: TYPE ,%CRLF ;ECHO <CR> AND <LF>
9762 025142 123727 001135 000001 CMPB $INTAG,#1 ;RE-ENABLE TTY KBD INTERRUPTS!
9763 025150 001003 BNE 15$ ;BRANCH IF NOT
9764 025152 052777 000100 153764 BIS #IE,@STKS ;RE-ENABLE TTY KBD INTERRUPTS
9765 025160 013777 025250 153762 15$: MOV GSAVIE,@STPS ;RESTORE THE STATE OF XMIT IE
9766 025166 000002 RTI ;RETURN
9767 025170 004737 024524 16$: JSR PC,$TYPEC ;ECHO CHAR
9768 025174 021627 000060 CMP (SP),#60 ;CHAR < 0?
9769 025200 002420 BLT 18$ ;BRANCH IF YES
9770 025202 021627 000067 CMP (SP),#67 ;CHAR > 7?
9771 025206 003015 BGT 18$ ;BRANCH IF YES
9772 025210 042726 000060 BIC #60,(SP)+ ;STRIP-OFF ASCII
9773 025214 005766 000002 TST 2(SP) ;IS THIS THE FIRST CHAR

```

```

9773 025220 001403          BEQ      17$      ;;BRANCH IF YES
9774 025222 006316          ASL      (SP)    ;;NO, SHIFT PRESENT
9775 025224 006316          ASL      (SP)    ;;  CHAR OVER TO MAKE
9776 025226 006316          ASL      (SP)    ;;  ROOM FOR NEW ONE.
9777 025230 005266 000002   17$: INC      2(SP)  ;;KEEP COUNT OF CHAR
9778 025234 056616 177776   BIS      -2(SP),(SP) ;;SET IN NEW CHAR
9779 025240 000675          BR        7$      ;;GET THE NEXT ONE
9780 025242 104401 001164   18$: TYPE    ,SQUES ;;TYPE ?<CR><LF>
9781 025246 000706          BR        20$    ;;SIMULATE CONTROL-U
9782 025250 000000          GSAVIE: .WORD  0    ;;STATE OF XMIT IE STORED HERE
9783 025252 052536 005015    000 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
9784 025257    136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
9785 025264 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
      025272 020075    000
9786 025275    040 047040 053505 $MNEW: .ASCIZ / NEW = /
      025302 036440 000040
9787 025306 005015 042044 053105 $DEVN: .ASCIZ <15><12>/$DEVN = /
      025314 020115 020075    000
9788          025322          .EVEN

```

9790

.SBTTL APT COMMUNICATIONS ROUTINE

```

(1) (1) 025322 112737 000001 025566 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
(1) (1) 025330 112737 000001 025564 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
(1) (1) 025336 000403 BR $ATYC
(1) (1) 025340 112737 000001 025566 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) (1) 025346 $ATYC:
(3) (3) 025346 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
(3) (3) 025350 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(1) (1) 025352 105737 025564 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
(1) (1) 025356 001450 BEQ 5$ ;;IF NOT: BR
(1) (1) 025360 122737 000001 001210 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
(1) (1) 025366 001031 BNE 3$ ;;IF NOT: BR
(1) (1) 025370 132737 000100 001211 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) (1) 025376 001425 BEQ 3$ ;;IF NOT: BR
(1) (1) 025400 017600 000004 MOV @4(SP),RO ;;GET MESSAGE ADDR.
(1) (1) 025404 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) (1) 025412 005737 001170 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) (1) 025416 001375 BNE 1$ ;;IF NOT: WAIT
(1) (1) 025420 010037 001204 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
(1) (1) 025424 105720 2$: TSTB (RO)+ ;;FIND END OF MESSAGE
(1) (1) 025426 001376 BNE 2$
(1) (1) 025430 163700 001204 SUB $MSGAD,RO ;;SUB START OF MESSAGE
(1) (1) 025434 006200 ASR RO ;;GET MESSAGE LNGTH IN WORDS
(1) (1) 025436 010037 001206 MOV RO,$MSGLGT ;;PUT LENGTH IN MAILBOX
(1) (1) 025442 012737 000004 001170 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) (1) 025450 000413 BR 5$
(1) (1) 025452 017637 000004 025476 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) (1) 025460 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) (3) 025466 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) (1) 025472 004737 024242 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) (1) 025476 000000 4$: .WORD 0
(1) (1) 025500 5$:
(1) (1) 025500 105737 025566 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
(1) (1) 025504 001416 BEQ 12$ ;;IF NOT: BR
(1) (1) 025506 005737 001210 TST $ENV ;;RUNNING UNDER APT?
