PDP-15/PDP-11
UNICHANNEL

IDENTIFICATION

Product Code:	MAINDEC-15-DAUCD-A-D
Product Name:	UC15 DEC/X11 Exerciser User Manual
Date:	October 16, 1973
Maintainer:	Diagnostic Group
Authors:	R. Koller, R. Christopher

1. ABSTRACT

This manual is a compilation of all the DEC/X11 documentation which is necessary for the MAINDEC-15-DAUCD-A paper tape exerciser module for the UC15. The manual contains the following:

Name	Maindec #
A. DEC/X11 User Manual	11-DXQAA-B-D
B. DEC/X11-UC15 Monitor	11-DXQAC-B-LA
C. XUCAB-DEC/X11 UC15 Module	11-DXUCA-B-D
D. XRKAA-DEC/X11 RK11 Module	11-DXRKA-A-D
E. XLPAB-DEC/X11 LP11 Module	11-DXLPA-B-D
F. XCRAB-DEC/X11 CR11 Module	11-DXCRA-B-D
G. XDPAB-DEC/X11 DP11 Module	11-DXDPA-B-D
H. XXYAB-DEC/X11 XY11 Module	11-DXXYA-B-D

.RFM !

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DXQAA-B-D

PRODUCT NAME: DEC/X11 USER MANUAL

DATE: JUNE 15, 1973

MAINTAINER: DIAGNOSTIC GROUP

AUTHOR(S): R. KOLLER

THIS DOCUMENT IS WRITTEN IN TWO SECTIONS:

SECTION 1 - DEC/X11 EXERCISER DOCUMENT

SECTION 2 - UC15 MONITOR (UNIMON) DIFFERENCES

!
.RFM !

****
* SECTION 1 *
* *
* DEC/X11 EXERCISER DOCUMENT *
****

TABLE OF CONTENTS

1. ABSTRACT

DEC/X11 IS A SYSTEM EXERCISER FOR THE PDP-11 FAMILY. IT IS DESIGNED TO PROMOTE SYSTEM INTERACTION, AND TO DETECT SYSTEM FAILURES, IF ANY, CAUSED BY SAID INTERACTION. ADDITIONALLY, DEC/X11 IS DESIGNED TO BE USED AS AN OVERALL SYSTEM CONFIDENCE TEST, AND TO PROVIDE AN INDICATION OF THE INTEGRITY OF INDIVIDUAL SYSTEM COMPONENTS. TO THAT EFFECT, THE INDIVIDUAL SYSTEM COMPONENT TEST MODULES CAN BE WRITTEN TO BE EITHER SIMPLE OR EXTENSIVE, DEPENDING ON THE THOROUGHNESS REQUIRED.

THE BASIC COMPONENTS OF THE DEC/X11 PACKAGE ARE:

A. DEC/X11 MONITORS (STANDARD MONITOR, UC15 MONITOR).
B. DEC/X11 OPTION/DEVICE TEST MODULES.
C. DEC/X11 CONFIGURATOR/LINKER PROGRAM.

THE MONITOR, TEST MODULES, AND THE CONFIGURATOR/LINKER PROGRAM ARE USED TO GENERATE AN "EXERCISER MODULE" THAT IS LOADABLE BY THE STANDARD ABS LOADER. IN THE EXERCISER MODULE ARE INCLUDED THE MONITOR AND ONLY THOSE TEST MODULES REQUIRED BY THE SYSTEM TO BE TESTED. THE CONFIGURATOR/LINKER PROGRAM IS USED TO GENERATE THE DESIRED EXERCISER MODULE. THIS DOCUMENT DOES NOT CONCERN ITSELF WITH THE CONFIGURATION-LINKING PROCESS. IT ASSUMES THAT AN EXERCISER MODULE HAS BEEN CREATED, AND PROVIDES INSTRUCTIONS FOR ITS USE. DESCRIPTION AND INSTRUCTIONS FOR THE CONFIGURATOR/LINKER ARE FOUND IN THE CONFIGURATOR/LINKER DOCUMENTATION. (MAINDEC-11-DXQBA CONFIGURATION AND PROGRAMMING MANUAL).

2. REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

2.1.1 PAPER TAPE CONFIGURATION

TO CONFIGURE FROM AND TO PAPER TAPE THE FOLLOWING HARDWARE IS NEEDED:

A. PDP-11 PROCESSOR
B. CONSOLE TELETYPE OR EQUIVALENT.
C. PAPER TAPE INPUT AND OUTPUT (PC11, OR ASR33/35 TELETYPE).
D. 8K MINIMUM STORAGE.

2.1.2 DECTAPE CONFIGURATION

TO CONFIGURE TO OR FROM DECTAPE THE FOLLOWING HARDWARE IS NEEDED INSTEAD OF PAPER TAPE:

A. TC11 DECTAPE CONTROL
B. TU56 DUAL DECTAPE TRANSPORT.

2.1.3 RK11 DISK CARTRIDGE CONFIGURATION

TO CONFIGURE FROM DECPACK THE FOLLOWING HARDWARE IS REQUIRED INSTEAD OF PAPER TAPE:

A. RK11 DISK CONTROL
B. RK05 HIGH DENSITY DISK DRIVE AND CARTRIDGE.

2.2 SOFTWARE REQUIREMENTS

2.2.1 PAPER TAPE CONFIGURATION

NO SPECIAL SOFTWARE REQUIREMENTS.

2.2.2 DECTAPE CONFIGURATION

IN ORDER TO CONFIGURE AND LOAD FROM DECTAPE, DDP2-DECTAPE DIAGNOSTIC PACKAGE SOFTWARE IS REQUIRED. (MAINDEC-11-DZQDD).

2.2.3 RK11 DISK CARTRIDGE CONFIGURATION

TO CONFIGURE AND LOAD FROM RK11 DISK CARTRIDGE, RKDP - RK11 DIAGNOSTIC PACKAGE SOFTWARE IS REQUIRED. (MAINDEC-11-DZQDE).

3. LOADING PROCEDURE

3.1 LOADING FROM PAPER TAPE

LOAD THE EXERCISER MODULE BY MEANS OF THE ABSOLUTE LOADER.

3.2 LOADING FROM DECTAPE

THE EXERCISER MODULE IS LOADED BY TYPING THE FILE NAME WHILE UNDER CONTROL OF THE DDP2 MONITOR. THE EXERCISER MODULE MUST BE A NAMED FILE ON THE DECTAPE AND MUST HAVE AN EXTENSION OF .BIN OR .BIC .
EXAMPLE: DECX1<CR> , OR
DECX1<ALTMODE> TO SELF START AT 000200.

3.3 LOADING FROM RK11 DISK CARTRIDGE

THE EXERCISER MODULE IS LOADED BY TYPING THE FILE NAME WHILE UNDER CONTROL OF THE RKDP MONITOR. THE EXERCISER MODULE MUST BE A NAMED FILE ON THE THE CARTRIDGE, AND MUST HAVE AND EXTENSION OF .BIN OR.BIC .
EXAMPLE: DECX2<CR> ,OR
DECX2<ALTMODE> TO SELF START AT 000200.

********** WARNING **********
USERS OF DDP2/TCDP OR RKDP PACKAGES MAY BE AWARE THAT THE "UPDATE" PROGRAM CAN BE USED TO LOAD PROGRAMS. BE AWARE THAT THE "UPDATE" PROGRAM DOES NOT INDICATE THE "LOAD MEDIUM" TO THE LOADED PROGRAM, AS DOES THE DDP2/TCDP OR RKDP MONITOR(S), AND THEREFORE, IT IS POSSIBLE TO WIPE OUT THE LOAD MEDIUM DEVICE. IT IS ALWAYS BEST TO LOAD PROGRAMS BY MEANS OF THE APPROPRIATE DIAGNOSTIC PACKAGE MONITOR, AND NOT THE "UPDATE" PROGRAM.

4. STARTING PROCEDURE

TO START: LOAD ADDR 000200 AND PRESS START.

TO RESTART: LOAD ADDR 001000 AND PRESS START.

IN EITHER A START OR RESTART, THE MONITOR TYPES ONE OF THE FOLLOWING MESSAGES:

```
DEC/X11 EXERCISER
.
```

OR

```
DEC/X11 EXERCISER
WRITE BUFFER ROTATION ENABLED, RANGE: XXXXXX  YYYYYY
.
```

THE SECOND PRINTOUT OCCURS ONLY IF THERE IS SUFFICIENT FREE CORE TO PERMIT ROTATION OF THE WRITE BUFFER. TO ROTATE, THERE MUST BE AT LEAST 1024 WORDS OF FREE CORE AVAILABLE.

THE DOT (.) INDICATES THAT THE MONITOR IS READY TO ACCEPT OPERATOR COMMANDS THROUGH THE KEYBOARD.

5. OPERATING PROCEDURE

THE DEC/X11 EXERCISER IS CONTROLLED BY MEANS OF KEYBOARD COMMANDS, AND THE SWITCH REGISTER (SR).

5.1 SWITCH REGISTER OPTIONS

SWITCH REGISTER OPTIONS APPLY ONLY DURING EXECUTION OF THE EXERCISER. THE OPTIONS ARE:

SR15 = 1	"HALT" MODULE AFTER ERROR. THE FAILING MODULE IS PREVENTED FROM FURTHER EXECUTION. NORMALLY, A "MODULE DROPPED" PRINTOUT PRECEDES HALTING OF THE MODULE.
SR14 = 1	INHIBIT MODULE HALT AFTER 20 ERRORS. SR14 SET TO A 1 PREVENTS THE MONITOR FROM HALTING THE FAILING MODULE AFTER 20 ERRORS. IF SET TO A 0, SR14 WILL ENABLE THE MONITOR TO HALT THE MODULE AFTER THE 20TH ERROR, AFTER A "MODULE DROPPED" MESSAGE.
SR13 = 1	INHIBIT ERROR PRINTOUTS.
SR12 = 1	INHIBIT "END OF PASS" PRINTOUTS.
SR11 = 1	LOCK-ON-ERROR SEQUENCE (WHENEVER IMPLEMENTED BY MODULES).

SETTING THE SR TO 074000 INHIBITS ALL PRINTOUTS, AND PREVENTS MODULE HALTS, IMPROVING THE CHANCES OF PERFORMING SCOPING OPERATIONS.

5.2 KEYBOARD COMMANDS

THE FOLLOWING CHARACTERS ARE CONSIDERED VALID BY THE MONITOR:

A THROUGH Z,
1 THROUGH 9,
SPACE, CARRIAGE RETURN <CR>, LINE FEED <LF>, RUBOUT <177>,
AND CONTROL C (^C).

ANY OTHER CHARACTERS TYPED ARE IGNORED.

A COMMAND IS ENDED BY TYPING <CR>. PRIOR TO TYPING <CR> CORRECTIONS MAY BE MADE BY USE OF THE RUBOUT KEY TO REMOVE A PREVIOUSLY TYPED CHARACTER, OR BY ^C TO KILL ALL INPUT AND START AGAIN.

IF TOO MANY CHARACTERS ARE TYPED, A "KBUF OFLO" MESSAGE IS TYPED. COMMAND MUST BE GIVEN OVER AGAIN.

AN "INVALID COMMAND" MESSAGE OCCURS IF THE COMMAND GIVEN IS NOT RECOGNIZED BY THE MONITOR, OR IF A "RUN" COMMAND IS GIVEN AND NO MODULES ARE SELECTED TO RUN.

AN "INVALID/NEX NAME" MESSAGE OCCURS IF USER SPECIFIES A NON-EXISTENT MODULE, OR INVALID CHARACTERS ARE USED.

AN "INVALID ADDR/DATA" MESSAGE OCCURS IF USER SPECIFIES AN ODD ADDRESS, OR IF NON-OCTAL NUMBERS ARE SPECIFIED.

WHENEVER A MODULE NAME MUST BE SPECIFIED AS PART OF A KEYBOARD COMMAND, REMEMBER THAT A NAME ALWAYS CONSISTS OF SIX (6) CHARACTERS.. EXAMPLE: XTCAAA

5.2.1 THE "MAP" COMMAND

TYPING MAP AND <CR> RESULTS IN THE MONITOR TYPING LIST OF RESIDENT MODULES WITH THEIR PC AND STATUS. EXAMPLE:

```
.MAP<CR>

XPAAAA  AT 012544  STAT 140000
XPABAA  AT 011470  STAT 160000
XKLAAA  AT 013706  STAT 140000
XDNAAA  AT 015402  STAT 040020
XCPAAA  AT 016176  STAT 040020
.
```

WHERE:

XPAAAA IS THE MODULE NAME. IN THE MODULE NAME TYPED THE FIRST FOUR CHARACTERS IDENTIFY THE TEST MODULE, THE FIFTH CHARACTER IS THE VERSION LETTER FOR THE MODULE, AND THE SIXTH CHARACTER INDICATES WHICH COPY OF THE MODULE IS BEING DESCRIBED. IF THERE WERE THREE COPIES OF THE MODULE AS PART OF THE EXERCISER MODULE, THE FIRST COPY'S SIXTH LETTER WOULD BE AN A, THE SECOND COPY'S A B, AND THE THIRD COPY'S A C.

AT 012544 DENOTES THE ADDRESS OF THE MODULE'S FIRST WORD. (STARTING ADDRESS OF MODULE'S CODE, NOT THE STARTING ADDRESS).

STAT 140000, MODULE'S STATUS. WHERE:

BIT15 = 1	MODULE IS AN I/O MODULE (IOMOD).
BIT15 = 0	MODULE IS A BACKGROUND MODULE (BKMOD).
BIT14 = 1	MODULE IS SELECTED FOR EXECUTION.
BIT14 = 0	MODULE IS DESELECTED.
BIT13 = 1	IN PREVIOUS RUN, MODULE WAS HALTED.
BIT13 = 0	IN PREVIOUS RUN, MODULE WAS NOT HALTED.

THE RIGHT HALF OF THE STATUS WORD INDICATES THE PROCESSOR STATUS ASSUMED WHEN RUNNING THE MODULE. (0 FOR IOMODS AND NBKMODS, 20 FOR BKMODS).

5.2.2 THE SEL(ECT) COMMAND

THE SEL(ECT) COMMAND IS USED TO ENABLE TO RUN ONE MODULE, OR ALL MODULES. EXAMPLES:

```
.SEL<CR>
.
```

SELECTS ALL MODULES FOR EXECUTION.

```
.SEL XDCAAA<CR>          SELECTS MODULE XDCAAA ONLY.
.
```

NOTE: WHEN EXERCISER IS FIRST LOADED, ALL MODULES ARE SELECTED.

5.2.3 THE DES(ELECT) COMMAND

THE DES(ELECT) COMMAND IS USED TO DISABLE ONE MODULE FROM RUNNING, OR ALL MODULES FROM RUNNING. EXAMPLES:

```
.DES<CR>                 DESELECTS ALL MODULES.
.
.DES XDCAAA<CR>          DESELECTS MODULE XDCAAA ONLY.
```

5.2.4 COMBINED USE OF SEL AND DES COMMANDS

TO SELECT ONE MODULE ONLY:

```
.DES<CR>                 DESELECT ALL MODULES.
.SEL XDCAAA<CR>          SELECT MODULE XDCAAA.
```

TO SELECT ALL MODULES BUT ONE:

```
.SEL<CR>                 ;SELECT ALL MODULES.
.DES XDCAAA              ;DESELECT UNWANTED MODULE XDCAAA.
```


5.2.5 THE MOD(IFY) COMMAND

THE MOD(IFY) COMMAND IS USED TO EXAMINE AND/OR MODIFY THE CONTENTS OF STORAGE. ALL ADDRESSES SPECIFIED MUST BE EVEN.

EXAMPLES:

```
.MOD 4000<CR>                    OPEN CONTENTS OF LOC 4000
```

MONITOR TYPES:

```
004000  123456                   LOC 4000 CONTAINS 123456
```

THE OPERATOR CAN:

A. CLOSE LOC 4000 BY TYPING <CR>, OR
B. TYPE A NEW VALUE AND CLOSE WITH <CR>, OR
C. TYPE A NEW VALUE AND OPEN NEXT WORD WITH <LF>, OR
D. CLOSE LOC 4000 AND OPEN NEXT WORD WITH <LF>.

```
.MOD XDCAAA 20<CR>               OPENS 20TH OCTAL WORD OF MODULE XDCAAA.
```

MONITOR TYPES:

```
012020  140000                   20TH OCTAL WORD OF MODULE XDCAAA CONTAINS
                                 140000. THE ACTUAL ADDR IS 012020.
```

SAME OPERATOR OPTIONS AS IN PREVIOUS EXAMPLE APPLY.

AS CAN BE SEEN, THE MOD COMMAND MAKES IT POSSIBLE TO OPEN AND MODIFY NOT ONLY ABSOLUTE ADDRESSES, BUT RELATIVE ADDRESSES, (RELATIVE TO START ADDRESS OF SPECIFIED MODULE). NOTE ALSO THAT WHEN A RELATIVE ADDRESS IS SPECIFIED, THE MONITOR RESPONDS BY TYPING THE ABSOLUTE ADDRESS OF RELATIVE ADDRESS SPECIFIED.

5.2.6 THE "RUN" COMMAND

THE RUN COMMAND STARTS THE EXERCISER RUNNING, ONCE IN "RUN MODE", THE MONITOR STARTS ONLY THOSE MODULES THAT HAVE BEEN SELECTED, (BIT14 OF MODULE'S STAT WORD IS SET), NRKMODS ARE RUN FIRST, ONE AT A TIME, I/O MODULES (INTERRUPT DRIVEN) ARE STARTED NEXT, AND THEN BACKGROUND MODULES ARE RUN ONE AT A TIME.

TO START EXECUTION, TYPE:

RUN<CR>

5.2.7 ENDING EXERCISER "RUN"

NORMALLY, ONCE STARTED, THE EXERCISER RUNS INDEFINITELY, UNLESS:

A. THE OPERATOR TYPES CTRL C (^C). THE MONITOR THEN STOPS ALL MODULES, AND TYPES A "RUN SUMMARY" THAT INDICATES THE MODULES THAT RAN, THE NUMBER OF PASSES MADE BY EACH MODULE, AND THE NUMBER OF ERRORS DETECTED BY EACH MODULE.

B. IF AFTER A PERIOD OF TIME, DUE TO MODULE DETECTED ERRORS ALL MODULES ARE DROPPED, THE MONITOR TYPES A "RUN SUMMARY", AND GOES BACK TO COMMAND MODE.

C. SYSTEM ERROR OCCURS. A SYSTEM ERROR IS DEFINED AS A BUS ERROR TRAP, (TRAP TO LOC 4), OR A RESERVED INSTRUCTION TRAP (TRAP TO LOC 10). THE MONITOR TYPES A "SYS ERROR" MESSAGE, A "RUN SUMMARY" AND THEN RETURNS TO COMMAND MODE.

5.2.8 THE "FILL" COMMAND

THE MONITOR'S TYPEOUT ROUTINE NORMALLY OUTPUTS TWELVE (12) FILLER CHARACTERS AFTER A CARRIAGE RETURN, IN ORDER TO PREVENT GARBLED TYPEOUTS WHEN USING THE LA30 S AS THE CONSOLE DEVICE. THE "FILL" COMMAND PERMITS THE USER TO CHANGE THE "FILLER" COUNT AND THE "FILL" CHARACTER ITSELF. TO USE, TYPE:

```
FILL<CR>          ;REQUESTS CURRENT FILLER DATA TO BE OUTPUTTED.

000014  XXXXXX    ;DATA IS OUTPUTTED. THE LEFT HALF IS THE FILLER
                  ;CHARACTER ITSELF. THE RIGHT HALF IS THE FILLER
                  ;COUNT. THE XXXXXX INDICATES THE PLACE WHERE THE
                  ;USER TYPES THE NEW DATA REQUIRED. EXAMPLE: 000001
                  ;CHANGES THE FILLER COUNT TO A 1, LEAVES FILL
                  ;CHARACTER AS 0.
```


5.2.9 HANDLING OF POWER FAILURE

IF A POWER FAILURE OCCURS, DEC/X11 WILL:

A. IF IN COMMAND MODE, THE MONITOR TYPES "PWR FAILURE" AND UPON RESTART, RETURNS TO COMMAND MODE.

B. IF IN RUN MODE, THE MONITOR TYPES "PWR FAILURE", AND UPON RESTART, RESTARTS RUN MODE WITHOUT CLEARING PREVIOUS PASS COUNT OR ERROR COUNT INFORMATION. THEREFORE, POWER FAILURES ARE NOT A PROBLEM IN OVERNIGHT TESTING.

5.2.10 "CHAIN" OPERATION OF DEC/X11

DEC/X11 IS "CHAINABLE" UNDER DDP MONITORS (DDP2, RKDP, ETC.). DEC/X11 CHAIN OPERATION IS AS FOLLOWS:

A. UPON STARTING, THE MONITOR DETERMINES THAT CHAIN MODE IS ENABLED, AND IMMEDIATELY GOES INTO RUN MODE.
B. EACH MODULE IS ALLOWED TO EXECUTE ONLY ONE PASS.
C. WHEN ALL MODULES HAVE COMPLETED ONE PASS, THE MONITOR ENDS RUN MODE AND EXITS TO THE DDP MONITOR.
D. IF THE CHAIN MONITOR RETURNS TO DEC/X11 THE MONITOR REPEATS STEPS B AND C. NO "RUN SUMMARY" IS TYPED AT END OF EACH CHAIN PASS.

6. ERRORS

6.1 SYSTEM ERROR

A SYSTEM ERROR PRINTOUT OCCURS WHENEVER INTENTIONALLY, OR UNINTENTIONALLY, A BUS ERROR TRAP (TRAP TO LOC 4) OR RESERVED INSTRUCTION TRAP OCCURS. A SYSTEM ERROR PRINTOUT LOOKS AS FOLLOWS:

```
SYS ERROR  SSSSSS  0000XX  YYYYYY  ZZZZZZ
```

WHERE:

SSSSSS CONTENTS OF STACK POINTER (R6) AT TIME OF TRAP.

0000XX 4 IF BUS ERROR, 10 IF RESERVED INSTRUCTION TRAP.

YYYYYY PC AT TIME OF FAILURE.

ZZZZZZ PROCESSOR STATUS AT TIME OF FAILURE.

FOLLOWING A SYS ERROR, THE MONITOR TYPES A RUN SUMMARY IF RUN MODE WAS ACTIVE, AND THEN RETURNS TO COMMAND MODE. IF RUN MODE WAS NOT ACTIVE, THE MONITOR SIMPLY RETURNS TO COMMAND MODE. IF IN CHAIN MODE, THE MONITOR EXITS TO THE DDP MONITOR.

THE MONITOR INTENTIONALLY CAUSES A SYSTEM ERROR, WHEN DUE TO SOME UNFORESEEN REASON, IT FINDS THAT ONE OF ITS QUEUES HAS OVERFLOWED. REFERRING TO THE TYPED PC WILL INDICATE WHICH OF THE QUEUES OVERFLOWED. IT IS NOT AN EXPECTED ERROR.

6.2 ERROR PRINTOUTS

6.2.1 THE ERROR PRINTOUT

TEST MODULES INDICATE ERRORS OTHER THAN A DATA ERROR BY MEANS OF THE "ERROR" PRINTOUT. THE "ERROR" PRINTOUT IS INVOKED BY MEANS OF AN "ERROR" CALL. THE "ERROR" PRINTOUT LOOKS AS FOLLOWS:

```
XDCAAA  PC XXXXXX  APC YYYYYY  ERR# NNNNNN
ACSR AAAAAA  CSRC CCCCCC  STATC SSSSSS
```

WHERE:

XDCAAA	FAILING MODULE NAME.
PC XXXXXX	ACTUAL PC OF ERROR CALL.
APC YYYYYY	ASSEMBLED PC OF ERROR CALL.
ERR# NNNNNN	ERROR COUNT IN CURRENT RUN (DECIMAL).
ACSR AAAAAA	CSR ADDR OF FAILING DEVICE. 0 IF NOT APPLICABLE.
CSRC CCCCCC	CONTENTS OF FAILING DEVICE CSR. 0 IF NOT APPLICABLE.
STATC SSSSSS	CONTENTS OF FAILING DEVICE STATUS REG, IF APPLICABLE.

USING THE VALUE TYPED IN APC YYYYYY THE USER SHOULD HAVE NO TROUBLE LOCATING IN THE LISTING THE ERROR CALL CAUSING THE PRINTOUT. ERROR CALLS ARE PRECEDED AND FOLLOWED BY A LINE OF ASTERISKS (*), TO MAKE THEM STAND OUT FROM THE LISTING. THE ERROR CALL ITSELF LOOKS AS FOLLOWS:

```
ERROR,,BEGIN            ;REASON FOR FAILURE.
```

6.2.2 THE "EXTENDED" ERROR PRINTOUT

```
XDCAAA  PC XXXXXX  APC YYYYYY  ERR# NNNNNN
ACSR AAAAAA  CSRC CCCCCC  STATC SSSSSS
XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX
```

THE FIRST TWO LINES OF THE EXTENDED ERROR PRINTOUT HAVE THE SAME MEANING AS THE ERROR PRINTOUT IN 6.2.1 ABOVE. THE THIRD AND ADDITIONAL LINES IF ANY,CONSIST OF UP TO EIGHT (8) OCTAL VALUES. THEY ARE PRINTED TO PROVIDE ADDITIONAL INFORMATION ON THE NATURE OF THE ERROR. THE USER MUST REFER TO THE ERRING MODULE'S DOCUMENTATION TO OBTAIN THE MEANING OF THE OCTAL VALUES PRINTED.

6.3 THE "DATA ERROR" PRINTOUT

TEST MODULES REPORT DATA ERRORS BY MEANS OF A DATA ERROR PRINTOUT WHICH IS INVOKED BY THE "DATER" CALL. THE DATA ERROR PRINTOUT LOOKS AS FOLLOWS:

```
XDCAAA  PC XXXXXX  APC YYYYYY  ERR# NNNNNN
ACSR AAAAAA  S/B BBBBBB  WAS WWWWWW  SBADR DDDDDD WASADR EEEEEE  DATA ERROR
```

WHERE:

XDCAAA	FAILING MODULE NAME.
PC XXXXXX	ACTUAL PC OF DATER CALL.
APC YYYYYY	ASSEMBLED PC OF DATER CALL.
ERR#NNNNNN	ERROR COUNT FOR CURRENT RUN. (DECIMAL).
ACSR AAAAAA	CSR ADDR OF FAILING DEVICE.
S/B BBBBBB	EXPECTED DATA (GOOD DATA)
WAS WWWWWW	OBTAINED DATA (BAD DATA)
S/BADR DDDDDD	ADDRESS OF EXPECTED DATA
WASADR EEEEEE	ADDRESS OF BAD DATA

USING THE VALUE TYPED IN APC YYYYYY THE USER SHOULD HAVE NO TROUBLE IN LOCATING IN THE MODULE LISTING THE DATER CALL. DATER CALLS ARE PRECEDED AND FOLLOWED BY ASTERISKS (*), TO MAKE THEM STAND OUT.

7. NORMAL PRINTOUTS

7.1 "ENDPAS" PRINTOUT

THE ENDPAS PRINTOUT IS USED BY A MODULE TO INDICATE THAT A PASS HAS BEEN COMPLETED. THE DEFINITION OF WHAT A "PASS" INVOLVES CAN BE FOUND IN THE DOCUMENTATION FOR EACH MODULE. FOLLOWING THE ENDPAS PRINTOUT MODULE EXECUTION CONTINUES, UNLESS:

A. THE MODULE IS A BACKGROUND MODULE, IN WHICH CASE THE MONITOR STARTS EXECUTION OF THE NEXT BACKGROUND MODULE.

B. CHAIN MODE IS ACTIVE, EACH MODULE IS ALLOWED ONE PASS ONLY.

THE ENDPAS PRINTOUT LOOKS AS FOLLOWS:

```
XDCAAA  PC XXXXXX  APC YYYYYY  ENDPAS NNNNN.
```

WHERE ENDPAS NNNNN. IS THE PASS NUMBER (DECIMAL) COMPLETED.

7.2 THE "DROPPED" PRINTOUT

THE "DROPPED" PRINTOUT IS CALLED BY MEANS OF AN "END" CALL, OR IT CAN BE GENERATED BY THE MONITOR. FOLLOWING THE "DROPPED" PRINTOUT, THE MODULE DROPPED IS NOT ALLOWED TO EXECUTE FOR THE REMAINDER OF THE EXERCISER RUN.THE "DROPPED" PRINTOUT OCCURS:

A. AFTER AN ERROR, WHETHER PRINTED OR NOT, IF SR15 IS SET TO A 1. (HALT MODULE AFTER ERROR).

B. AFTER THE 20TH ERROR, WHETHER PRINTED OR NOT, IF SR14 IS SET TO 0. SR14 SET TO A 1 INHIBITS MODULE HALT AFTER 20TH ERROR.

C. WHEN A MODULE, DUE TO AN ABNORMAL CONDITION, DETERMINES THAT IT IS BEST TO DROP FROM EXECUTION. (NO DRIVES AVAILABLE ON DECTAPE, ETC.).

7.3 "RUN SUMMARY" PRINTOUT

A "RUN SUMMARY" PRINTOUT OCCURS AT THE END OF AN EXERCISER RUN. AN EXERCISER RUN ENDS WHEN "RUN MODE" IS CLEARED BY ONE OF THE FOLLOWING:

A. DEPRESSING ^C (CONTROL C) ON THE KEYBOARD, OR
B. SYSTEM ERROR OCCURS. (SEE SECTION 6.1).

THE INTENT OF THE RUN SUMMARY IS TO INDICATE THE MODULES THAT PARTICIPATED IN THE EXERCISER RUN, THE NUMBER OF PASSES MADE BY EACH MODULE, AND THE NUMBER OF ERRORS DETECTED BY EACH MODULE. THE RUN SUMMARY IS USEFUL IN COMPARING SYSTEM PERFORMANCE AT DIFFERENT TIMES. EXAMPLE:

THE RUN SUMMARY SERVES AS A MEANS OF DETECTING MODULES THAT HAVE BECOME "HUNG", DUE TO NO INTERPUPTS RECEIVED FROM A DEVICEW. DEC/X11 DOES NOT HAVE "WATCH DOG" TIMERS. A RUN SUMMARY LOOKS AS FOLLOWS:

```
RUN SUMMARY
XDCAAA  AT XXXXXX  STAT SSSSSS  PASCNT CCCCC,  ERRCNT EEEEE,
XLPAAA  AT XXXXXX  STAT SSSSSS  PASCNT CCCCC,  ERRCNT EEEEE,
XTCAAA  AT XXXXXX  STAT SSSSSS  PASCNT CCCCC,  ERRCNT EEEEE,
XTMAAA  AT XXXXXX  STAT SSSSSS  PASCNT CCCCC,  ERRCNT EEEEE,
```

WHERE PASCNT AND ERRCNT ARE DECIMAL NUMBERS.

NOTE: TYPING A 2ND ^C WILL INHIBIT FURTHER TYPING OF RUN SUMMARY.

7.4 THE "ROTATION ENABLED" PRINTOUT

```
WRITE BUFFER ROTATION ENABLED, RANGE: XXXXXX  YYYYYY
```

THE ABOVE PRINTOUT OCCURS WHEN THE MONITOR DETERMINES THAT THERE IS SUFFICIENT "FREE CORE" ABOVE THE LAST TEST MODULE, TO PERMIT REASSIGNING THE WRITE BUFFER ADDRESS. IF NO FREE CORE EXISTS, THE PRINTOUT DOES NOT OCCUR, AND THE WRITE BUFFER IS ASSIGNED WITHIN THE ADDRESS RANGE OF THE MONITOR CODE.

IN THE DEC/X11 EXERCISER, TEST MODULES CONTAIN THEIR OWN INTERNAL READ BUFFER, BUT MUST USE THE ADDRESS ASSIGNED BY THE MONITOR AS THE STARTING ADDRESS OF THEIR WRITE BUFFER. THE WRITE BUFFER CANNOT EXCEED 1024 WORDS(10). ROTATING OF THE WRITE BUFFER ADDRESS THROUGHOUT FREE CORE HELPS TO INSURE THAT NPR TRANSFERS OCCUR FROM EVERY BANK OF FREE CORE UP TO 28K.

7.5 "PWR FAILURE" PRINTOUT

```
PWR FAILURE
```

THIS PRINTOUT OCCURS UPON RESTART FROM POWER FAILURE, TO INDICATE ITS OCCURRENCE. IF RUN MODE WAS ACTIVE AT TIME OF POWER FAILURE, RUN MODE IS REACTIVATED. IF NOT, MONITOR GOES TO COMMAND MODE TO AWAIT KEYBOARD COMMANDS.

7.6 "ASCII PRINTOUTS"

THE DEC/X11 MONITOR PROVIDES THE CAPABILITY FOR MODULES TO OUTPUT "ASCII" MESSAGES, IN addition TO THE STANDARD CANNED MESSAGES. THE ASCII MESSAGE CAPABILITY CAN BE USED BY A MODULE TO REPORT AN ERROR CONDITION, STATUS CONDITION, END OF PASS STATISTICS, ETC. EXAMPLE:

```
XLPAAA  PC XXXXXX  APC YYYYYY
LP IS OFF LINE

XRKAAA  PC XXXXXX  APC YYYYYY
DATA TRANSFERS: XXXXXX
SOFT ERRORS:    YYYYYY
HARD ERRORS:    ZZZZZZ
```

8. DESCRIPTION

THE BASIC COMPONENTS OF THE DEC/X11 PACKAGE ARE:

A. DEC/X11 MONITORS (CSXMON STANDARD MONITOR, AND UNIMON UC15 MONITOR).
B. DEC/X11 TEST MODULES.
C. DEC/X11 CONFIGURATOR/LINKER PROGRAM.

DEC/X11 HAS BEEN DESIGNED TO PROVIDE THE USER WITH THE MEANS TO CUSTOM MAKE A SYSTEM EXERCISER UNIQUELY TAILORED TO THE SYSTEM TO BE TESTED, AS OPPOSED TO ANOTHER APPROACH THAT WOULD PROVIDE A FIXED, GENERAL PURPOSE EXERCISER THAT WOULD CONTAIN ALL POSSIBLE MODULES FOR ALL DEVICES AND OPTIONS. THE DISADVANTAGES OF THE LATTER APPROACH ARE:

A. GREATER CORE REQUIREMENTS.
B. NEED TO RE-DO AND RETEST THE ENTIRE EXERCISER WHENEVER A NEW DEVICE OR OPTION BECOMES AVAILABLE.
C. GREATER INTERFACING PRORLEMS FOR CUSTOMER WRITTEN MODULES.

SOME ADVANTAGES OF THE DEC/X11 APPROACH ARE:

A. ONLY THOSE MODULES NEEDED FOR A PARTICULAR SYSTEM NEED BE INCLUDED.
B. CORE REQUIREMENTS ARE LESS.
C. WHEN A NEW DEVICE/OPTION BECOMES AVAILABLE ALL THAT NEED BE DONE IS TO WRITE AND TEST A TEST MODULE.

8.1 DEC/X11 MONITOR (CSXMON)

CSXMON IS THE CONTROL PROGRAM FOR DEC/X11. IT HAS THE FOLLOWING FUNCTIONS:

A. ACCEPTS KEYBOARD COMMANDS AND PERFORMS THEM.
B. CONTROLS EXECUTION OF UP TO 100(10) MODULES.
C. PERFORMS WRITE BUFFER ROTATION WHEN POSSIBLE.
D. IS ABLE TO RECOVER FROM POWER FAILURE.
E. PROVIDES PRINTING SERVICES FOR TEST MODULES.

8.1.1 MONITOR ROUTINES

INPUT — INPUT ROUTINE INITIALIZES THE MONITOR STACK, AND SETS UP TO ACCEPT KEYBOARD INPUT. IT THEN DROPS TO ROUTINE QUETST.

QUETST — QUETST ROUTINE CHECKS QUEUES AND INDICATORS TO SEE IF ANY ROUTINE OR MODULE REQUIRES SERVICE. QUETST CHECKS IN THE FOLLOWING PRIORITY SEQUENCE:

IOQUE REQUESTS - SERVICE IOQUE REQUEST.
TYPQUE REQUESTS - SERVICE TYPE QUEUE.
BKQUE INDICATOR - RESUME BACKGROUND MODULE.
RUN MODE INDICATOR - GO TO RUNSVC AND START MODULE.

RUNSVC — RUNSVC PERFORMS THE FOLLOWING FUNCTIONS:

A. ROTATES WRITE BUFFER POINTER.
B. QUEUES UP MODULES FOR EXECUTION ACCORDING TO FOLLOWING SCHEME:

1. ALL I/O MODULES ARE STARTED FIRST.
2. BACKGROUND MODULES ARE STARTED NEXT ONE AT A TIME. A BACKGROUND MODULE MUST RUN TO COMPLETION BEFORE THE NEXT IS STARTED. TO ACCOMPLISH THIS, LOC "BRAKE" IS SET WHENEVER A BACKGROUND MODULE IS STARTED. IT IS NOT CLEARED UNTIL THE MODULE COMPLETES ONE PASS.