(1) (1) 025512 001413 BEQ 12$ ;;IF NOT: BR
(1) (1) 025514 005737 001170 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
(1) (1) 025520 001375 BNE 11$ ;;IF NOT: WAIT
(1) (1) 025522 017637 000004 001172 MOV @4(SP),$FATAL ;;GET ERROR #
(1) (1) 025530 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) (1) 025536 005237 001170 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
(1) (1) 025542 105037 025566 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
(1) (1) 025546 105037 025565 CLRB $LFLG ;;CLEAR LOG FLAG
(1) (1) 025552 105037 025564 CLRB $MFLG ;;CLEAR MESSAGE FLAG
(3) (3) 025556 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) (3) 025560 012600 MOV (SP)+,RO ;;POP STACK INTO RO
(1) (1) 025562 000207 RTS PC ;;RETURN
(1) (1) 025564 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
(1) (1) 025565 000 $LFLG: .BYTE 0 ;;LOG FLAG
(1) (1) 025566 000 $FFLG: .BYTE 0 ;;FATAL FLAG
(1) (1) 025570 .EVEN
(1) (1) 000200 APTSIZE=200
(1) (1) 000001 APTENV=001
(1) (1) 000100 APTPOOL=100
  
```

CVKDBAO PDT-11 110/130 SYS EX. MACY11 30A(1052) 09-APR-79 16:30 C 9
CVKDBA.P11 09-APR-79 08:20 APT COMMUNICATIONS ROUTINE PAGE 81-19

SEQ 0106

(1) 000040 . APTCSUP=040

```

9841 .SBTTL ERROR HANDLER ROUTINE
(1)
(2) ;*****
(1) ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1) ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) ;*AND GO TO MYTYPE ON ERROR
(1) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) ;*SW15=1 HALT ON ERROR
(1) ;*SW13=1 INHIBIT ERROR TYPEOUTS
(1) ;*SW10=1 BELL ON ERROR
(1) ;*CALL
(1) ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1) $ERROR:
(3) 025570 005237 026332 INC PERR ;COUNT THE PASS ERRORS
(3) 025574 122737 000001 001210 CMPB #APTENV,$ENV ;CHECK IF APT
(3) 025602 001024 BNE 90$ ;IF NOT DO NORMAL ERROR PROCEDURE
(3) 025604 012737 037014 177420 MOV #<AMOD!B9600!CHAR8>,PARAM ;OPEN ASYN COMM PORT TO WORLD
(3) 025612 052737 000016 176610 BIS #<RTS!SXMIT!DTR>,AMRC ;TOGGLE DTR TO CLEAR EXT LOOPBACK
(3) 025620 042737 000016 176610 BIC #<RTS!SXMIT!DTR>,AMRC ;
(3) 025626 005737 176610 TST AMRC ;CLEAR DSC/DSI BITS
(3) 025632 005037 176610 CLR AMRC ;CLEAR OUT THE INT ENA BITS FOR COMM
(3) 025636 005037 176614 CLR AMXC ;
(3) 025642 105037 001135 CLRB $INTAG ;CLEAR RECEIVER INT MODE INDICATOR
(3) 025646 042737 000100 177546 BIC #IE,CLK ;CLEAR THE CLOCKS INTERRUPT ENABLE BIT
(3) 025654 104407 90$: SAVEVT ;SAVE THE VT100 DISPLAY STATE
(3) ;*SW10=1 BLINKING AND BOLD ON ERROR
(3) 025656 032777 002000 153254 BIT #BIT10,@SWR ;CHECK SWITCH 10
(3) 025664 001406 BEQ 100$ ;BRANCH IF NOT SET
(3) 025666 032777 020000 153244 BIT #BIT13,@SWR ;TYPEOUTS INHIBITED?