IOQSVC — I/O QUEUE SERVICE ROUTINE, THIS ROUTINE SERVICES ALL REQUESTS OTHER THAN A "TYPE" REQUEST. AN IOQUE REQUEST CAN BE A REQUEST FOR DELAYED SERVICE (PIRQ CALL), OR A REQUEST TO TRANSFER CONTROL TO A PARTICULAR MODULE (MONITOR ONLY) OR ADDRESS.

IOQUE REQUESTS ARE SERVICED ON A FIRST-IN FIRST-OUT BASIS, THE I/O QUEUE CAN HOLD UP TO 200(10) REQUESTS BEFORE OVERFLOWING.

AN IOQUE REQUEST CONSISTS OF THREE WORDS AS FOLLOWS:

CALL,DESTINATION ADDR,MODULE ADDR

IOQSVC PERFORMS THE FOLLOWING ACTIONS:

A. GET DESTINATION ADDR,
B. CHECK MODULE ADDR, IF 0, GO TO DESTINATION ADDR.
(0 INDICATES THE MONITOR MADE THE CALL).
C. IF MODULE ADDR NOT 0,
D. SAVE MONITOR REGISTERS,
E. SAVE MONITOR STACK POINTER,
F. RESTORE MODULE'S REGISTERS,
G. RESTORE MODULE'S STACK POINTER,
H. GO TO MODULE'S DESTINATION ADDR.

BKQSVC — BACKGROUND MODULE SERVICE ROUTINE, BKQSVC CAUSES A BACKGROUND MODULE THAT HAS BEEN SUSPENDED TO BE RESUMED, BY FETCHING THE MODULE'S PC AND STATUS, AND THEN GOING TO IOQSVC ROUTINE TO PERFORM THE TRANSFER.

TYPSVC — TYPSVC SERVICES THE TYPE QUEUE, THE TYPE QUEUE IS 300(10) WORDS LONG, TYPSVC SERVICES THE FOLLOWING TYPE REQUESTS:

MSG,	MESSAGE
END	DROPPED MODULE PRINTOUT
ENDPAS	END OF PASS PRINTOUT
ERROR	ERROR PRINTOUT
DATERR	DATA ERROR PRINTOUT

MSG, — MSG CALL SERVICE ROUTINE, OUTPUTS WHATEVER ASCII MESSAGE THE MONITOR REQUIRES.

MSGN, — OUTPUTS MODULES' ASCII MESSAGES.

ENDSVC — END CALL SERVICE ROUTINE, OUTPUTS MODULE DROPPED MESSAGE

PASEND — ENDPAS CALL SERVICE ROUTINE, OUTPUTS END OF PASS PRINTOUT.

ERRSVC
ERSVC1
ERSVC2 — ERROR, ERRORN, AND DATERR CALL SERVICE ROUTINES.

TRCI — TRACE TRAP SERVICE ROUTINE, BACKGROUND MODULES TRAP TO THIS ROUTINE AFTER EACH MODULE INSTRUCTION, TRCI CHECKS TO SEE IF ANY IOQUE OR TYPQUE REQUESTS ARE PENDING, IF NOT, TRCI PERFORMS AN RTI OR RTT INSTRUCTION TO RETURN TO THE BACKGROUND MODULE, IF YES, TRCI SAVES THE MODULE'S PC AT TRCPC, AND THE STATUS AT TRCPSW, IT THEN GOES TO ROUTINE EXIT.

EXIT. — ROUTINE EXIT, IS ENTERED WHENEVER A MODULE MUST GIVE UP CONTROL TO THE MONITOR (EXIT CALL, PIRQ CALL), EXIT, SAVES THE MODULE'S REGISTERS AND STACK POINTER, RESTORES THE MONITOR'S REGISTERS AND STACK POINTER, AND THEN GOES TO ROUTINE QUETST TO CHECK QUEUES.

TYPQ.	THIS ROUTINE QUEUES UP END AND ENDPAS CALLS ON THE TYPE QUEUE AND THEN PERFORMS AN AUTOMATIC EXIT. (MODULE DOES NOT GET CONTROL UNTIL MESSAGE IS PRINTED).
TYPQ1.	QUEUES UP MSG CALL (MONITOR ONLY), AND THEN GOES TO ROUTINE QUETST.
TYPQ2.	QUEUES UP ERROR AND DATERR CALLS IN TYPQ QUEUE, AND PERFORMS AUTOMATIC EXIT.
PIRQ.	QUFUES UP PIRQ CALL IN I/O QUEUE, AND THEN PERFORMS AN RTI TO EXIT MODULE'S INTERRUPT SERVICE.
QUE.	QUEUES UP QUE CALL (MONITOR ONLY) AND GOES TO QUEIST.
KBSVRC	KEYBOARD SERVICE ROUTINE.
CTRLCA CTRLCB	CTRL C (^C) SERVICE ROUTINE. IF NOT IN RUN MODE, ISSUES RESET, CLEARS QUEUES, OUTPUTS ^C, AND GOES TO ROUTINE "INPUT". IF IN RUN MODE, ISSUES RESET, CLEARS QUEUES, OUTPUTS ^C, CHECKS FOR CHAIN MODE, IF IN CHAIN MODE, GOES TO ROUTINE CHNOUT TO EXIT TO DDP MONITOR. IF NOT IN CHAIN MODE, OUTPUTS RUN SUMMARY.
TYPE	TYPE SUBROUTINE. OUTPUTS ASCII STRINGS TO TELEPRINTER.
COMTAB	TABLE OF VALID KEYBOARD COMMANDS AND POINTERS TO DESIRED ROUTINES.
DECODE	COMMAND DECODER ROUTINE. USES CONTENTS OF COMTAB TO DETERMINE COMMAND TO BE EXECUTED.

RUN	ROUTINE TO SET UP "RUN MODE". PERFORMS THE FOLLOWING: A. CLEARS MODULE COUNTER. B. IF NOT IN CHAIN MODE, OR UP FROM POWER FAIL, CLEARS MODULES' PASCOUNT AND ERROR COUNTERS, AND STOP BITS. C. INCREMENTS MODULE COUNTER FOR EACH MODULE FOUND READY TO RUN (SELECTED, NOT STOPPED). D. IF MODULE COUNTER NOT 0, SETS RUN MODE INDICATOR, AND GOES TO QUETST. E. IF MODULE COUNTER IS 0, TYPES "INVALID COMMAND" MESSAGE.
MAP	TYPES DIRECTORY OF CORE RESIDENT MODULES, THEIR START ADDRESS, AND THEIR STATUS.
SEL	ROUTINE TO SELECT A MODULE OR ALL MODULES. SETS BIT14 OF LOC "STAT" IN THE MODULE.
DES	ROUTINE TO DESELECT A MODULE OR ALL MODULES. CLEARS BIT14 OF LOC "STAT" IN MODULE.
MOD	MODIFY ROUTINE. EXAMINES (TYPES OUT) CONTENTS OF A CORE LOCATION (EVEN), AND CHANGES CONTENTS TO NEW VALUE IF DESIRED (VIA KEYBOARD).
CLRQUS	ROUTINE TO CLEAR VARIABLES, QUEUES, AND TO FILL VECTOR AREA WITH .+2 AND HALT.
RUSERR RESINT	ROUTINES TO OUTPUT "SYS ERROR" PRINTOUT.
PWRDN	UPON POWER FAILURE, POINTS POWER FAIL VECTOR TO PWRUP ROUTINE, SAVES CONTENTS OF RUN MODE INDICATOR, AND HALTS.
PWRUP	UPON POWER FAIL RESTART, POINTS POWER FAIL VECTOR TO PWRDN ROUTINE, RESETS STACK, CLEARS QUEUES, AND OUTPUTS "POWER FAILURE" MESSAGE. IF RUN MODE WAS ACTIVE, GOES TO "RUN" ROUTINE. IF RUN MODE WAS NOT ACTIVE, GOES TO "INPUT"

8.2 DEC/X11 MODULES

TEST MODULES WRITTEN FOR DEC/X11 ARE RELOCATABLE OBJECT MODULES THAT MUST BE LINKED TOGETHER WITH THE MONITOR IN ORDER TO PRODUCE A USABLE ABSOLUTE FORMAT FILE LOADABLE BY THE "ABS" LOADER.

TWO TYPES OF MODULES CAN BE WRITTEN: I/O MODULES (IOMOD), AND BACKGROUND MODULES (BKMOD).

8.2.1 I/O MODULES (IOMOD)

AN I/O MODULE (IOMOD) IS DEFINED AS ONE THAT ONCE STARTED BY THE MONITOR IS DRIVEN STRICTLY BY INTERRUPTS AND RUNS CONTINUOUSLY. AN IOMOD DEPENDS ON EXPECTED INTERRUPTS TO OCCUR IN ORDER TO CONTINUE EXECUTION. IF AN EXPECTED INTERRUPT SHOULD NOT OCCUR, THE MODULE BECOMES "HUNG", THERE ARE CURRENTLY NO MEANS TO DETECT A HUNG MODULE, EXCEPT BY THE OPERATOR NOTING THAT ENDPAS PRINTOUTS ARE NO LONGER OCCURRING, AND BY THE RUN SUMMARY PRINTOUT. AN IOMOD DOES NOT RUN IN TRACE MODE.

8.2.2 TRACE MODE BACKGROUND MODULES (BKMOD)

A BACKGROUND MODULE CAN BE INTERRUPT DRIVEN, IN WHICH CASE IT ACTS VERY MUCH LIKE AN IOMOD (IT CAN GET HUNG), OR IT CAN CONSIST OF NON-INTERRUPTING CODE. A BACKGROUND MODULE IS RUN IN TRACE MODE (A TRACE TRAP OCCURS AFTER EVERY MODULE INSTRUCTION), IN ORDER TO PERMIT SERVICING I/O MODULES. BKMODS ARE RUN ONE MODULE AT A TIME.

8.2.3 NON-TRACE MODE BACKGROUND MODULES (NBKMOD)

NBKMOD MODULES ARE RUN ONE AT A TIME BEFORE ANY OTHER TYPE OF MODULE CAN BE RUN. NBKMOD MODULES DO NOT RUN IN TRACE MODE. THEIR MAIN FUNCTION IS TO RUN FIRST IN ORDER TO SET UP SPECIAL CONDITIONS. FOR EXAMPLE: A PARITY MODULE WOULD RUN TO INSURE THAT ALL PARITY MEMORY HAS CORRECT PARITY, AND THEN WOULD TERMINATE. FROM THAT POINT ON, THE PARITY MODULE WOULD AWAKEN ONLY IN CASE OF A PARITY ERROR.

8.2.4 MODULE ORGANIZATION

TEST MODULES ARE ORGANIZED IN TWO SECTIONS:

1. MODULE FRONT END INTERFACE,
2. MODULE CODE ITSELF.

8.2.5 MODULE FRONT END INTERFACE

A MODULE'S FRONT END INTERFACE (SEE APPENDIX A) IS REQUIRED BY THE MONITOR IN ORDER TO CONTROL OPERATION OF THE MODULE. THE MODULE'S INTERFACE CONSISTS OF 56 WORDS USED AS FOLLOWS:

MODNAM: 6 BYTES (3 WORDS), MODULE NAME IN ASCII.
ADDR: 1 WORD, CONTAINS ADDRESS OF FIRST REGISTER OF DEVICE TO BE TESTED.
VECTOR: 1 WORD, CONTAINS ASSIGNED DEVICE VECTOR.
BR1: 1 BYTE, 1ST BR LEVEL.
BR2: 1 BYTE, 2ND BR LEVEL IF ANY.
DVID1: 1 WORD, DEVICE COUNT, USED TO INDICATE NUMBER OF DRIVES, OR DEVICE MULTIPLES TO BE TESTED, ONE BIT IS SET FOR EACH ONE.
EXAMPLE1: IF A MAGTAPE CONTROL HAS 8 DRIVES, BITS 0 THROUGH 7 OF DVID1 WOULD BE SET, BIT 0 INDICATING DRIVE 0, AND BIT7 INDICATING DRIVE 7.
EXAMPLE2: IF A MODULE TESTING A DC11 IS TO TEST 16 DC11'S, ALL BITS WOULD BE SET IN DVID1, DVID1 BIT0 WOULD CORRESPOND TO DC11 #0, AND DVID1 BIT15 WOULD REPRESENT DC11 #15.

SR1: 1 WORD, INTERNAL SWITCH REGISTER FOR MODULE.

STAT: 1 WORD, MODULE STATUS WORD, HIGH ORDER BITS PROVIDE INFORMATION ABOUT THE MODULE AS FOLLOWS:

BIT15 = 1	MODULE IS AN I/O MODULE,
BIT15 = 0	MODULE IS BACKGROUND MODULE,
BIT14 = 1	MODULE IS SELECTED FOR RUNNING,
BIT14 = 0	MODULE IS NOT SELECTED FOR RUNNING,
BIT13 = 1	MODULE HAS BEEN STOPPED,
BIT13 = 0	MODULE HAS NOT BEEN STOPPED,

THE LOW ORDER BYTE IS USED TO INDICATE THE PROCESSOR STATUS TO BE USED WHEN GIVING CONTROL TO THE A MODULE. THE STATUS IS 0 FOR IOMODS AND NBKMODS, AND 20 FOR BKMODS (TRACE MODE).

INIT:	1 WORD, CONTAINS THE MODULE'S START ADDR.
SPOINT:	1 WORD, CONTAINS ADDR TO LOAD IN STACK POINTER WHEN FIRST STARTING THE MODULE.
PASCNT:	1 WORD, PASS COUNTER.
ERRCNT:	1 WORD, ERROR COUNTER.
SVR0 - SVR6:	6 WORDS, LOCATIONS TO SAVE CONTENTS OF MODULE'S REGISTERS AND STACK POINTER WHEN MODULE GIVES CONTROL TO THE MONITOR.
CSRA:	1 WORD, CONTAINS ADDR OF FAILING DEVICE CSR.
SBADR/ACSR:	1 WORD, WHEN DATA ERROR OCCURS, CONTAINS ADDRESS OF GOOD DATA, WHEN ERROR CALL OCCURS, CONTAINS CONTENTS OF FAILING DEVICE CSR.
WASADP/ASTAT:	1 WORD, CLEARED AFTER ERROR PRINTOUT, WHEN DATA ERROR OCCURS, CONTAINS ADDR OF BAD DATA, IF ERROR, CONTAINS CONTENTS OF FAILING DEVICE STATUS REGISTER, IF APPLICABLE.
ASB:	1 WORD, CLEARED AFTER ERROR PRINTOUT, CONTAINS EXPECTED GOOD DATA.
AWAS:	1 WORD, CLEARED AFTER ERROR PRINTOUT, CONTAINS ACTUAL DATA, (BAD DATA).
LOC 64-162	32 WORDS, MODULE'S STACK, WHEN A MODULE RUNS, IT OPERATES ON ITS OWN STACK.

8.2.6 MODULE CODE

THE MODULE'S CODE CONSISTS OF STANDARD PDP-11 CODE, WITH THE FOLLOWING RESTRICTIONS LISTED BELOW. ADDITIONAL INFORMATION IN CODING DEC/X11 MODULES IS DESCRIBED IN MAINDEC-11-DXQAE, MODULE PROGRAMMER'S GUIDE.

A. CODE MUST EXECUTE IN ALL PDP-11 FAMILY PROCESSORS.
B. NO HALT INSTRUCTIONS.
C. NO WAIT INSTRUCTIONS.
D. NO EMT CALLS.
E. NO TRAP CALLS EXCEPT FOR THOSE SPECIFIED IN SECTION 8.2.7
F. NO PROCESSOR STATUS WORD MODIFICATIONS.
G. I/O MODULES MUST NOT PERFORM WAITING LOOPS THAT INHIBIT OTHER MODULES FROM RUNNING.
H. GENERAL REGISTERS ARE TO BE USED IN INTERRUPT SEQUENCES ONLY AFTER FIRST BEING SAVED, AND MUST BE RESTORED PRIOR TO EXITING THE INTERRUPT SEQUENCE.
I. THE STACK POINTER MUST NOT BE MODIFIED IN ORDER TO EXIT AN INTERRUPT SEQUENCE (USE PIRQ CALL).

ESPECIALLY IN THE CASE OF AN IOMOD, MODULE CODE CAN BE BROKEN DOWN INTO 3 SECTIONS:

A. INITIALIZATION. CODE REQUIRED TO SET UP THE TEST, AND TO ISSUE THE FIRST I/O COMMAND. CODE IS TERMINATED WITH AN EXIT CALL TO THE MONITOR. MODULE DOES NOT REGAIN CONTROL UNTIL INTERRUPT OCCURS.

B. INTERRUPT SERVICE. EXCEPT FOR DEVICES THAT HAVE BR LATENCY PROBLEMS, THIS IS THE CODE REQUIRED TO ACKNOWLEDGE THE FACT THAT AN INTERRUPT HAS BEEN RECEIVED, AND TO QUEUE UP A REQUEST TO SERVICE THE INTERRUPTING DEVICE AT A LATER TIME. THE PHILOSOPHY APPLIED SAYS THAT MODULES MUST EXECUTE ONLY A MINIMAL AMOUNT OF CODE AT A PROCESSOR STATUS OTHER THAN 0 IN ORDER TO PREVENT LOCKING OUT OTHER DEVICES FROM INTERRUPTING. QUEUEING UP FOR DEFERRED SERVICE IS ACCOMPLISHED BY MEANS OF THE PIRQ CALL. THE PIRQ CALL REQUESTS THE MONITOR TO GIVE CONTROL TO THE MODULE AT A SPECIFIED ADDRESS AT ITS EARLIEST OPPORTUNITY. THE MONITOR STORES THE REQUEST, AND THEN PERFORMS AN RTI INSTRUCTION TO EXIT THE MODULE'S INTERRUPT SERVICE SEQUENCE.
MODULES WITH BR LATENCY PROBLEMS MUST SERVICE THEIR DEVICE AT THE INTERRUPTING STATUS, BUT MUST MAKE THE DEVICE SERVICE AS SHORT AS POSSIBLE, AND THEN EXIT WITH AN RTI INSTRUCTION, IF AN ABNORMAL CONDITION IS ENCOUNTERED, THEN THE SERVICE OF THAT CONDITION MUST BE DEFERRED, AND A PIRQ CALL IS USED TO EXIT THE INTERRUPT SEQUENCE.

C. DEVICE SERVICE. THIS CODE IS EXECUTED AFTER THE INTERRUPT SERVICE SEQUENCE. IT CONSISTS OF THE CODE REQUIRED TO SEE THAT AN I/O OPERATION HAS OCCURRED SUCCESSFULLY, TO SERVICE ABNORMAL CONDITIONS, AND TO PREPARE AND ISSUE THE NEXT I/O COMMAND.

8.2.7 MONITOR CALLS

A DEC/X11 MODULE COMMUNICATES WITH THE MONITOR VIA MONITOR CALLS, WHICH ARE CODED TRAP CALLS. EXISTING MONITOR CALLS ARE:

EXIT CALL: SHOWN IN LISTING AS EXIT. ;RETURN TO MONITOR.

THE EXIT CALL IS USED BY THE MODULE TO RETURN CONTROL TO THE MONITOR. IT IS GIVEN AFTER AN I/O COMMAND HAS BEEN GIVEN AND THE MODULE HAS NOTHING TO DO BUT WAIT FOR AN INTERRUPT.

ERROR CALL: SHOWN IN LISTING AS ERROR.,BEGIN ;REASON FOR CALL.
USED BY MODULE TO REPORT AN ERROR OTHER THAN A DATA ERROR.

EXTENDED ERROR CALL: SHOWN IN LISTING AS ERRN.,ADR,BEGIN ;REASON
USED TO OUTPUT ADDITIONAL ERROR DATA.

DATERR CALL: SHOWN IN LISTING AS DATER.,BEGIN ;DATA ERROR.
USED TO REPORT A DATA ERROR.

MSGN CALL: SHOWN IN LISTING AS MSGN.,ADR,BEGIN .USED BY MODULE TO OUTPUT ASCII MESSAGES.

ENDPAS CALL: SHOWN IN LISTING AS ENDPS.,ADDR,BEGIN ;END OF PASS.

WHERE:
ADDR IS ADDRESS TO START NEXT PASS.
BEGIN IS MODULE ADDRESS.

USED BY MODULE TO INDICATE END OF PASS.

END CALL: SHOWN IN LISTING AS END.,BEGIN ;REASON FOR END CALL.

USED BY MODULE TO REQUEST THAT MODULE BE DROPPED FROM EXECUTION DUE TO AN ABNORMAL CONDITION.

THE USE OF THE ABOVE MONITOR CALLS IS FURTHER DESCRIBED IN SECTION 3 OF MAINDEC-11-DXQBA DEC/X11 CONFIGURATION AND PROGRAMMING MANUAL.

9. SPECIAL MODIFICATIONS

9.1 FIXING WRITE BUFFER ADDRESS

IN SYSTEMS WHERE WRITE BUFFER ROTATION HAS BEEN ENABLED (PRINTOUT), ROTATION MAY BE INHIBITED BY ZEROING BYTE LOCATION "ROTI" IN THE MONITOR. ADDITIONALLY, IF THE USER WANTS TO SET THE WRITE BUFFER ADDRESS TO A SPECIFIC VALUE HE MAY DO SO BY CHANGING THE CONTENTS OF LOC 56 (WORD). THE VALUE MUST BE EVEN, AND WITHIN THE RANGE SPECIFIED BY THE "WRITE BUFFER ROTATION ENABLED" PRINTOUT.

9.2 MODIFYING MODULE LOC "DVID1"

MODULE LOCATION "DVID1" MAY BE CHANGED TO OTHER THAN ITS USUAL VALUE BY MEANS OF THE "MOD" COMMAND IN ORDER TO RUN LESS THAN A FULL COMPLEMENT OF DEVICES. EXAMPLE:

MODULE XTCAAA HAS BEEN CONFIGURED FOR 8 DRIVES. LOC DVID1 THEREFORE, CONTAINS THE VALUE 000377. TO RUN THE MODULE WITH ONLY DRIVES 0 AND 1, CHANGE THE VALUE IN DVID1 TO 000003.

9.3 HALT AFTER 20 ERRORS

THE MONITOR NORMALLY WILL DROP A MODULE AFTER 20 ERRORS UNLESS PREVENTED BY SR14. THE NUMBER MAY BE INCREASED OR DECREASED BY CHANGING LOCATION "ERRLIM" BY MEANS OF THE "MOD" COMMAND.

9.4 HARD HALT ON ERROR

TO HALT PROCESSOR UPON ERROR OR DATA ERROR, PLACE A HALT IN MONITOR LOCATION "TYPQ2,".

9.5 HARD HALT ON ERROR TRAP

TO HALT ON ERROR TRAP INSTEAD OF TYPING "SYS ERROR" MESSAGE, CHANGE CONTENTS OF LOC4 TO 6, 6 TO 0, 10 TO 12, AND 12 TO 0. USEFUL WHEN USER WISHES TO EXAMINE CONTENTS OF STACK WHEN ERROR TRAP OCCURS.

10. DEBUGGING AIDS

PROBLEM 1. MODULE X FAILS, FIVE OTHER MODULES RUNNING AT TIME THAT FAILURE OCCURS.

PROCEDURE: RUN MODULE X ALONE, IF FAILURE REOCCURS, ISOLATE PROBLEM WITH MODULE X, OR USE DEVICE/OPTION DIAGNOSTIC. IF PROBLEM DOES NOT SHOW, ADD MODULES UNTIL THE PROBLEM REOCCURS. GOAL: CAUSE FAILURE TO OCCUR WITH MINIMUM NUMBER OF MODULES.

COMMENT: CERTAIN COMBINATIONS OF HARDWARE MAY NOT RUN SUCCESSFULLY AT THE SAME TIME.

PROBLEM 2. MODULE X HAS NOT PRINTED ENDPAS PRINTOUT, OR ANY OTHER PRINTOUT SINCE THE RUN STARTED, IS IT RUNNING?

PROCEDURE. MAKE SURE THAT MODULE IS SELECTED (CHECK RUN SUMMARY). IF SELECTED, SET HALTS , ONE AT A TIME, IN THE MODULE CODE, AND RUN. THE INTENT IS TO TRACE EXECUTION OF THE MODULE CODE UNTIL REASON FOR MODULE HANGUP IS FOUND.

PROBLEM 3. BACKGROUND MODULE Y HAS NOT PRINTED ENDPAS PRINTOUT SINCE RUN STARTED.

PROCEDURE. MAKE SURE MODULE IS SELECTED. (LOOK AT RUN SUMMARY). BACKGROUND MODULES ARE RUN ONE AT A TIME. DEPENDING ON NUMBER OF OTHER BACKGROUND MODULES PRESENT, ITS TURN MAY NOT HAVE COME YET. ALSO, BACKGROUND MODULES ARE SERVICED AT A LOWER PRIORITY THAN I/O MODULES. THE NUMBER OF I/O MODULES ACTIVE WILL AFFECT SPEED OF EXECUTION OF BACKGROUND MODULES.

PROBLEM 4. PROCESSOR HALTS IN VECTOR AREA (60-774)

PROCEDURE. RUN EACH MODULE ALONE UNTIL FAILURE REOCCURS. CHECK THE OFFENDING MODULE'S DEVICE'S INTERRUPT CARD FOR CORRECT VECTOR. EITHER THE DEVICE OR THE MODULE HAS AN INCORRECT VECTOR SPECIFIED.

APPENDIX A, MODULE INTERFACE SAMPLE

```
        IOMOD   <SAMPL >,123456,200,7,6
        MODULE  140000,SAMPL ,123456,200,7,6
        .TITLE  SAMPL
;*************************************************************************
        BEGIN:
MODNAM: .ASCII  /SAMPL /                ;MODULE NAME.
ADDR:   123456+0                        ;1ST DEVICE ADDR.
VECTOR: 200+0                           ;1ST DEVICE VECTOR.
BR1:    .BYTE   PRTY7+0                 ;1ST BR LEVEL.
BR2:    .BYTE   PRTY6+0                 ;2ND BR LEVEL.
DVID1:  1                               ;DEVICE INDICATOR 1.
SR1:    OPEN                            ;SWITCH REGISTER 1
;*************************************************************************
STAT:   140000                          ;STATUS WORD.
INJT:   START                           ;MODULE START ADDR.
SPOINT: MODSP                           ;MODULE STACK POINTER.
PASCNT: 0                               ;PASS COUNTER.
ERRCNT: 0                               ;ERROR COUNTER.
SVR0:   OPEN                            ;LOC TO SAVE R0.
SVR1:   OPEN                            ;LOC TO SAVE R1.
SVR2:   OPEN                            ;LOC TO SAVE R2.
SVR3:   OPEN                            ;LOC TO SAVE R3.
SVR4:   OPEN                            ;LOC TO SAVE R4.
SVR5:   OPEN                            ;LOC TO SAVE R5.
SVR6:   OPEN                            ;LOC TO SAVE R6.
CSRA:   OPEN                            ;ADDR OF CURRENT CSR.
SBADR:                                  ;ADDR OF GOOD DATA, OR
ACSR:   OPEN                            ;CONTENTS OF CSR.
WASADR:                                 ;ADDR OF BAD DATA, OR
ASTAT:  OPEN                            ;STATUS REG CONTENTS.
ASB:    OPEN                            ;EXPECTED DATA.
AWAS:   OPEN                            ;ACTUAL DATA.
        .REPT   SPSIZ                   ;MODULE STACK STARTS HERE.
        .NLIST
        .WORD   0
        .LIST
        .ENDR
MODSP:
;*************************************************************************
;
.REM    1
```

```
*****************************************
*                                       *
*              SECTION 2                *
*                                       *
*  UC15 MONITOR (UNIMON) DIFFERENCES    *
*                                       *
*****************************************
```

TABLE OF CONTENTS

1. ABSTRACT

THIS SECTION DOCUMENTS THE DIFFERENCES BETWEEN THE DEC/X11 STANDARD MONITOR MAINDEC-11-DXQAB, AND THE SPECIALIZED VERSION FOR THE UC15; MAINDEC-11-DXQAC.

IN THE UC15 SYSTEM THE PDP-11 DOES NOT HAVE A CONSOLE DEVICE. UNIMON PASSES ALL ITS TTY MESSAGES TO THE PDP-15 PROCESSOR CONSOLE DEVICE (VIA THE PDP-15 SYSTEM EXERCISER MODULE TTY11). BECAUSE THERE IS NO CONSOLE DEVICE ON THE PDP-11, NO KEYBOARD COMMANDS ARE USED TO CONTROL THE DEC/X11 EXERCISER. THE PDP-11 SWITCH REGISTER IS USED IN PLACE OF THE KEYBOARD COMMANDS. ADDITIONAL DIFFERENCES ARE:

A. UNIMON DOES NOT PERMIT CHAINING

B. THE MOD (MODIFY) COMMAND IS NOT IMPLEMENTED

C. SYS ERROR PRINTOUTS ARE REPLACED BY THE PROCESSOR HALTING IN THE TRAP AREA

D. UNIMON'S WRITE BUFFER (WHICH IS BUILT BY THE 15'S TTY11 MODULE AND MAY BE USED TO TEST 18 BIT NPR DEVICES) IS NOT ROTATED

2. REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

2.1.1 PAPER TAPE CONFIGURATION

SAME AS CSXMON

2.1.2 DECTAPE CONFIGURATION

N/A TO UNIMON

2.1.3 RK11 DISK CARTRIDGE CONFIGURATION

N/A TO UNIMON

2.2 SOFTWARE REQUIREMENTS

2.2.1 PAPER TAPE CONFIGURATION

SAME AS CSXMON

2.2.2 DECTAPE CONFIGURATION

N/A TO UNIMON

2.2.3 RK11 DISK CARTRIDGE CONFIGURATION

N/A TO UNIMON

3. LOADING PROCEDURE

3.1 LOADING FROM PAPER TAPE

LOAD THE EXERCISER MODULE BY MEANS OF THE SPECIAL PDP-15
ABSL11 LOADER

3.2 LOADING FROM DECTAPE

N/A TO UNIMON

3.3 LOADING FROM RK11 DISK CARTRIDGE

N/A TO UNIMON

4. STARTING PROCEDURE

TO START: LOAD ADDR 000200 SET SWITCHES 8&9=11 TO OBTAIN CORE MAP
PRESS START
PROGRAM WILL HALT
PRESS CONTINUE
PROGRAM WILL TYPE CORE MAP AND HALT
SELECT/DESELECT APPROPRIATE MODULES (SEE SEC 5.1 BELOW)
AFTER ALL MODULE HAVE BEEN SELECTED/DESELECTED
SET SWITCHES 8&9=00 PRESS CONTINUE
PROGRAM IS NOW RUNNING SELECTED MODULES.

TO RESTART: LOAD ADDR 001000 AND PRESS START.

IN EITHER A START OR RESTART, THE MONITOR TYPES THE FOLLOWING
MESSAGE:

DEC/X11 EXERCISER
.

THE DOT (.) INDICATES THAT THE MONITOR IS READY TO ACCEPT OPERATOR
COMMANDS FROM THE SWITCH REGISTER.

NOTE: THE UNIMON DEC/X11 EXERCISER ABSOLUTELY MUST NOT BE STARTED OR RESTARTED BEFORE THE PDP-15 EXERCISER HAS BEEN GIVEN THE EXECUTE (X) COMMAND AND TYPED THE MESSAGE TTY11 000001, AND IF THE UC15 MODULES WILL BE RUN, THE MESSAGE UC15 000001. THIS PROCEDURE IS NECESSARY TO SYNC UP BOTH EXERCISERS AND MUST BE FOLLOWED.

NOTE: PDP15 AC SWITCHES 1-3 MUST BE DOWN UNTIL ABOVE MESSAGES ARE TYPED.

5. OPERATING PROCEDURE

THE UNIMON EXERCISER IS CONTROLLED BY MEANS OF THE PDP-11 SWITCH REGISTER (SR).

5.1 SWITCH REGISTER OPTIONS AND CONTROL FUNCTIONS

SWITCH REGISTER OPTIONS APPLY ONLY DURING EXECUTION OF THE EXERCISER.

SR15 = 1 "HALT" MODULE AFTER ERROR. THE FAILING MODULE IS PREVENTED FROM FUTHER EXECUTION. NORMALLY, A "MODULE DROPPED" PRINTOUT PRECEDES HALTING OF THE MODULE.

SR14 = 1 INHIBIT MODULE HALT AFTER 20 ERRORS. SR14 SET TO A 1 PREVENTS THE MONITOR FROM HALTING THE FAILING MODULE AFTER 20 ERRORS. IF SET TO A 0, SR14 WILL ENABLE THE MONITOR TO HALT THE MODULE AFTER THE 20TH ERROR, AFTER A "MODULE DROPPED" MESSAGE.

SR13 = 1 INHIBIT ERROR PRINTOUTS.

SR12 = 1 INHIBIT "END OF PASS" PRINTOUTS.

SETTING THE SR TO 074000 INHIBITS ALL PRINTOUTS, AND PREVENTS MODULE HALTS, IMPROVING THE CHANCES OF PERFORMING SCOPING OPERATION

THE CONTROL FUNCTIONS ARE:

SR10 = 1 (^C)TYPES RUN SUMMARY AND HALTS

SR8 & 9 = 00 = RUN ALL SELECTED MODULES
01 = SELECT MODULE SPECIFIED IN SR 7-0 TO BE RUN
10 = DESELECT MDOULE SPECIFIED IN SR 7-0 FROM RUNNING
11 = MAP = TYPE AVAILABLE MODULES AND THEIR STATUS

SR7 - 0 = TO SEQUENCE NUMBER OF DESIRED MODULE (NUMBER WHICH IS OUTPUT DURING MAP), WILL SEL OR DES THAT MODULE. WHEN EQUAL TO 0, ALL MODULES WILL BE SELECTED OR DESELECTED.

NOTE: WHEN THE OPERATOR DESIRES TO ISSUE A ^C TO UNIMON, HE MUST NOT FIRST ISSUE A ^C TO THE PDP-15 SYSTEM EXERCISER AS IT MUST CONTINUE TO RUN IN ORDER TO OUTPUT THE UNIMON MESSAGES, AFTER UNIMON OUTPUTS THE (.) THE PDP-15 EXERCISER MAY BE GIVEN THE ^C. THE UNIMON EXERCISER WILL HALT AFTER THE ^C (PDP-11) COMMAND. IF THE OPERATOR DESIRES TO CONTINUE THE EXERCISERS FROM THIS POINT WITHOUT RESTARTING UNIMON OR RELOADING THE PDP-15 MODULES, HE MUST ISSUE THE EXECUTE (X) COMMAND TO THE PDP-15 EXERCISER, WAIT UNTIL IT IS AGAIN RUNNING AND THEN AND ONLY THEN PRESS CONTINUE TO START THE UNIMON EXERCISER RUNNING AGAIN. THIS PROCEDURE IS NECESSARY TO KEEP THE TWO EXERCISERS RUNNING IN SYNC, AND IF IT IS NOT FOLLOWED, THE EXERCISERS WILL BOTH HAVE TO BE RESTARTED FROM SCRATCH TO AGAIN SYNC THEM UP.

5.2 KEYBOARD COMMANDS

N/A TO UNIMON

5.2.1 THE "MAP" COMMAND

SAME AS CSXMON EXCEPT THAT A MODULE NUMBER IS TYPED JUST BEFORE THE MODULE NAME. THE NUMBER IS USED WHEN IT IS NECESSARY TO REFER TO A MODULE VIA THE PDP-11 SWITCH REGISTER.

5.2.2 THE SEL(ECT) COMMAND

LOAD MODULE NUMBER OBTAINED FROM MAP COMMAND INTO SR0-7
SET SWITCHES 8&9=01 AND PRESS CONTINUE
IF YOU DESIRE TO SELECT ALL MODULES
SET SR0-7 =0, SR 8&9=01 AND PRESS CONT.

5.2.3 THE DES(ELECT) COMMAND

LOAD MODULE NUMBER OBTAINED FROM MAP COMMAND INTO SR0-7
SET SWITCHES 8&9=10 AND PRESS CONTINUE
IF YOU DESIRE TO DELECT ALL MODULES
SET SR0-7=0, SR 8&9=10 AND PRESS CONTINUE

5.2.4 THE MOD(IFY) COMMAND

N/A TO UNIMON

5.2.5 THE "RUN" COMMAND

SAME AS CSXMON EXCEPT THAT RUN IS INDICATED VIA SWITCH REG.

5.2.6 ENDING EXERCISER "RUN"

SET SP 10=1

5.2.7 HANDLING OF POWER FAILURE

SAME AS CSXMON

5.2.8 "CHAIN" OPERATION OF DEC/X11

N/A TO UNIMON

6. ERRORS

6.1 SYSTEM ERROR

UNIMON HALTS IN THE TRAP AREA UPON DETECTING A SYSTEM ERROR.