(3) 025674 001002 BNE 100$ ;BRANCH IF YES
(3) 025676 104401 016320 TYPE ,BLKBLD ;BLINKING BOLD FOR ERROR TYPEOUT
(3) 025702 100$: 7$: INCB $ERFLG ;;SET THE ERROR FLAG
(1) 025702 105237 001103 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
(1) 025706 001775 MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 025710 013777 001102 153224 BIT #BIT10,@SWR ;;BELL ON ERROR?
(1) 025716 032777 002000 153214 BEQ 1$ ;;NO - SKIP
(1) 025724 001402 TYPE ,SBELL ;;RING BELL
(1) 025726 104401 001160 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
(1) 025732 005237 001112 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 025736 011637 001116 SUB #2,$ERRPC
(1) 025742 162737 000002 001116 MOVB @$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 025750 117737 153142 001114 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
(1) 025756 032777 020000 153154 BNE 20$ ;;SKIP TYPEOUTS
(1) 025764 001004 JSR PC,MYTYPE ;;GO TO USER ERROR ROUTINE
(1) 025766 004737 026306 TYPE ,$CRLF
(1) 025772 104401 001165 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 025776 122737 000001 001210 BNE 2$ ;;NO,SKIP APT ERROR REPORT
(1) 026004 001007 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 026006 113737 001114 026020 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
(1) 026014 004737 025340 21$: .BYTE 0
(1) 026020 000 .BYTE 0
(1) 026021 000
  
```

```
(1) 026022 000777          22$: BR      22$      ;;APT ERROR LOOP
(1) 026024 005777 153110  2$:  TST    @SWR    ;;HALT ON ERROR
(1) 026030 100001          BPL     3$      ;;SKIP IF CONTINUE
(1) 026032 000000          HALT                    ;;HALT ON ERROR!
(1) 026034
(3) 026034 104401 024102   3$:  TYPE    ,ATTOFF   ;ALL ATTRIBUTES OFF IN CASE TYPED IN BLINKING BOLD
(3) 026040 104401          RESTVT                ;RESTORE VT100 TO DISPLAY STATE
(3) 026042 017746 153102   MOV     @STPS,-(SP)   ;SAVE STATE OF XMIT IE
(3) 026046 042777 000100 153074 BIC     #IE,@STPS   ;AND THEN CLEAR IT
(3) 026054 105237 001250   INCB   TYPFLG      ;KEEP TYPE FROM SAVING CURSORS
(3) 026060 104401 016236   TYPE    ,SAVCON    ;SAVE THE CURSOR POSITION ONLY
(3) 026064 104401 016212   TYPE    ,ERRPOS    ;POSITION CURSOR OVER ERROR COUNT
(4) 026070 013746 001112   MOV     $ERTTL,-(SP) ;SAVE $ERTTL FOR TYPEOUT
(4) 026074 104405          TYPDS                ;;GO TYPE--DECIMAL ASCII WITH SIGN
(3) 026076 104401 016227   TYPE    ,SERPOS    ;POSITION CURSOR OVER SOFT ERROR COUNT
(4) 026102 013746 001256   MOV     $ERTTL,-(SP) ;SAVE $ERTTL FOR TYPEOUT
(4) 026106 104405          TYPDS                ;;GO TYPE--DECIMAL ASCII WITH SIGN
(3) 026110 104401 011577   TYPE    ,RESCUR    ;RESTORE THE CURSOR POSITION
(3) 026114 012677 153030   MOV     (SP)+,@STPS ;RESTORE STATE OF XMIT IE
(3) 026120 105337 001250   DECB   TYPFLG      ;RESTORE STATE OF TYPFLG
(3) 026124 105737 001252   TSTB   GFLAG      ;SEE IF ^G TYPED DURING TYPEOUTS
(3) 026130 001407          BEQ     101$       ;BR IF NOT (NOTE THAT GFLCHK SUB. CANNOT BE USED)