6.2 THE ERROR PRINTOUT

SAME AS CSXMON

6.3 THE "DATA ERROR" PRINTOUT

SAME AS CSXMON

7. NORMAL PRINTOUTS

7.1 "ENDPAS" PRINTOUT

SAME AS CSXMON EXCEPT THAT CHAIN MODE IS N/A TO UNIMON

7.2 THE "DROPPED" PRINTOUT

SAME AS CSXMON

7.3 "RUN SUMMARY" PRINTOUT

SAME AS CSXMON

7.4 THE "ROTATION ENABLED" PRINTOUT

N/A TO UNIMON

7.5 "PWR FAILURE" PRINTOUT

PWR FAILURE

SAME AS CSXMON

8.0 PROGRAM RESTRICTIONS

PDP15 SYSTEM EXERCISER MODULES SHOULD NOT BE LOCATED ABOVE:

1) 24K FOR 4K PDP11 LOCAL MEMORY
2) 20K FOR 0K PDP11 LOCAL MEMORY

PDP15 AC SWITCHES 1-3 SHOULD BE DOWN UNTIL
TTY RUN STATEMENT AND
UC15 RUN STATEMENT (IF LOADED) IS
TYPED

APPENDIX A SAMPLE LOAD AND RUN PROCEDURE

LOADED THE DECX11 PROGRAM IN THE PDP-11 USING THE ABSL11 LOADER. I THEN LOADED AND STARTED SYSTST IN THE PDP-15.

```
SYSTST V1D
$L

*SYSTEM LOADER V2B

MEMSIZE TYPE 8K,12K,16K,20K,24K,28K,OR 32K:  16K
TITLE 01 DECTAP
TITLE 02 FP15T2
TITLE 03 EAEPT2
TITLE 04 XRLR
TITLE 05 TTY11
TITLE 06 UC15
TITLE 07

DECTAP  031247
FP15T2  023057
XR/LR   020064
TTY11   015223
UC15    030031
EAEPT2  014275


SYSTST V1D
$P

*PARAMETER MODE
01  DECTAP  600004 000000 000000 000000 400000
02  FP15T2  ^C

SYSTST V1D
$X

*OPERATING SYSTEM V3B

API ON

TTY11   000001

UC15    000001
```


STARTED DECX11 IN THE PDP-11 AT 200 WITH SWITCHES 8 AND 9 SET TO OBTAIN THE CORE MAP SHOWN BELOW.

```
DEC/X11 EXERCISER

THE PROCESSOR STOPPED AND I PRESSED CONTINUE

000001  XUCAAA AT 007504 STAT 040020
000002  XRKAAA AT 011102 STAT 140000
000003  XLPAAA AT 012162 STAT 140000
000004  XCRAAA AT 012716 STAT 140000
```

THE PROCESSER STOPPED AFTER TYPING THE MODULES LOADED BY DECX11 SHOWN ABOVE.

DELETED MODULE #3(XLPAAA) BY SETTING SWITCHES 8 AND 9=10 AND SWITCHES 0-7=3 AND PRESSING CONTINUE

THEN DELETED MODULE #4(XRKAAA) USING THE ABOVE SEQUENCE I THEN DELETED MODULE #2(XCRAAA)

THEN REQUESTED ANOTHER CORE MAP. NOTE THAT THE STATUS WORD FOR MODULES 2,3,2ND 4 BIT #14 IS 0 INDICATING THAT THE MODULE HAS BEEN DESELECTED.

```
000001  XUCAAA AT 007504 STAT 040020
000002  XRKAAA AT 011102 STAT 100000
000003  XLPAAA AT 012162 STAT 100000
000004  XCRAAA AT 012716 STAT 100000
```

THEN SET ALL SWITCHES=0 AND PRESSED CONTINUE TO START THE MODULE RUNNING.

```
DECTAP DONE

UC15 DONE

XUCAAA PC 010720 APC 001214 ENDPAS 00001.

FP15T2 DONE
```

E

```
DXQAA-R  DEC/X11 USER MANUAL    MACY11,624  2-AUG-73  15:39  PAGE 44-1
XDOC1,P11        SYMBOL TABLE

.       = 000000R

          000000


? ERRORS DETECTED:  1


DXQAA-R  DEC/X11 USER MANUAL    MACY11,624  2-AUG-73  15:39  PAGE 44-2
XDOC1,P11

 *,XDOC1,PRT_XDOC1,P11
 RUN-TIME:  5  9  0 SECONDS
 CORE USED:  3K
```

```
                        .REM    -
```

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DXQAC-B-LA

PRODUCT NAME: DECX11 - UC15 MONITOR

DATE: JUNE 15,1973

MAINTAINER: DIAGNOSTIC GROUP

AUTHOR(S): R. KOLLER/B. CRISTOPHER


```
                        .LIST   SEQ
                        .TITLE  XQACB  UNIMON - DEC/X11 UC15 MONITOR
                        .ASECT
                        .LIST   ME
                        .NLIST  TOC,MC,CND

                        .GLOBL  LOCORE,HICORE,EABITS,WBUF,OACNV,BDCNV,MODQ
                ;SWITCH REGISTER OPTIONS
                ;SR15=1  HALT MODULE AFTER ERROR,
                ;SR14=1  INHIBIT MODULE HALT AFTER 20 ERRORS,
                ;SR13=1  INHIBIT ERROR PRINT,
                ;SR12=1  INHIBIT ENDPAS PRINTOUT,
                ;SR11=1 LOCK ON ERROR SEQUENCE
                ;****************START OF UNIMON SPECIAL CODE*********************
                ;SR10=  CTRLC (^C)
                ;SR8&9= 00 RUN
                ;       01 SEL (SEE SR7-0 BELOW)
                ;       10 DES (SEE SR7-0 BELOW)
                ;       11 MAP
                ;SR7-0= WILL SEL OR DES THE NUMBERED MODULE,
                ;       WHEN EQUAL TO 0, ALL MODULES WILL
                ;       BE SELECTED OR DESELECTED,
                ;****************END OF UNIMON SPECIAL CODE**********************

                        ;A FEW DEFINITIONS,
000000                  R0      =%0
000001                  R1      =%1
000002                  R2      =%2
000003                  R3      =%3
000004                  R4      =%4
000005                  R5      =%5
000006                  R6      =%6
000006                  SP      =%6
000007                  PC      =%7
177776                  PS      =177776
177776                  PSW=    177776
177570                  SR      =177570
000004                  PIRQ    =IOT
000000                  OPEN    =0
100000                  BIT15   =100000
040000                  BIT14   =40000
020000                  BIT13   =20000
010000                  BIT12   =10000
004000                  BIT11   =4000
002000                  BIT10   =2000
001000                  BIT9    =1000
000400                  BIT8    =400
000200                  BIT7    =200
000100                  BIT6    =100
000040                  BIT5    =40
000020                  BIT4    =20
000010                  BIT3    =10
000004                  BIT2    =4
```

```
   125          000002                  BIT1    =2
   126          000001                  BIT0    =1
   127          000340                  PRTY7   =340
   128          000300                  PRTY6   =300
   129          000240                  PRTY5   =240
   130          000200                  PRTY4   =200
   131          005746                  PUSH    =005746
   132          024646                  PUSH2   =024646
   133          005726                  POPSP   =005726
   134          022626                  POPSP2  =022626
   135          000100                  IE      =BIT6
   136          000040                  KBUFL   =32.
   138          000310                  IOQL    =200.
   139          000310                  TYPQL   =200.
   145          007754                  TYPLIM  =TYPEQ+TYPQL
   146          007444                  IOQLIM  =IOQ+IOQL
   147          005611                  ACRLF   =CTRLC+2
   148          000020                  STAT    =16.
   149          000021                  STAT1   =17.
   150          000022                  INIT    =18.
   151          000024                  SPOINT  =20.
   152          000026                  PSCNT   =22.
   153          000030                  ERCNT   =24.
   154          000032                  SVR0    =26.
   155          000034                  SVR1    =28.
   156          000036                  SVR2    =30.
   157          000040                  SVR3    =32.
   158          000042                  SVR4    =34.
   159          000044                  SVR5    =36.
   160          000046                  SVR6    =38.
   161          000050                  CSRA    =40.
   162          000052                  ACSR    =42.
   163          000052                  SBADR   =42.
   164          000054                  ASTAT   =44.
   165          000054                  WASADR  =44.
   166          000056                  ASB     =46.
   167          000060                  AWAS    =48.
   168                                  .MACR   TOKN    STRING,ADDRESS
   169                                  .ASCII  %'STRING'%
   170                                  .NLIST
   171                                  .BYTE   0
   172                                  .BYTE   .&1-1&1+3
   173                                  .EVEN
   174                                  .LIST
   175                                  .WORD   ADDRESS
   176                                  .ENDM
   177                                  .MACRO  TRPDEF  NAMEA,NAMEB
   178                                  NAMEB                   ;POINTER FOR TRAP CALL NAMEA
   179                                  .NLIST
   180                                  NAMEA=TRAP+TRAPX
   181                                  TRAPX=TRAPX+1
   182                                  .LIST
   183                                  .ENDM
   185                                  .MACRO  LINE1
```



```
   186                                  ;***************START OF UNIMON SPECIAL CODE*******************
   187                                          .ENDM
   188                                          .MACRO  LINE2
   189                                  ;***************END OF UNIMON SPECIAL CODE*********************
   190                                          .ENDM
```

```
                                .=0
                                .WORD   0,0
000000  000000  000000  BUSEV:  .+2,HALT
000004  000006  000000  RESIV:  .+2,HALT
000010  000012  000000  TRCV:   TRCI                    ;TRACE TRAP POINTER.
000014  003240                  0
000016  000000  IOTV:   PIRQ,                   ;
000020  003560                  PRTY7
000022  000340  PWRFV:  PWRDN                   ;POWER FAIL POINTER.
000024  005436                  PRTY7
000026  000340  EMTV:   .+2                     ;EMT POINTER
000030  000032                  HALT
000032  000000  TRPV:   TRPINT                  ;TRAP POINTER.
000034  005306                  0
000036  000000          .=40
000040  000040          ,WORD   OPEN            ;LOAD MEDIUM INDICATOR.
000040  000000  DDPPTR: .WORD   0               ;CHAIN MODE ONLY. POINTS TO DDPMON.
000042  000000          .WORD   0
000044  000000          ,WORD   0
000046  000000  LOCORE: OPEN                    ;ADDR OF FIRST FREE CORE LOCATION.
000050  000000  HICORE: OPEN                    ;CONTAINS ADDR OF HIGHEST BUFFER.
000052  000000  EABITS: OPEN                    ;CONTAINS EXTENDED ADDR BITS.
000054  000000  WBUF:   START                   ;CONTAINS CURRENT WRITE BUFFER ADDR.
000056  001136          ;FROM HERE THROUGH 776 FILLED WITH .+2 AND HALT.


                000200          .=200
000200  000167  000732          JMP     START           ;GO TO START OF MONITOR.

                001000          .=1000
001000  000167  000132          JMP     START           ;GO TO START OF MONITOR.
```



```
001004      000         IOQUE:  ,BYTE   OPEN            ;IOQUE AND TYPQUE MUST BE IN SAME WORD!!
001005      000         TYPQUE: ,BYTE   OPEN
001006      000         SPCFLG: ,BYTE   OPEN            ;SPCFLG AND DIRIND MUST BE IN SAME WORD!!
001007      000         DIRIND: ,BYTE   OPEN
001010      000         BKQUE:  ,BYTE   OPEN
001011      000         BRAKE:  ,BYTE   OPEN
001012      000         RMODE:  ,BYTE   OPEN
001013      000         TTYBSY: ,BYTE   OPEN            ;TELETYPE BUSY FLAG. 0= NOT BUSY.
001014      000         MODCNT: ,BYTE   OPEN
001015      000         MODCTR: ,BYTE   OPEN
001016      000         ERRIND: .BYTE   OPEN            ;0=ERROR, NOT0=DATA ERROR.
001017      000         FILCTR: ,BYTE   OPEN
                                ,EVEN
001020  000000          KBPTR:  OPEN
001022  000000          MODPTR: OPEN                    ;MODULE POINTER
001024  000000          ADDR:   OPEN
001026  000000          NUMBER: OPEN
001030  000000          DSTADR: OPEN
001032  000000          SRETRN: OPEN
001034  000000          IOBKID: OPEN
001036  000000          YES:    OPEN
001040  000000          TABADR: OPEN
001042  000000          RSTAT:  OPEN
001044  000000          TRCPC:  OPEN
001046  000000          TRCPSW: OPEN
001050  000000          IOQ1:   OPEN                    ;I/O QUEUE POINTERS.
001052  000000          IOQ2:   OPEN
001054  000000          TYPQ1:  OPEN                    ;TYPE QUEUE POINTERS
001056  000000          TYPQ2:  OPEN
001060  000000          MONR0:  OPEN                    ;MONITOR REGISTER SAVE AREA.
001062  000000          MONR1:  OPEN
001064  000000          MONR2:  OPEN
001066  000000          MONR3:  OPEN
001070  000000          MONR4:  OPEN
001072  000000          MONR5:  OPEN
001074  000000          SPSAV:  OPEN
001076  177560          TKS:    177560
001100  177562          TKB:    177562
001102  177564          TPS:    177564
001104  177566          TPB:    177566
001106  000024          ERRLIM: 20.
001110  000000          TTYBYT: OPEN
001112      014         FILCNT: ,BYTE   12.
001113      000         FILLER: ,BYTE   0
001114      000         SYSERI: ,BYTE   OPEN            ;SYSERI AND PWRFI MUST BE IN SAME WORD!!!
001115      000         PWRFI:  ,BYTE   OPEN
001116      000         CHN:    ,BYTE   OPEN
001117      000         ROTI:   ,BYTE   OPEN            ;ROTATE BUFFERS INDICATOR, 0= NO.
001120      000         FILLID: ,BYTE   OPEN            ;FILL INDICATOR.
                001122          ,EVEN
001122  167774  ;****************START OF UNIMON SPECIAL CODE********************
                DRIDB1: 167774          ;BITS 8&9= LOCAL MEM SIZ
                                        ;01=    4K LOCAL MEM
                                        ;10=    8K LOCAL MEM
```

```
                                                        ;11=    12K LOCAL MEM
001124  167760                 DRCSR2: 167760         ;BIT7= TCBP FLG
001126  167764                 DRIDB2: 167764         ;CONTAINS BITS 3-17 OF TCBP
                                ;***************END OF UNIMON SPECIAL CODE**********************
```



```
                                ;COMMON QUE CALL ROUTINE.
001130  104401                 COMQUE: QUE                    ;QUE CALL.
001132  000000                 CADDR:  OPEN                   ;DESTINATION ADDR.
001134  000000                 CSTART: OPEN                   ;MODULE START ADDR. (0 FOR MONITOR).

001136  000005                 START:  RESET                  ;CLEAR THE WORLD.
001140  012706  007134                 MOV     #SPBOT,R6     ;SET UP STACK.
001144  012767  005436  176652         MOV     #PWRDN,PWRFV  ;SET UP POWER FAIL VECTOR.
001152  005767  176672                 TST     LOCORE        ;DONE BUFFER SETUP?
001156  001002                         BNE     4$            ;BR IF YES.
001160  004767  005750                 JSR     PC,SETBUF     ;NO, DO IT.
                                ;***************START OF UNIMON SPECIAL CODE**********************
001164  004767  004002         4$:     JSR     PC,CLRQUS     ;CLEAR QUEUES.
001170  105777  177730         5$:     TSTB    @DRCSR2       ;TCBP FLG SET?
001174  100375                         BPL     5$            ;BR IF NO.
001176  017701  177724                 MOV     @DRIDB2,R1    ;SYNC UP.
001202  105777  177716         6$:     TSTB    @DRCSR2       ;FLG SET?
001206  100375                         BPL     6$            ;BR IF NOT. (WAIT FOR IT).
001210  017701  177706                 MOV     @DRIDB1,R1    ;DETERMINE
001214  006101                         ROL     R1            ;EXACT
001216  006101                         ROL     R1            ;AMOUNT
001220  006101                         ROL     R1            ;OF
001222  006101                         ROL     R1            ;LOCAL
001224  006101                         ROL     R1
001226  042701  117777                 BIC     #117777,R1    ;MEMORY
001232  067701  177670                 ADD     @DRIDB2,R1    ;ADDR OF WBUF FOR MODULES.
001236  010167  176614                 MOV     R1,WBUF       ;
001242  162701  000002                 SUB     #2,R1         ;LOC TPB CONTAINS ADDR OF TTY BUFF +0.
001246  010167  177632                 MOV     R1,TPB        ;
001252  104406  005542                 MSG,TITLE             ;TYPE TITLE.
001256  104406  005566                 MSG,DOT               ;TYPE DOT.
001262  005767  176302                 TST     SR            ;SR = 0?
001266  001002                         BNE     INPUT         ;BR IF NOT.
001270  000167  002674                 JMP     RUN           ;YES, START RUN MODE.

001274  012706  007134         INPUT:  MOV     #SPBOT,R6     ;RESET STACK.
001300  005067  177610                 CLR     SYSERI        ;CLEAR SYSERI AND PWRFI INDICATORS.
001304  000000                         HALT                  ;
001306  016700  176256                 MOV     SR,R0         ;
001312  042700  176377                 BIC     #176377,R0    ;
001316  005700                         TST     R0            ;RUN SELECTED?
001320  001002                         BNE     1$            ;BR IF NO.
001322  000137  004170                 JMP     @#RUN         ;
001326  022700  000400         1$:     CMP     #400,R0       ;SEL SELECTED?
001332  001002                         BNE     2$            ;BR IF NO.
001334  000137  004512                 JMP     @#SEL         ;
001340  022700  001000         2$:     CMP     #1000,R0      ;DES SELECTED?
001344  001002                         BNE     3$            ;BR IF NO.
001346  000137  004530                 JMP     @#DES         ;
001352  000137  004326         3$:     JMP     @#MAP         ;MAP SELECTED.
001356  005067  176414         QUETST: CLR     PSW           ;CLEAR STATUS.
001362  105767  177416                 TSTB    IOQUE         ;IO QUE REQUEST PENDING?
                                ;***************END OF UNIMON SPECIAL CODE**********************
001366  001112                         BNE     IOQ8VC        ;BR IF YES.
```

```
001370  005767  177440          TST     IOBKID          ;STARTING I/O MODULES?
001374  100415                  BMI     QTSTC           ;BR IF YES.
001376  105767  177403          TSTB    TYPQUE          ;TYPE REQUEST PENDING?
001402  001405                  BEQ     QTSTB           ;BR IF NOT.
001404  105767  177403          TSTB    TTYBSY          ;TTY BUSY?
001410  001002                  BNE     QTSTB           ;BR IF YES.
001412  000167  000414          JMP     TYPSVC          ;NO, GO SERVICE TYPE QUEUE.
                        ;***************START OF UNIMON SPECIAL CODE*******************
001416  004767  003504  QTSTB: JSR     PC,SR10CK       ;CHECK FOR SR10 (CTRLC).
001422  005767  177416          TST     TRCPC           ;BACKGROUND MODULE PENDING?
001426  001166                  BNE     BKQSVC          ;BR IF YES.
001430  105767  177356  QTSTC: TSTB    RMODE           ;IN RUN MODE?
001434  001750                  BEQ     QUETST          ;BR IF NOT.
                        ;***************END OF UNIMON SPECIAL CODE*********************
```



```
                        ;RUN MODE SERVICE ROUTINE.
001436  105767  177347  RUNSVC: TSTB    BRAKE           ;IS THE BRAKE ON?
001442  001345                  BNE     QUETST          ;BR IF YES, DO NOT INIT MORE MODULES.
001444  105767  177345          TSTB    MODCTR          ;MODCTR =0?
001450  001003                  BNE     4$              ;BR IF NOT.
001452  105767  177440          TSTB    CHN             ;YES, ARE WE IN CHAIN MODE?
001456  001337                  BNE     QUETST          ;BR IF YES, DO NO MORE.
001460  010046          4$:     MOV     R0,-(6)         ;SAVE R0.
001462  062767  000002  177332  ADD     #2,MODPTR       ;POINT TO NEXT MODULE.
001470  017700  177326          MOV     @MODPTR,R0      ;MODULE ADDR TO R0.
001474  001427                  BEQ     1$              ;BR IF NO ADDR.
001476  026067  000020  177330  CMP     STAT(0),IOBKID  ;CORRECT MODULE TO RUN?
001504  001041                  BNE     3$
001506  005760  000020          TST     STAT(0)         ;BACKGROUND MODULE?
001512  100403                  BMI     2$              ;BR IF NOT.
001514  112767  177777  177267  MOVB    #-1,BRAKE       ;YES, APPLY BRAKE.
001522  016060  000024  000046  2$:     MOV     SPOINT(0),SVR6(0) ;SET UP MODULE SP POINTER.
001530  016067  000022  177374  MOV     INIT(0),CADDR   ;SET UP DESTINATION ADDR.
001536  010067  177372          MOV     R0,CSTART       ;SET UP MODULE START ADDR.
001542  012600                  MOV     (6)+,R0         ;RESTORE R0.
001544  105367  177245          DECB    MODCTR          ;DECR COUNT OF MODS INITED.
001550  000167  177354          JMP     COMQUE          ;GO TO COMMON QUE CALL.
001554  022767  040000  177252  1$:     CMP     #40000,IOBKID   ;IS IT NON TRACE BACKGROUND?
001562  001004                  BNE     5$              ;BR IF NOT.
001564  012767  140000  177242  MOV     #140000,IOBKID  ;YES, SWITCH TO IOMOD.
001572  000403                  BR      6$
001574  012767  040020  177232  5$:     MOV     #40020,IOBKID   ;SWITCH TO BACKGROUND MODE.
001602  012767  006422  177212  6$:     MOV     #MODQ+2,MODPTR  ;POINT TO MODULE TABLE START.
001610  012600                  3$:     MOV     (6)+,R0         ;RESTORE R0.
001612  000661                  BR      QUETST
```

```
                                        ;I/O QUE SERVICE ROUTINE.
001614  026727  177232  007444  IOQSVC: CMP     IOQ2,#IOQLIM    ;REACHED LIMIT OF QUEUE?
001622  103403                          BLO     1$              ;BR IF NOT.
001624  012767  007134  177220          MOV     #IOQ,IOQ2       ;RESET IOQ2.
001632  012767  000340  176136  1$:     MOV     #PRTY7,PSW      ;SET PRIORITY 7.
001640  017700  177206                  MOV     @IOQ2,R0        ;GET PC.
001644  062767  000002  177200          ADD     #2,IOQ2         ;UPDATE IOQ2.
001652  105367  177126                  DECB    IOQUE           ;DECREMENT REQUEST COUNT.
001656  005067  176114                  CLR     PSW             ;CLEAR STATUS.
001662  012067  177142                  MOV     (0)+,DSTADR     ;GET DESTINATION ADDR.
001666  012000                          MOV     (0)+,R0         ;GET MODULE ADDR. IS IT 0?
001670  001002                          BNE     2$              ;BR IF NOT. IT'S A MODULE.
001672  000177  177132                  JMP     @DSTADR         ;GO DO MONITOR FUNCTION.

001676  116067  000020  177136  2$:     MOVB    STAT(0),RSTAT   ;GET RUN STATUS.
001704  032760  020000  000020  IOQSVA: BIT     #BIT13,STAT(0)  ;MODULE STOPPED?
001712  001221                          BNE     QUETST          ;BR IF YES. FORGET IT!
001714  010067  177140                  MOV     R0,MONR0        ;SAVE R0. (MODULE ADDR).
001720  010167  177136                  MOV     R1,MONR1        ;SAVE MONITOR REGS.
001724  010267  177134                  MOV     R2,MONR2
001730  010367  177132                  MOV     R3,MONR3
001734  010467  177130                  MOV     R4,MONR4
001740  010567  177126                  MOV     R5,MONR5
001744  010667  177124                  MOV     R6,SPSAV        ;SAVE MONITOR STACK TOO!.
001750  062700  000050                  ADD     #SVR6+2,R0      ;RESTORE MODULE'S REGS,
001754  014006                          MOV     -(0),R6         ;STARTING WITH STACK POINTER.
001756  014005                          MOV     -(0),R5
001760  014004                          MOV     -(0),R4
001762  014003                          MOV     -(0),R3
001764  014002                          MOV     -(0),R2
001766  014001                          MOV     -(0),R1
001770  014000                          MOV     -(0),R0
001772  016746  177044          IOQSVB: MOV     RSTAT,-(6)      ;LOAD RUN STATUS.
001776  016746  177026                  MOV     DSTADR,-(6)     ;LOAD DESTINATION ADDR.
002002  000006                  RTT1:   RTT                     ;GO TO DESTINATION.

                                        ;BACKGROUND QUEUE SERVICED HERE.
002004  016767  177034  177016  BKQSVC: MOV     TRCPC,DSTADR    ;SET UP DESTINATION ADDR.
002012  116767  177030  177022          MOVB    TRCPSW,RSTAT    ;SET UP RUN STATUS.
002020  017700  176776                  MOV     @MODPTR,R0      ;MODULE START ADDR TO R0.
002024  005067  177014                  CLR     TRCPC           ;CLEAR BK MODULE WAITING INDICATOR.
002030  000725                          BR      IOQSVA          ;GO GET GOING.
```



```
                                        ;TYPE QUE SERVICE ROUTINE.
002032  026727  177020  007754  TYPSVC: CMP     TYPQ2,#TYPLIM ;REACHED UPPER END OF QUEUE?
002040  103403                          BLO     1$              ;BR IF NOT.
002042  012767  007444  177006          MOV     #TYPEQ,TYPQ2    ;YES, RESET TYPQ2.
002050  105267  176737          1$:     INCB    TTYBSY          ;INDICATE TTY BUSY.
002054  017701  176776                  MOV     @TYPQ2,R1       ;GET PC OF CALL.
002060  062767  000002  176770          ADD     #2,TYPQ2        ;UPDATE TYPQ2.
002066  105367  176713                  DECB    TYPQUE          ;DECREMENT REQUEST COUNT.
002072  010146                          MOV     R1,-(6)         ;SAVE R1.
002074  005741                          TST     -(1)            ;POINT TO CALL.
002076  010167  176724                  MOV     R1,NUMBER       ;SAVE IT FOR LATER.
002102  011101                          MOV     (1),R1          ;GET CALL
002104  006301                          ASL     R1              ;TIMES 2.
002106  016101  171110                  MOV     TYPTAB-TRP2-4(1),R1     ;FORM SERVICE ADDR.
002112  000201                          RTS     R1              ;GO TO IT, RESTORE R1.