(3) 026132 104407          SAVEVT                ;SAVE THE POSITION
(3) 026134 104401 025257   TYPE    ,SCNTLG
(3) 026140 104406          GTSWR
(3) 026142 105037 001252   CLRB   GFLAG      ;CLEAR THE CONTROL G TYPED FLAG
(3) 026146 104410          RESTVT
(3) 026150 000002          101$: RTI
```

9843

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

(1)
(2)
(1)
(1)
(1)
(1)
(1) 026152
(1) 026152 104401 001165
(1) 026156 010046
(1) 026160 005000
(1) 026162 153700 001114
(1) 026166 001004
(1)
(2) 026170 013746 001116
(2)
(2) 026174 104402
(1) 026176 000426
(1) 026200 005300
(1) 026202 006300
(1) 026204 006300
(1) 026206 006300
(1) 026210 062700 001276
(1) 026214 012037 026224
(1) 026220 001404
(1) 026222 104401
(1) 026224 000000
(1) 026226 104401 001165
(1) 026232 012037 026242
(1) 026236 001404
(1) 026240 104401
(1) 026242 000000
(1) 026244 104401 001165
(1) 026250 011000
(1) 026252 001004
(1) 026254 012600
(1) 026256 104401 001165
(1) 026262 000207
(1) 026264
(2) 026264 013046
(2) 026266 104402
(1) 026270 005710
(1) 026272 001770
(1) 026274 104401 026302
(1) 026300 000771
(1) 026302 020040 000
(1) 026306

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

\$ERRTYP:

TYPE , \$CRLF ; "CARRIAGE RETURN" & "LINE FEED"
MOV RO, -(SP) ; SAVE RO
CLR RO ; PICKUP THE ITEM INDEX
BISB @#\$ITEMB, RO
BNE 1\$; IF ITEM NUMBER IS ZERO, JUST
; TYPE THE PC OF THE ERROR
MOV \$ERRPC, -(SP) ; SAVE \$ERRPC FOR TYPEOUT
; ERROR ADDRESS
TYPCC ; GO TYPE--OCTAL ASCII(ALL DIGITS)
BR 6\$; GET OUT
1\$: DEC RO ; ADJUST THE INDEX SO THAT IT WILL
ASL RO ; WORK FOR THE ERROR TABLE
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"
2\$: .WORD 0 ; "ERROR MESSAGE" POINTER GOES HERE
TYPE , \$CRLF ; "CARRIAGE RETURN" & "LINE FEED"
3\$: MOV (RO)+, 4\$; PICKUP "DATA HEADER" POINTER
BEQ 5\$; SKIP TYPEOUT IF 0
TYPE ; TYPE THE "DATA HEADER"
4\$: .WORD 0 ; "DATA HEADER" POINTER GOES HERE
TYPE , \$CRLF ; "CARRIAGE RETURN" & "LINE FEED"
5\$: MOV (RO), RO ; PICKUP "DATA TABLE" POINTER
BNE 7\$; GO TYPE THE DATA
6\$: MOV (SP)+, RO ; RESTORE RO
TYPE , \$CRLF ; "CARRIAGE RETURN" & "LINE FEED"
RTS PC ; RETURN
7\$: MOV @ (RO)+, -(SP) ; SAVE @ (RO)+ FOR TYPEOUT
TYPCC ; 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\$: .ASCIIZ / / ; TWO(2) SPACES
.EVEN

9844

9845

9846

9847 026306 113737 001114 026330 MYTYPE: MOVB \$ITEMB, ERRNUM ; FOR ERROR TYPEOUT

9848 026314 013737 001116 001174 MOV \$ERRPC, \$TESTN ; FOR APT

9849 026322 004737 026152 JSR PC, \$ERRTYP ; TYPE ERR MSG

9850 026326 000207 RTS PC

9851

9852 026330 000000 ERRNUM: 0 ; FOR ERR TYPEOUT

9853

9854 026332 000000
9855 026334 000000

PERR: 0
PSERR: 0

;PASS ERR COUNT
;PASS SOFT ERR COUNT... ALL FOR EOP TYPEOUT


```

9859      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(2)      ;*****
(1)      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)      ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)      ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)      ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)      ;*REPLACED WITH SPACES.