002114  003114  003016  002312  TYPTAB: .WORD   PASEND,ENDSVC,ERRSVC,ERSVC1
002122  002304
002124  002222  002256  002274          .WORD   MSG.,BREAK.,ERSVC2,MSGN.
002132  002134

                                        ;MSGN. ROUTINE. SERVICES CALLS TO TYPE ASCII MESSAGES.
002134  012167  176700          MSGN.:  MOV     (1)+,TABADR     ;GET ASCII TABLE ADDR.
002140  012167  176770                  MOV     (1)+,CSTART     ;MODULE ADDR TO CSTART.
002144  010167  176762                  MOV     R1,CADDR        ;RESUME ADDR TO CADDR.
002150  016701  176760                  MOV     CSTART,R1       ;MODULE ADDR TO R1.
002154  004767  000540                  JSR     PC,ENDCOM       ;DO COMMON STUFF.
002160  032767  020000  175402          BIT     #BIT13,SR       ;INHIBIT ERROR PRINT?
002166  001024                          BNE     MSG1.           ;BR IF YES.
002170  012746  006123                  MOV     #AEND,-(6)      ;TYPE COMMON HEADER.
002174  004767  001570                  JSR     PC,TYPE
002200  016701  176634                  MOV     TABADR,R1       ;TABLE ADDR TO R1.
002204  012146                  1$:     MOV     (1)+,-(6)       ;GET MESSAGE ADDR.
002206  022716  177777                  CMP     #-1,(6)         ;TERMINATOR?
002212  001412                          BEQ     MSG1.           ;BR IF YES. DONE.
002214  004767  001550                  JSR     PC,TYPE         ;NO. TYPE MESSAGE.
002220  000771                          BR      1$              ;GO DO IT AGAIN.

                                        ;MSG CALL SERVICED HERE.
002222  012146                  MSG.:   MOV     (1)+,-(6)       ;ASCII MESSAGE ADDR TO STACK.
002224  010167  176702                  MOV     R1,CADDR        ;RESUME ADDR TO CADDR.
002230  005067  176700                  CLR     CSTART          ;INDICATE MONITOR QUE CALLING.
002234  004767  001530                  JSR     PC,TYPE         ;GO TYPE DESIRED MESSAGE.
002240  105067  176547          MSG1.:  CLRB    TTYBSY
                                !***************START OF UNIMON SPECIAL CODE*******************
002244  112777  000377  176632          MOVB    #377,@TPB
                                !***************END OF UNIMON SPECIAL CODE*********************
002252  000167  176652                  JMP     COMQUE          ;GO QUEUE UP TO RESUME.

                                        ;BREAK ROUTINE. SERVICES BREAK CALL.
002256  012167  176652          BREAK.: MOV     (1)+,CSTART     ;GET MODULE ADDR.
002262  010167  176644                  MOV     R1,CADDR        ;GET DESTINATION ADDR.
                                !***************START OF UNIMON SPECIAL CODE*******************
002266  004767  002634                  JSR     PC,SR10CK       ;CHECK FOR ^C.
                                !***************END OF UNIMON SPECIAL CODE*********************
002272  000762                          BR      MSG1.
```

```
                                        ;ERROP CALLS ARE SERVICED HERE.
002274  112767  000002  176514  ERSVC2: MOVB    #2,ERRIND       ;INDICATE ERRORN CALL.
002302  000406                  BR      ERSVCA
002304  105067  176506          ERSVC1: CLRB    ERRIND          ;INDICATE DATA ERROR.
002310  000403                  BR      ERSVCA
002312  112767  000001  176476  ERRSVC: MOVB    #1,ERRIND       ;INDICATE "NORMAL" ERROR.
002320  012167  176610          ERSVCA: MOV     (1)+,CSTART     ;SAVE START ADDR OF MODULE.
002324  122767  000002  176464          CMPB    #2,ERRIND       ;ERRORN CALL?
002332  001002                  BNE     1$              ;BR IF NOT.
002334  012167  176500          MOV     (1)+,TABADR     ;SAVE TABLE ADDR.
002340  010167  176566          1$:     MOV     R1,CADDR        ;RESUME ADDR TO PSENDB.
002344  016701  176564          MOV     CSTART,R1       ;GET BACK START ADDR.
002350  004767  000344          JSR     PC,ENDCOM       ;DO COMMON STUFF.
002354  005261  000030          INC     ERCNT(1)        ;INCREMENT MODULE'S ERROR COUNT.
002360  001002                  BNE     ERSVCB          ;BR IF RESULT NOT 0.
002362  005161  000030          COM     ERCNT(1)        ;RESET COUNT TO -1.
002366  012702  000005  ERSVCB: MOV     #5,R2           ;GET TYPE DATA FROM QUEUE TO STACK.
002372  004767  000356  ERSVCC: JSR     PC,TYPDAT       ;DO IT.
002376  005302                  DEC     R2              ;DONE?
002400  001374                  BNE     ERSVCC          ;BR IF NOT.
002402  016146  000030          MOV     ERCNT(1),-(6)   ;ERROR COUNT TO STACK.
002406  004567  002402          JSR     R5,BDCNV        ;CONVERT ERROR COUNT TO DECIMAL.
002412  006406                  AERNMB
002414  105767  176376          TSTB    ERRIND          ;DATA ERROR?
002420  001020                  BNE     7$              ;BR IF NOT.
002422  004567  002312          JSR     R5,OACNV        ;CONVERT WAS TO OCTAL.
002426  006321                  ADTE2
002430  004567  002304          JSR     R5,OACNV        ;CONVERT S/B TO OCTAL.
002434  006306                  ADTE3
002436  004567  002276          JSR     R5,OACNV        ;CONVERT WASADR TO OCTAL.
002442  006354                  ADTE4
002444  004567  002270          JSR     R5,OACNV        ;CONVERT SBADR TO OCTAL.
002450  006336                  ADTE5
002452  004567  002262          JSR     R5,OACNV        ;CONVERT CSR ADDR TO OCTAL.
002456  006273                  ADTE6
002460  000412                  BR      6$
002462  022626          7$:     POPSP2                  ;SKIP WAS AND S/B.
002464  004567  002250          JSR     R5,OACNV        ;CONVERT STAT REG CONTENTS TO OCTAL.
002470  006254                  ASTATC
002472  004567  002242          JSR     R5,OACNV        ;CONVERT CSR CONTENTS TO OCTAL.
002476  006237                  ACSRC
002500  004567  002234          JSR     R5,OACNV        ;CONVERT CSR ADDR TO OCTAL.
002504  006223                  ACSRAC
002506  032767  020000  175054  6$:     BIT     #BIT13,SR       ;INHIBIT ERROR PRINT?
002514  001054                  BNE     1$              ;BR IF YES.
002516  012746  006123          MOV     #AEND,-(6)      ;TYPE COMMON HEADER.
002522  004767  001242          JSR     PC,TYPE
002526  012746  006400          MOV     #ERRNMB,-(6)    ;TYPE ERROR NUMBER.
002532  004767  001232          JSR     PC,TYPE
002536  105767  176254          TSTB    ERRIND          ;DATA ERROR?
002542  001003                  BNE     4$              ;BR IF NOT.
002544  012746  006265          MOV     #ADTERR,-(6)    ;TYPE DATA ERROR MESSAGE.
002550  000402                  BR      5$
002552  012746  006215          4$:     MOV     #AERROR,-(6)    ;TYPE ERROR MESSAGE.
```



```
002556  004767  001206          5$:     JSR     PC,TYPE
002562  122767  000002  176226          CMPB    #2,ERRIND       ;ERRORN CALL?
002570  001026                  BNE     1$              ;BR IF NOT.
002572  016702  176242          MOV     TABADR,R2       ;TABLE ADDR TO R2.
002576  012703  000010          8$:     MOV     #8.,R3          ;WILL TYPE 8 VALUES PER LINE.
002602  022712  177777          9$:     CMP     #-1,(2)         ;TERMINATOR?
002606  001417                  BEQ     1$              ;BR IF YES, DONE.
002610  013246                  MOV     @(2)+,-(6)      ;PUT VALUE IN STACK.
002612  004567  002122          JSR     R5,OACNV        ;CONVERT IT TO OCTAL.
002616  006111                  AOCTAL
002620  012746  006111          MOV     #AOCTAL,-(6)    ;TYPE IT.
002624  004767  001140          JSR     PC,TYPE
002630  005303                  DEC     R3              ;DONE 6 PER LINE?
002632  001363                  BNE     9$              ;BR IF NOT.
002634  012746  005611          MOV     #ACRLF,-(6)     ;OUTPUT CRLF.
002640  004767  001124          JSR     PC,TYPE
002644  000754                  BR      8$              ;GO FOR MORE.
002646  012746  005611  1$:     MOV     #ACRLF,-(6)     ;OUTPUT CRLF.
002652  004767  001112          JSR     PC,TYPE
002656  105067  176131          CLRB    TTYBSY          ;CLEAR TTY BUSY INDICATOR.
                        ;***************START OF UNIMON SPECIAL CODE*********************
002662  112777  000377  176214          MOVB    #377,@TPB
                        ;***************END OF UNIMON SPECIAL CODE***********************
002670  005767  174674          TST     SR              ;HALT MODULE ON ERROR?
002674  100410                  BMI     2$              ;BR IF YES.
002676  026761  176204  000030          CMP     ERRLIM,ERCNT(1) ;ERROR COUNT 20 OR GREATER?
002704  003153                  BGT     PSENDB          ;BR IF NOT, CONTINUE MODULE EXECUTION.
002706  032767  040000  174654          BIT     #BIT14,SR       ;YES, HALT MODULE AFTER 20 ERRORS?
002714  001147                  BNE     PSENDB          ;BR IF NOT, GO QUE MODULE TO RESUME.
002716  000442          2$:     BR      ENDSVA          ;YES, GO HALT MODULE.

002720  016746  176102  ENDCOM: MOV     NUMBER,-(6)     ;SAVE PC OF CALL.
002724  011646                  MOV     (6),-(6)        ;SAVE IT AGAIN.
002726  160116                  SUB     R1,(6)          ;COMPUTE ASSEMBLY PC.
002730  004567  002004          JSR     R5,OACNV        ;CONVERT ASSEMBLY PC TO ASCII.
002734  006153                  AEND2
002736  004567  001776          JSR     R5,OACNV        ;CONVERT PC TO ASCII.
002742  006137                  AEND1
002744  004567  001662          JSR     R5,FILLNM       ;LET'S GET MODULE NAME.
002750  006124                  AEND+1
002752  000207                  RTS     PC              ;LET'S GET OUT.

                        ;TYPDAT SUB. LOADS QUEUED DATA ONTO STACK.
002754  011646  TYPDAT: MOV     (6),-(6)        ;SAVE EXIT ON STACK AGAIN.
002756  026727  176074  007754          CMP     TYPQ2,#TYPLIM   ;REACHED END OF QUEUE?
002764  103403                  BLO     1$              ;BR IF NOT.
002766  012767  007444  176062          MOV     #TYPEQ,TYPQ2    ;YES, POINT TO START OF QUEUE.
002774  017766  176056  000002  1$:     MOV     @TYPQ2,2(6)     ;QUEUE DATA TO STACK.
003002  062767  000002  176046          ADD     #2,TYPQ2        ;UPDATE QUEUE POINTER.
003010  105367  175771          DECB    TYPQUE          ;DECREMENT COUNT.
003014  000207                  RTS     PC              ;EXIT, LEAVE DATA IN STACK.
```

```
                                ;END CALL SERVICED HERE.
003016  011101                  ENDSVC: MOV     (1),R1          ;GET START ADDR.
003020  004767  177674                  JSR     PC,ENDCOM       ;DO COMMON STUFF.
003024  052761  020000  000020  ENDSVA: BIS     #BIT13,STAT(1)  ;SET STOP BIT IN MODULE STAT.
003032  012746  006123                  MOV     #AEND,-(6)      ;TYPE COMMON HEADER.
003036  004767  000726                  JSR     PC,TYPE
003042  012746  006163                  MOV     #MODEND,-(6,    ;TYPE END MESSAGE.
003046  004767  000716                  JSR     PC,TYPE
003052  105067  175735          ENDSVB: CLRB    TTYBSY          ;CLEAR TTY BUSY INDICATOR.
                                ;***************START OF UNIMON SPECIAL CODE*********************
003056  112777  000377  176020          MOVB    #377,@TPB
                                ;***************END OF UNIMON SPECIAL CODE***********************
003064  005761  000020          ENDSVE: TST     STAT(1)         ;BACKGROUND MODULE?
003070  100402                          BMI     1$              ;BR IF NOT.
003072  105067  175713                  CLRB    BRAKE           ;RELEASE BRAKE.
003076  105367  175712          1$:     DECB    MODCNT          ;DECR COUNT OF MODULES RUNNING.
003102  001002                          BNE     ENDSVD          ;BR IF COUNT NOT 0.
003104  000167  000546                  JMP     CTRLCB          ;COUNT 0, TERMINATE RUN MODE.
003110  000167  176242          ENDSVD: JMP     QUETST          ;GO BACK TO SERVICE QUEUES.


                                ;PASEND ROUTINE. TYPES END OF PASS MESSAGE.
                                ;BACKGROUND MODULES ARE NOT ALLOWED TO MAKE MULTIPLE PASSES.
                                ;IN CHAIN MODE NO MODULE ALLOWED TO MAKE MULTIPLE PASSES.
003114  012167  176012          PASEND: MOV     (1)+,CADDR      ;GET RESUME ADDR.
003120  012101                          MOV     (1)+,R1         ;GET MODULE START ADDR.
003122  010167  176006                  MOV     R1,CSTART       ;SAVE IT FOR LATER QUE CALL.
003126  004767  177566                  JSR     PC,ENDCOM       ;DO COMMON STUFF.
003132  005261  000026                  INC     PSCNT(1)        ;INCREMENT PASS COUNT.
003136  016146  000026                  MOV     PSCNT(1),-(6)   ;NOW GET IT.
003142  004567  001646                  JSR     R5,BDCNV        ;CONVERT IT TO DECIMAL ASCII.
003146  006205                          BPSCNT                  ;STUFF IT AT BPSCNT.
003150  032767  010000  174412          BIT     #BIT12,SR       ;INHIBIT ENDPAS PRINTOUT?
003156  001010                          BNE     PSENDA          ;BR IF YES.
003160  012746  006123                  MOV     #AEND,-(6)      ;TYPE COMMON HEADER.
003164  004767  000600                  JSR     PC,TYPE
003170  012746  006175                  MOV     #APSEND,-(6)    ;TYPE ENDPAS AND PAS COUNT TOO.
003174  004767  000570                  JSR     PC,TYPE
003200  105067  175607          PSENDA: CLRB    TTYBSY          ;CLEAR TTY BUSY INDICATOR.
                                ;***************START OF UNIMON SPECIAL CODE*********************
003204  112777  000377  175672          MOVB    #377,@TPB
                                ;***************END OF UNIMON SPECIAL CODE***********************
003212  105767  175700                  TSTB    CHN             ;IN CHAIN MODE?
003216  001322                          BNE     ENDSVE          ;BR IF YES TO END MODULE EXECUTION.
003220  005761  000020                  TST     STAT(1)         ;BACKGROUND MODULE?
003224  100403                          BMI     PSENDB          ;BR IF NOT.
003226  105067  175557                  CLRB    BRAKE           ;RELEASE BRAKE.
003232  000726                          BR      ENDSVD          ;IGNORE ENDPAS POINTER.
003234  000167  175670          PSENDB: JMP     COMQUE          ;GO TO COMMON QUE CALL.
```



```
                                ;TRACE TRAP ENTERS HERE.
003240  005767  175540          TRCI:   TST     IOQUE           ;I/O OR TYPE QUE WAITING?
003244  001001                          BNE     TRCIB           ;BR IF YES.
003246  000006                  TRCIA:  RTT                     ;NO, EXIT, (RTT?).
                                ;***************START OF UNIMON SPECIAL CODE*********************
003250  004767  001652          TRCIB:  JSR     PC,SR10CK       ;CHECK FOR CTRL C.
003254  012667  175564                  MOV     (6)+,TRCPC      ;SAVE MODS PC.
                                ;***************END OF UNIMON SPECIAL CODE***********************
003260  012667  175562                  MOV     (6)+,TRCPSW     ;SAVE MOD'S PSW.
003264  000401                          BR      EXIT1.


                                ;EXIT CALL ENTERS HERE.
003266  022626                  EXIT.:  POPSP2
003270  010046                  EXIT1.: MOV     R0,-(6)         ;SAVE R0 IN STACK.
003272  016700  175562                  MOV     MONR0,R0        ;MODULE ADDR IN R0.
003276  062700  000032                  ADD     #SVR0,R0        ;POINT TO MOD'S REG SAVE AREA.
003302  012620                          MOV     (6)+,(0)+       ;SAVE R0, (FROM STACK).
003304  010120                          MOV     R1,(0)+         ;SAVE REMAINING REGS.
003306  010220                          MOV     R2,(0)+
003310  010320                          MOV     R3,(0)+
003312  010420                          MOV     R4,(0)+
003314  010520                          MOV     R5,(0)+
003316  010610                          MOV     R6,(0)
003320  016706  175550                  MOV     SPSAV,R6        ;RESTORE MONITOR STACK.
003324  016701  175532                  MOV     MONR1,R1        ;RESTORE MONITOR REGS.
003330  016702  175530                  MOV     MONR2,R2
003334  016703  175526                  MOV     MONR3,R3
003340  016704  175524                  MOV     MONR4,R4
003344  016705  175522                  MOV     MONR5,R5
003350  005067  174422          EXIT2.: CLR     PSW             ;CLEAR STATUS.
003354  000240                          NOP
003356  000167  175774                  JMP     QUETST


                                ;TYPQ2, ROUTINE. SERVICES ERROR AND DATA ERROR CALLS.
003362  011646                  TYPQ2.: MOV     (6),-(6)        ;SAVE PC OF CALL AGAIN.
003364  004767  000074                  JSR     PC,LDTYPQ       ;QUEUE UP CALL.
003370  010046                          MOV     R0,-(6)         ;SAVE R0.
003372  017600  000002                  MOV     @2(6),R0        ;GET MODULE ADDR.
003376  062700  000062                  ADD     #AWAS+2,R0      ;
003402  014046                          MOV     -(0),-(6)       ;GET AWAS.
003404  005010                          CLR     (0)             ;CLEAR IT.
003406  014046                          MOV     -(0),-(6)       ;GET ASB.
003410  005010                          CLR     (0)             ;CLEAR IT.
003412  014046                          MOV     -(0),-(6)       ;GET ASTAT.
003414  005010                          CLR     (0)             ;CLEAR IT.
003416  014046                          MOV     -(0),-(6)       ;GET ACSR.
003420  005010                          CLR     (0)             ;CLEAR IT.
003422  014046                          MOV     -(0),-(6)       ;GET CSRA.
003424  012700  000005                  MOV     #5,R0           ;LOAD TYPE DATA ONTO QUEUE.
003430  004767  000030          1$:     JSR     PC,LDTYPQ       ;DO IT.
003434  005300                          DEC     R0              ;DONE?
003436  001374                          BNE     1$              ;BR IF NOT.
003440  012600                          MOV     (6)+,R0         ;RESTORE R0.
003442  000711                          BR      EXIT.
```

```
  761
  762  003444  004767  000014          ;TYPQ, ROUTINE, SERVICES END AND ENDPAS CALLS,
  763  003450  005726                  TYPQ.:  JSR     PC,LDTYPQ       ;QUEUE UP CALL,
  764  003452  000706                          POPSP
  765                                          BR      EXIT1,
  766
  767  003454  004767  000004          ;TYPQ1, ROUTINE, SERVICES MSG CALL,
  768  003460  005726                  TYPQ1.: JSR     PC,LDTYPQ       ;QUEUE UP CALL,
  769  003462  000732                          POPSP
  770                                          BR      EXIT2,
  771  003464  005067  174306          LDTYPQ: CLR     PSW             ;CLEAR STATUS,
  772  003470  105767  175311                  TSTB    TYPQUE          ;REQUEST COUNT 0?
  773  003474  001406                          BEQ     1$              ;BR IF YES,
  774  003476  026767  175352  175352          CMP     TYPQ1,TYPQ2     ;NO, TYPQ1 AND TYPQ2 SAME?
  775  003504  001002                          BNE     1$              ;BR IF NOT,
  776  003506  005767  174267                  TST     1               ;YES, QUEUE OFLO, CRASH SYSTEM BY REF
  777                                                                  ;TO ODD ADDRESS,
  778  003512  026727  175336  007754  1$:     CMP     TYPQ1,#TYPLIM   ;REACHED HIGH LIMIT?
  779  003520  001003                          BNE     2$              ;BR IF NOT,
  780  003522  012767  007444  175324          MOV     #TYPEQ,TYPQ1    ;RESET TYPQ1,
  781  003530  016677  000002  175316  2$:     MOV     2(6),@TYPQ1     ;STORE PC OF PENDING CALL,
  782  003536  105267  175243                  INCB    TYPQUE          ;UPDATE REQUEST COUNTS,
  783  003542  062767  000002  175304          ADD     #2,TYPQ1        ;UPDATE TYPQ1,
  784  003550  012616                          MOV     (6)+,(6)        ;
  785  003552  000207                          RTS     PC              ;EXIT,
  786
  787                                  ;QUE, ROUTINE, SERVICES QUE CALL,
  788  003554  005066  000002          QUE.:   CLR     2(6)            ;INDICATE QUE CALL,
  789
  790                                  ;PIRQ ROUTINE HANDLES PIRQ CALLS,
  791  003560                          PIRQ.:
  792
  793                                  ;LDIOQ ROUTINE,
  794  003560  012767  000340  174210  LDIOQ:  MOV     #PRTY7,PSW      ;ASSUME PRIORITY 7,
  795  003566  105767  175212                  TSTB    IOQUE           ;REQUEST COUNT 0?
  796  003572  001406                          BEQ     1$              ;BR IF YES,
  797  003574  026767  175250  175250          CMP     IOQ1,IOQ2       ;IOQ1 AND IOQ2 SAME?
  798  003602  001002                          BNE     1$              ;BR IF NOT,
  799  003604  005767  174171                  TST     1               ;QUE OFLO, CRASH SYSTEM BY REF TO
  800                                                                  ;ODD ADDRESS,
  801  003610  026727  175234  007444  1$:     CMP     IOQ1,#IOQLIM    ;REACHED HIGH LIMIT?
  802  003616  001003                          BNE     2$              ;BR IF NOT,
  803  003620  012767  007134  175222          MOV     #IOQ,IOQ1       ;RESET IOQ1,
  804  003626  012677  175216          2$:     MOV     (6)+,@IOQ1      ;STORE PC OF PENDING CALL,
  805  003632  105267  175146                  INCB    IOQUE           ;UPDATE REQUEST COUNTS,
  806  003636  062767  000002  175204          ADD     #2,IOQ1         ;UPDATE IOQ1,
  807  003644  005726                          TST     (6)+            ;CHECK FOR QUE CALL,
  808  003646  001640                          BEQ     EXIT2,
  809  003650  000002                          RTI                     ;EXIT INTERRUPT,
```



```
  876
  877
  878  003652  005726                  ;SERVICE CTRL C, ENDS RUN MODE ALSO,
  879  003654  022626                  CTRLCA: POPSP
  880  003656  005067  175152                  POPSP2                  ;REMOVE INTERRUPT FROM STACK,
  881  003662  105767  175124          CTRLCB: CLR     IOBKID          ;CLEAR MODULE TYPE INDICATOR,
  882  003666  001004                          TSTB    RMODE           ;IN RUN MODE?