(1)      ;*CALL:
(1)      ;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
(1)      ;*      TYPDS      ;;GO TO THE ROUTINE
(1)
(1)      $TYPDS:
(3)      026440      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
(3)      026440      010046      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
(3)      026442      010146      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
(3)      026444      010246      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
(3)      026446      010346      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
(3)      026450      010546      MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
(1)      026452      012746      020200      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
(1)      026456      016605      000020      BPL      1$      ;;BR IF INPUT IS POS.
(1)      026462      100004      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
(1)      026464      005405      MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
(1)      026466      112766      000055      000001      1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
(1)      026474      005000      MOV      #DBLK,R3      ;;SETUP THE OUTPUT POINTER
(1)      026476      012703      026654      MOVB    #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
(1)      026502      112723      000040      2$:      CLR      R2      ;;CLEAR THE BCD NUMBER
(1)      026506      005002      MOV      $DTBL(R0),R1      ;;GET THE CONSTANT
(1)      026510      016001      026644      3$:      SUB      R1,R5      ;;FORM THIS BCD DIGIT
(1)      026514      160105      BLT      4$      ;;BR IF DONE
(1)      026516      002402      INC      R2      ;;INCREASE THE BCD DIGIT BY 1
(1)      026520      005202      BR      3$
(1)      026522      000774      4$:      ADD      R1,R5      ;;ADD BACK THE CONSTANT
(1)      026524      060105      TST      R2      ;;CHECK IF BCD DIGIT=0
(1)      026526      005702      BNE      5$      ;;FALL THROUGH IF 0
(1)      026530      001002      TSTB     (SP)      ;;STILL DOING LEADING 0'S?
(1)      026532      105716      BMI      7$      ;;BR IF YES
(1)      026534      100407      5$:      ASLB     (SP)      ;;MSD?
(1)      026536      106316      BCC      6$      ;;BR IF NO
(1)      026540      103003      MOVB     1(SP),-1(R3)      ;;YES--SET THE SIGN
(1)      026542      116663      000001      177777      6$:      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1)      026550      052702      000060      7$:      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)      026554      052702      000040      MOVB    R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)      026560      110223      TST      (R0)+      ;;JUST INCREMENTING
(1)      026562      005720      CMP      R0,#10      ;;CHECK THE TABLE INDEX
(1)      026564      020027      000010      BLT      2$      ;;GO DO THE NEXT DIGIT
(1)      026570      002746      BGT      8$      ;;GO TO EXIT
(1)      026572      003002      MOV      R5,R2      ;;GET THE LSD
(1)      026574      010502      BR      6$      ;;GO CHANGE TO ASCII
(1)      026576      000764      8$:      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
(1)      026600      105726      BPL      9$      ;;BR IF NO
(1)      026602      100003      MOVB    -1(SP),-2(R3)      ;;YES--SET THE SIGN FOR TYPING
(1)      026604      116663      177777      177776      9$:      CLRB     (R3)      ;;SET THE TERMINATOR
(3)      026612      105013      MOV      (SP)+,R5      ;;POP STACK INTO R5
(3)      026614      012605      MOV      (SP)+,R3      ;;POP STACK INTO R3
(3)      026616      012603      MOV      (SP)+,R2      ;;POP STACK INTO R2
(3)      026620      012602
    
```

```
(3) 026622 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
(3) 026624 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
(1) 026626 104401 026654  TYPE      $DBLK      ;;NOW TYPE THE NUMBER
(1) 026632 016666 000002 000004  MOV      2(SP),4(SP)   ;;ADJUST THE STACK
(1) 026640 012616      MOV      (SP)+,(SP)
(1) 026642 000002      RTI                          ;;RETURN TO USER
(1) 026644 023420      $DTBL: 1000.
(1) 026646 001750      1000.
(1) 026650 000144      100.
(1) 026652 000012      10.
(1) 026654 000004      $DBLK: .BLKW 4
```


(1)	027026	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
(1)	027030	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
(1)	027034	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
(1)	027040	110337	027104		MOVB	R3,8\$::SAVE FOR TYPING
(1)	027044	104401	027104		TYPE	,8\$::GO TYPE THIS DIGIT
(1)	027050	105337	027106	7\$:	DECB	\$OCNT	::COUNT BY 1
(1)	027054	003347			BGT	2\$::BR IF MORE TO DO
(1)	027056	002402			BLT	6\$::BR IF DONE
(1)	027060	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
(1)	027062	000744			BR	2\$::GO DO THE LAST DIGIT
(1)	027064	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
(1)	027066	012604			MOV	(SP)+,R4	::RESTORE R4
(1)	027070	012603			MOV	(SP)+,R3	::RESTORE R3
(1)	027072	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
(1)	027100	012616			MOV	(SP)+,(SP)	
(1)	027102	000002			RTI		::RETURN
(1)	027104	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
(1)	027105	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
(1)	027106	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
(1)	027107	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
(1)	027110	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

9863

.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.

(1) 027112 010046
 (1) 027114 016600 000002
 (1) 027120 005740
 (1) 027122 111000
 (1) 027124 006300
 (1) 027126 016000 027146
 (1) 027132 000200

\$TRAP: MOV R0,-(SP) ;;SAVE R0
 MOV 2(SP),R0 ;;GET TRAP ADDRESS
 TST -(R0) ;;BACKUP BY 2
 MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
 ASL R0 ;;POSITION FOR INDEXING
 MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE
 RTS R0 ;;GO TO ROUTINE

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

(1) 027134 011646
 (1) 027136 016666 000004 000002
 (1) 027144 000002

\$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

(3) 027146 027134
 (3) 027150 024242
 (3) 027152 026710
 (3) 027154 026664
 (3) 027156 026724
 (3) 027160 026440

\$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)
 \$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

(1) 027162 024754

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

9864 027164 013026
 9865 027166 013106

T.SAVT ;;CALL=SAVEVT TRAP+7(104407) SAVE VT100 CURSOR AND ATTRIBUTES
 T.RSVT ;;CALL=RESTVT TRAP+10(104410) RESTORE VT100 CURSOR AND ATTRIBUTES

9866

9867

9868

9869

WRITE DATA PACKET BUFFER

9870 027170 000000
 9876 027372 000000

WTPK1: .WORD 0 ;WRITE DATA PACKET 1
 CKPK1: .WORD 0 ;WRITE DATA PACKET 1 CHECKSUM WORD

9878 027374 000000
 9884 027576 000000

WTPK2: .WORD 0 ;WRITE DATA PACKET 2
 CKPK2: .WORD 0 ;WRITE DATA PACKET 2 CHECKSUM WORD

9886 027600 000000
 9892 030002 000000

WTPK3: .WORD 0 ;WRITE DATA PACKET 3
 CKPK3: .WORD 0 ;WRITE DATA PACKET 3 CHECKSUM WORD

9894 030004 000000
 9900 030206 000000

WTPK4: .WORD 0 ;WRITE DATA PACKET 4
 CKPK4: .WORD 0 ;WRITE DATA PACKET 4 CHECKSUM WORD

9901
9902
9903
9904
9905
9906
9912
9913
9914
9915

030210 000000
031664 000000

031666
000001

* TUS8 READ BUFFER (TOTAL RESERVED LOCATIONS = 407 WORDS) *

TURDBF: .WORD 0
TURDLA: .WORD 0

LSTAD:
.END

OPAR = 000020	6768#	7495	7498											
OPCHTX 016430	7277*	9379#												
OPCODE 007030	7716*	7838#	7997*	8003*	8008*	8152	8211	8381	8386	9465	9466	9467		
OPTCHR 014012	7248	7250	7264	7266	7270	7272*	7277	8729*	8968*	8982#	9463			
OPTPRE 016371	7278	9378#												
ORERR = 040000	6804#	6839#												
PARAM = 177420	6692#	7108*	7332*	7495*	7498*	7586*	7626*	7666*	8587*	9217*	9841*			
PARERR= 010000	6806#	6840#												
PASPOS 016175	8486	9358#												
PAT 003240	7155*	7159	7165	7176*	7182*	7185	7191	7203*	7229#	9457				
PERR 026332	7120*	8497	9841*	9854#										
PIRQ = 177772	6654#													
PIRQVE= 000240	6654#													
PKTERR 007062	7758*	7786*	7858#	8268*	8391*	8404	9466	9467						
PKTPNT 007060	7854#	8324*	8347											
PORT 016605	7360	9394#												
PRB = 177516	6697#	9080*												
PRCHR 014512	7338*	9080	9081	9083	9085	9087*	9090*	9092*	9094*	9103#				
PRINT 016561	7357	8608	8614	9391#										
PRPORT 012616	7349	7358	8589*	8605*	8683#									
PRPRES 012614	7327	8588*	8592*	8596	8601	8606*	8682#							
PRS = 177514	6696#	7340*	7351*	8590	8603	9076	9099*							
PRSRV 014350	7333	9076#												
PRT = 020000	6740#	7332												
PRTST 003702	7299	7303	7313	7317	7327#									
PRVEC = 000200	6668#	7333*	7334*											
PRO = 000000	6654#	7099												
PR1 = 000040	6654#													
PR2 = 000100	6654#													
PR3 = 000140	6654#													
PR4 = 000200	6654#													
PR5 = 000240	6654#													
PR6 = 000300	6654#													
PR7 = 000340	6654#	7035	7040	7136	7281	8482	8523	8706	8708	8780				
PS = 177776	6654#													
PSERR 026334	7121*	8423*	8499	9855#										
PSW = 177776	6654#													
PWRVEC= 000024	6654#													
RACT = 004000	6824#													
RCVDUN 014014	8731*	8746	8751	8972*	8977*	8986#								
RCVPTR 014016	8728*	8966	8970	8974*	8975	8987#								
RDONE = 000200	6878#	9321												
RDY = 000200	6810#	6865#	6883#	8776	8792	9021	9284	9669	9671					
REGHLD 013372	8852*	8858	8872#											
RESCUR 011577	8489	8518#	9841											
RESDSP 016264	8791	9365#	9642											
RESTVT= 104410	7292	7321	7363	7467	7562	7612	7652	7692	7747	7815	7830	8938	9841	
	9865#													
RESVEC= 000010	6654#													
RING = 040000	6789#	6821#	7402	7475	7513									
RTS = 000004	6798#	6833#	7106	7107	7385	7388	7390	7425	7492	7493	7499	7501	7536	
	7574	8584	8585	8595	9221	9222	9246	9841						
RUN 017045	7291	7320	7362	7561	7611	7651	7691	7746	9408#					
SAVCON 016236	8483	9362#	9841											
SAVCUR 016241	8782	9363#	9621											
SAVEVT= 104407	7289	7318	7356	7464	7559	7608	7648	7688	7744	7811	7826	8479	8934	

.SAPTH	5261#	6624#	6924
.SAPTY	5436#	6624#	9790
.SASTA	5307#		
.SCATC	905#	6622#	6907
.SCMTA	1016#	6622#	6942
.SDB2D	4591#		
.SDB2O	4714#		
.SDIV	4494#		
.SEOP	2162#	6622#	
.SERRO	2643#	6623#	9841
.SERRT	2838#	6623#	9843
.SMULT	4431#		
.SPOWE	4143#		
.SRAND	4218#	6624#	9857
.SRDDE	3814#		
.SRDOC	3723#		
.SREAD	3328#	6622#	
.SR2AZ	4858#		
.SSAVE	3889#		
.SSB2D	4675#		
.SSB2O	4776#		
.SSCOP	2397#	6623#	
.SSIZE	4271#		
.SSUPR	4814#		
.STRAP	3991#	6623#	9863
.STYPB	3221#		
.STYPD	3144#	6623#	9859
.STYPE	2925#	6622#	
.STYPO	3048#	6624#	9861
.S4OCA	944#		
.1170	498#		

. ABS. 031666 000

ERRORS DETECTED: 0

CVKDBA, CVKDBA/CRF=SYSMAC.SML, CVKDBA.P11
RUN-TIME: 53 62 4 SECONDS
RUN-TIME RATIO: 257/121=2.1
CORE USED: 34K (67 PAGES)