  883  003670  004767  000032                  BNE     1$              ;BR IF YES,
  884  003674  000167  000246                  JSR     PC,CTRLX        ;CLEAR QUEUES,TYPE ^C
  885  003700  004767  000022                  JMP     COMCO3          ;BACK TO KYBD ROUTINE,
  886  003704  105767  175206          1$:     JSR     PC,CTRLX        ;CLEAR QUEUES,TYPE ^C
  887  003710  001004                          TSTB    CHN             ;IN CHAIN MODE?
  888  003712  104406  005571                  BNE     CTRLCD          ;BR IF YES, BYPASS SUMMARY,
  889  003716  004767  000422          CTRLCC: MSG,SUMARY              ;TYPE RUN END SUMMARY TITLE,
  890  003722  000167  001562                  JSR     PC,DIRA         ;TYPE RUN SUMMARY,
  891  003726  012767  000340  174042  CTRLCD: JMP     CHNOUT          ;EXIT, OR RETURN TO KYBD RTN,
  892  003734  012767  000062  175172  CTRLX:  MOV     #PRTY7,PSW      ;ASSUME STATUS 7,
  893  003742  004767  001224                  MOV     #50.,CSTART     ;CLEAR QUEUES AND DELAY TOO,
  894  003746  005367  175162          1$:     JSR     PC,CLRQUS       ;CLEAR QUEUES,
  895  003752  001373                          DEC     CSTART          ;DONE?
  896  003754  000005                          BNE     1$              ;BR IF NOT,
  897  003756  005067  174014                  RESET
  898  003762  104406  005607                  CLR     PSW             ;ASSUME STATUS 0,
  902  003766  000207                          MSG,CTRLC               ;OUTPUT ^C
  903                                          RTS     PC              ;EXIT,
```

```
                                        ;TYPE SUBROUTINE.
003770  112767  000001  175021  TYPE:   MOVB    #1,FILCTR       ;SET FILCTR TO 1.
003776  117646  000002          TYPEA:  MOVB    @2(6),-(6)      ;GET CHAR.
004002  001006                          BNE     TYPEB           ;BR IF NOT TERMINATOR.
004004  116716  175103                  MOVB    FILLER,(6)      ;OUTPUT FILLER.
004010  004767  000072                  JSR     PC,TTYOUT
004014  012616                  TYPEA:  MOV     (6)+,(6)
004016  000207                          RTS     PC              ;EXIT.
004020  122716  000045          TYPEB:  CMPB    #45,(6)         ;IS IT %?
004024  001020                          BNE     TYPEC           ;BR IF NOT.
004026  112716  000015          TYPEC:  MOVB    #15,(6)         ;OUTPUT CR.
004032  004767  000050                  JSR     PC,TTYOUT
004036  112746  000012                  MOVB    #12,-(6)        ;OUTPUT LF.
004042  004767  000040                  JSR     PC,TTYOUT
004046  116767  175040  174743          MOVB    FILCNT,FILCTR   ;GET FILL COUNT.
004054  001002                          BNE     TYPEE           ;BR IF NOT 0.
004056  105267  174735                  INCB    FILCTR          ;OOPS, MAKE IT A 1.
004062  116746  175025          TYPEF:  MOVB    FILLER,-(6)     ;OUTPUT FILLER.
004066  004767  000014          TYPED:  JSR     PC,TTYOUT
004072  105367  174721                  DECB    FILCTR          ;DECREMENT FILL COUNTER.
004076  001371                          BNE     TYPEE           ;BR IF NOT 0.
004100  005266  000002                  INC     2(6)            ;UPDATE CHAR POINTER.
004104  000731                          BR      TYPE
                                        ;TTYOUT SUBROUTINE.
                                        ;****************START OF UNIMON SPECIAL CODE*******************
004106  105777  174772          TTYOUT: TSTB    @TPB            ;TTY READY?
004112  001403                          BEQ     1$              ;BR IF YES.
                                        ;****************END OF UNIMON SPECIAL CODE*********************
004114  104401  004106  000000          QUE,TTYOUT,0            ;QUE TO CHECK AGAIN.
004122  116677  000002  174754  1$:     MOVB    2(6),@TPB       ;OUTPUT THE CHAR.
004130  000731                          BR      TYPEA

004132  104406  005613          COMCO1: MSG,INVCMD              ;TYPE INVALID COMMAND.
004136                          COMCON:
004136  005067  174644          COMCO2: CLR     SPCFLG          ;CLEAR SPECIAL FLAG AND
                                                                ;DIR COMMAND INDICATOR.
004142  105067  174752                  CLRB    FILLID          ;CLEAR FILL COMMAND INDICATOR.
004146  104406  005566          COMCO3: MSG,DOT                 ;TYPE DOT.
004152  000167  175116          COMCO4: JMP     INPUT           ;GO GET MORE INPUT.

                                        ;SPECIAL INPUT ROUTINE.
004156  012667  174650          SINPUT: MOV     (6)+,SRETRN     ;SAVE RETURN ADDR.
004162  105267  174620                  INCB    SPCFLG          ;SET SPECIAL FLAG.
004166  000771                          BR      COMCO4
```



```
                                        ;RUN ROUTINE. STARTS EXERCISER EXECUTION.
004170  105067  174620          RUN:    CLRB    MODCNT          ;CLEAR COUNT OF MODULES TO BE RUN.
004174  012702  006424                  MOV     #MODQ,R2        ;CLEAR MODULES PASCNT AND ERRCNT.
004200  012201                  2$:     MOV     (2)+,R1         ;MODULE ADDR TO R1.
004202  001430                          BEQ     RUNC            ;BR IF NO MORE.
004204  105767  174705                  TSTB    PWRFI           ;UP FROM POWER FAIL?
004210  001012                          BNE     1$              ;BR IF YES.
004212  105767  174700                  TSTB    CHN             ;IN CHAIN MODE?
004216  001007                          BNE     1$              ;BR IF YES.
004220  005061  000026                  CLR     PSCNT(1)        ;CLEAR MOD'S PASCNT.
004224  005061  000030                  CLR     ERCNT(1)        ;CLEAR MOD'S ERROR COUNT.
004230  042761  020000  000020          BIC     #BIT13,STAT(1)  ;CLEAR STOPPED BIT.
004236  032761  040000  000020  1$:     BIT     #BIT14,STAT(1)  ;MODULE SELECTED?
004244  001755                          BEQ     2$              ;BR IF NOT.
004246  032761  020000  000020          BIT     #BIT13,STAT(1)  ;MODULE STOPPED?
004254  001351                          BNE     2$              ;BR IF YES.
004256  105267  174532                  INCB    MODCNT          ;NO, UP COUNT OF RUNNABLE MODULES.
004262  000746                          BR      2$              ;GO CHECK NEXT MODULE.
004264  105767  174524          RUNC:   TSTB    MODCNT          ;ANY RUNNABLE MODULES?
004270  001720                          BEQ     COMCO1          ;BR IF NOT, INVALID COMMAND!
004272  116767  174516  174515          MOVB    MODCNT,MODCTR
004300  012767  006422  174514          MOV     #MODQ-2,MODPTR  ;MODULE TABLE ADDR TO MODPTR.
004306  012767  040000  174520          MOV     #40000,IOBKID   ;START WITH NON-TRACE BACKGROUND MODULES.
004314  105267  174472                  INCB    RMODE           ;ACTIVATE RUN MODE.
                                        ;****************START OF UNIMON SPECIAL CODE*******************
004320  000167  175032                  JMP     QUETST
                                        ;****************END OF UNIMON SPECIAL CODE*********************
004324  000704                  RUNB:   BR      COMCON          ;OUT.
```

```
1039
1040  004326  105267  174455          ;MAP ROUTINE. TYPES RESIDENT MODULES AND THIEIR START ADDRESS.
1041  004332  105067  001514  MAP:    INCB    DIRIND          ;SET DIR INDICATOR.
1042  004336  004767  000002          CLRB    MDIRE           ;TERMINATE ASCII STRING EARLY.
1043  004342  000770                  JSR     PC,DIRA         ;TYPE MAP.
1044                                  BR      RUNB
1046
1047  004344  005003          ;***************START OF UNIMON SPECIAL CODE*******************
1048  004346  012702  006424  DIRA:   CLR     R3              ;CLEAR MODULE NUMBER.
1049                                  MOV     #MODQ,R2        ;GET MODULE TABLE ADDR.
1054  004352  012201          ;***************END OF UNIMON SPECIAL CODE*********************
1055  004354  001455  1$:     MOV     (2)+,R1         ;GET MODULE ADDR.
1056  004356  032761  040000  000020  BEQ     5$              ;BR IF 0. ALL DONE.
1057  004364  001003          BIT     #BIT14,STAT(1)  ;MODULE SELECTED?
1058  004366  105767  174415  BNE     2$              ;BR IF YES.
1059  004372  001767          TSTB    DIRIND          ;TYPING DIRECTORY?
1060  004374  004567  000232  BEQ     1$              ;BR IF NOT. DONT TYPE UNSELECTED MODS.
1061  004400  006016  2$:     JSR     R5,FILLNM       ;FILL MOD NAME IN ASCII STRING.
1062  004402  010146          AMODNM+1                ;ADDR TO STUFF NAME IN.
1063  004404  004567  000330  MOV     R1,-(6)         ;MODULE ADDR TO STACK.
1064  004410  006030          JSR     R5,OACNV        ;CONVERT MOD ADDR TO ASCII.
1065  004412  016146  000020  APC
1066  004416  004567  000316  MOV     STAT(1),-(6)    ;CONVERT MODULE STATUS.
1067  004422  006044          JSR     R5,OACNV
1068  004424  016146  000026  AMDSTA
1069  004430  004567  000360  MOV     PSCNT(1),-(6)   ;MOD'S PASS COUNT TO STACK.
1070  004434  006063          JSR     R5,BDCNV        ;CONVERT IT TO DECIMAL ASCII.
1071  004436  016146  000030  APSCNT
1072  004442  004567  000346  MOV     ERCNT(1),-(6)   ;MOD'S ERROR COUNT TO STACK.
1073  004446  006102          JSR     R5,BDCNV        ;CONVERT IT TO DECIMAL ASCII.
1075                          AERRS
1076  004450  104406  005611  ;***************START OF UNIMON SPECIAL CODE*******************
1077  004454  105767  174327          MSG,ACRLF
1078  004460  001004          TSTB    DIRIND          ;TYPING DIRECTORY?
1079  004462  112767  000040  001362  BNE     6$              ;BR IF YES.
1080  004470  000404          MOVB    #40,MDIRE       ;NO, ALLOW FULL STRING TYPING.
1081  004472  005203          BR      3$
1082  004474  010346  6$:     INC     R3              ;INCREMENT MODULE NUMBER.
1083  004476  004767  000106  MOV     R3,-(6)         ;MOVE IT TO STACK.
1084                          JSR     PC,ITOA         ;TYPE IT.
1091  004502  104406  006015  ;***************END OF UNIMON SPECIAL CODE*********************
1092  004506  000721  3$:     MSG,AMODNM              ;TYPE ASCII LINE.
1093  004510  000207          BR      1$              ;DO IT AGAIN.
                        5$:     RTS     PC              ;EXIT.
```



```
1095
1096                            ;SELECT MODULE(S) ROUTINE.
1097  004512  012767  052761  000040  SEL:    MOV     #52761,SLDSB    ;MODIFY COMMON TO SELECT MODULE(S).
1098  004520  012767  052761  000052          MOV     #52761,SLDSE    ;
1099  004526  000406                  BR      SLDS            ;GO TO COMMON.
1100
1101                            ;DESELECT MODULE(S) ROUTINE.
1102
1103  004530  012767  042761  000022  DES:    MOV     #42761,SLDSB    ;MODIFY COMMON TO DESELFCT MODULE(S).
1104  004536  012767  042761  000034          MOV     #42761,SLDSE
1105
1106                            ;SELECT/DESELECT COMMON.
1107
1108  004544                  SLDS:
1109  004544  004767  000110  SLDSA: JSR     PC,GETNAM       ;CHECK NAME. GET MODULE ADDR.
1110  004550  000407                  BR      SLDSC           ;NO NAME. SELECT/DESELECT ALL.
1111  004552  000664                  BR      RUNB            ;INVALID OR NEX NAME.
1112  004554  016701  174242          MOV     MODPTR,R1       ;MODULE ADDR TO R1.
1113  004560  042761  040000  000020  SLDSB:  BIC     #BIT14,STAT(1)  ;SELECT/DESELECT MODULE.
1114  004566  000656                  BR      RUNB            ;DONE.
1115  004570  012702  006424  SLDSC:  MOV     #MODQ,R2        ;MODULE TABLE ADDR TO R2.
1116  004574  012201  SLDSD:  MOV     (2)+,R1         ;GET MODULE ADDR.
1117  004576  001652          BEQ     RUNB            ;BR IF 0,END OF TABLE, DONE.
1118  004600  042761  040000  000020  SLDSE:  BIC     #BIT14,STAT(1)  ;SELECT/DESELECT MODULE.
1119  004606  000772          BR      SLDSD           ;DO IT AGAIN.
1120                          ;LOCATIONS SLDSB AND SLDSE ARE PURE WHILE IN RUN MODE.
```

```
1156
1157  004610  016646  000002          ;BINARY TO ASCII TYPE ROUTINE, TYPES CONTENTS OF "NUMBER".
1158  004614  004567  000120  ITOA:   MOV     2(6),-(6)
1159  004620  006111                  JSR     R5,OACNV        ;CONVERT NUMBER TO ASCII.
1160  004622  104406  006111          .OCTAL
1161  004626  012616                  MSG,AOCTAL              ;TYPE OCTAL VALUE.
1162  004630  000207                  MOV     (6)+,(6)
1163                                  RTS     PC              ;EXIT.
1203
1204  004632  010346          FILLNM: MOV     R3,-(6)         ;SAVE R3.
1205  004634  012503                  MOV     (5)+,R3         ;STORE ADDR TO R3.
1206  004636  012704  000006          MOV     #6,R4           ;WILL DO 6 TIMES.
1207  004642  112123          1$:     MOVB    (1)+,(3)+       ;MOVE CHAR.
1208  004644  005304                  DEC     R4              ;DONE?
1209  004646  001375                  BNE     1$              ;BR IF NOT.
1210  004650  162701  000006          SUB     #6,R1           ;RESTORE R1.
1211  004654  012603                  MOV     (6)+,R3         ;RESTORE R3.
1212  004656  000205                  RTS     R5              ;EXIT.
1214                                  ;***************START OF UNIMON SPECIAL CODE*******************
1215                                  ;GETNAM SUBROUTINE.
1216  004660  005067  174136  GETNAM: CLR     MODPTR          ;
1217  004664  116767  172700  174130  MOVB    SR,MODPTR       ;GET MOD NUMBER
1218  004672  001001                  BNE     1$              ;BR IF A MODULE # SET.
1219  004674  000207                  RTS     PC              ;SELECT THEM ALL.
1220  004676  006367  174120  1$:     ASL     MODPTR          ;
1221  004702  062767  006422  174112  ADD     #MODQ-2,MODPTR  ;
1222  004710  017767  174106  174104  MOV     @MODPTR,MODPTR  ;GET MODULE ADDR.
1223  004716  001005                  BNE     2$              ;BR IF LEGAL ADDR.
1224  004720  104406  005657          MSG,INVNAM              ;
1225  004724  062716  000002          ADD     #2,(6)          ;SET UP INVALID NAME EXIT.
1226  004730  000207                  RTS     PC              ;
1227  004732  062716  000004  2$:     ADD     #4,(6)          ;SET UP SUCCESS EXIT.
1228  004736  000207                  RTS     PC              ;
1229
1230                                  ;***************END OF UNIMON SPECIAL CODE*********************
1290
1291
1292  004740  004467  000200          ;OCTAL TO ASCII CONVERT ROUTINE.
1293  004744  016600  000014  OACNV:   JSR     R4,SAV04        ;SAVE REGS 0-4.
1294  004750  012501                  MOV     12.(6),R0       ;GET OCTAL VALUE.
1295  004752  012702  000006          MOV     (5)+,R1         ;GET DEST ADDR.
1296  004756  060201                  MOV     #6,R2           ;SET CONVERT COUNT TO 6.
1297  004760  010003                  ADD     R2,R1           ;DEVELOP ADDR TO STORE 1ST CHAR.
1298  004762  042703  177770  1$:     MOV     R0,R3           ;GET VALUE TO R3.
1299  004766  062703  000060          BIC     #177770,R3      ;ISOLATE LEAST SIGNIFICANT DIGIT.
1300  004772  110341                  ADD     #60,R3          ;CONVERT TO ASCII.
1301  004774  042700  000007          MOVB    R3,-(1)         ;STORE IT.
1302  005000  006000                  BIC     #7,R0           ;CLEAR DIGIT JUST CONVERTED.
1303  005002  006000                  ROR     R0              ;SHIFT IN NEXT DIGIT.
1304  005004  006000                  ROR     R0
1305  005006  005302                  ROR     R0
1306  005010  001363                  DEC     R2              ;DONE 6 DIGITS?
1307  005012  000434                  BNE     1$              ;BR IF NOT.
1308                                  RR      XX
```



```
1309
1310  005014  004467  000124          ;BINARY TO DECIMAL ASCII CONVERT ROUTINE.
1311  005020  016601  000014  BDCNV:   JSR     R4,SAV04        ;SAVE REGS 0-4.
1312  005024  012700  006415          MOV     12.(6),R1       ;GET BIN VALUE.
1313  005030  012702  005114          MOV     #DECVAL,R0      ;GET ADDR OF DECVAL STRING.
1314  005034  012703  000005          MOV     #TENPWR,R2      ;ADDR OF TENPWR TO R2.
1315  005040  005004                  MOV     #5,R3           ;SET UP TO DO 5 CONVERSIONS.
1316  005042  161201          1$:     CLR     R4              ;CLEAR RESULT.
1317  005044  103402          2$:     SUB     (2),R1          ;SUBTRACT TEN POWER.
1318  005046  005204                  BCS     3$              ;BRIFUNSUCCESSFUL .
1319  005050  000774                  INC     R4              ;ADD 1 TO RESULT.
1320  005052  062201                  BR      2$              ;DO IT AGAIN.
1321  005054  062704  000060  3$:     ADD     (2)+,R1         ;RESTORE SUBTRACTED VALUE.
1322  005060  110420                  ADD     #60,R4          ;MAKE IT ASCII.
1323  005062  005303                  MOVB    R4,(0)+         ;SAVE IT.
1324  005064  001365                  DEC     R3              ;DONE 5?
1325  005066  012501                  BNE     1$              ;BR IF NOT.
1326  005070  012702  000005          MOV     (5)+,R1         ;GET FINAL STORE ADDR.
1327  005074  060201                  MOV     #5,R2           ;GET DIGIT COUNT DESIRED.
1328  005076  114041                  ADD     R2,R1           ;COMPUTE ADDR OF 1ST DIGIT.
1329  005100  005302          4$:     MOVB    -(0),-(1)       ;TRANSFER CHAR.
1330  005102  001375                  DEC     R2              ;DONE?
1331  005104  004767  000046          BNE     4$              ;BR IF NOT.
1332  005110  012616          XX:     JSR     PC,RST04        ;RESTORE REGS 0-4.
1333  005112  000205                  MOV     (6)+,(6)
1334  005114  023420  001750  000144  RTS     R5              ;EXIT.
      005122  000012  000001  TENPWR: 10000.,1000.,100.,10.,1
1336
1337  005126  032767  002000  172434  ;***************START OF UNIMON SPECIAL CODE*******************
1338  005134  001402          SR10CK: BIT     #2000,SR        ;SR10 SET?
1339  005136  000167  176514          BEQ     1$              ;BR IF NOT.
1340  005142  000207                  JMP     CTRLCB          ;YES, GO.
1341                          1$:     RTS     PC              ;EXIT.
                                        ;***************END OF UNIMON SPECIAL CODE*********************
```

```
 1344
 1345  005144  010346                    ;SAVE REGS 0-4 SUBROUTINE.
 1346  005146  010246                    SAV04:  MOV     R3,-(SP)        ;SAVE R3
 1347  005150  010146                            MOV     R2,-(SP)        ;SAVE R2
 1348  005152  010046                            MOV     R1,-(SP)        ;SAVE R1
 1349  005154  010407                            MOV     R0,-(SP)        ;SAVE R0
 1350                                            MOV     R4,PC           ;R4 IS ALREADY SAVED
 1351
 1352  005156  012604                    ;RESTORE REGS 0-4 SUBROUTINE.
 1353  005160  012600                    RST04:  MOV     (SP)+,R4        ;RETURN ADDRESS
 1354  005162  012601                            MOV     (SP)+,R0        ;RESTORE R0
 1355  005164  012602                            MOV     (SP)+,R1        ;R1
 1356  005166  012603                            MOV     (SP)+,R2        ;R2
 1357  005170  000204                            MOV     (SP)+,R3
 1358                                            RTS     R4              ;RESTORE R4 AND RETURN
 1359  005172  012700  001004            CLRQUS: MOV     #IOQUE,R0       ;CLEAR VARIABLES.
 1360  005176  012701  000072                    MOV     #TKS-IOQUE,R1
 1361  005202  105020                    1$:     CLRB    (0)+
 1362  005204  005301                            DEC     R1
 1363  005206  001375                            BNE     1$
 1364  005210  012767  007134  173632            MOV     #IOQ,IOQ1
 1365  005216  012767  007134  173626            MOV     #IOQ,IOQ2
 1366  005224  012767  007444  173622            MOV     #TYPEQ,TYPQ1
 1367  005232  012767  007444  173616            MOV     #TYPEQ,TYPQ2
 1368  005240  012700  007134                    MOV     #IOQ,R0         ;SET UP TO
 1369  005244  012701  000620                    MOV     #IOQL+TYPQL,R1  ;CLEAR QUEUES.
 1370  005250  105020                    2$:     CLRB    (0)+
 1371  005252  005301                            DEC     R1              ;DONE?
 1372  005254  001375                            BNE     2$              ;BR IF NOT.
 1373                                    ;***************START OF UNIMON SPECIAL CODE*******************
 1374  005256  012700  000062                    MOV     #62,R0          ;FILL VECTOR AREA WITH .+2
 1375  005262  012701  000060                    MOV     #60,R1          ;AND HALT.
 1376                                    ;***************END OF UNIMON SPECIAL CODE*********************
 1377  005266  010021                    3$:     MOV     R0,(1)+
 1378  005270  005021                            CLR     (1)+
 1379  005272  010100                            MOV     R1,R0
 1380  005274  005720                            TST     (0)+
 1381  005276  020027  001002                    CMP     R0,#1002        ;FILLED UP?
 1382  005302  001371                            BNE     3$              ;BR IF NOT.
 1383  005304  000207                            RTS     PC              ;YES. EXIT.
 1384
```



```
 1385
 1386  005306  010046                    ;TRAP INTERPRETER ROUTINE.
 1387  005310  016600  000002            TRPINT: MOV     R0,-(6)         ;PUSH R0.
 1388  005314  014000                            MOV     2(6),R0         ;GET TRAP PC.
 1389  005316  006300                            MOV     -(0),R0         ;GET TRAP CALL.
 1390  005320  016000  174340                    ASL     R0              ;MULTIPLY BY 2.
 1391  005324  020027  005364                    MOV     TRPTAB-TRP2(0),R0  ;FORM TRAP ROUTINE ADDR.
 1392  005330  103001                            CMP     R0,#TRPLIM      ;WITHIN LIMITS?
 1393  005332  000200                            BHIS    1$              ;BR IF NOT.
 1394  005334  005767  172441                    RTS     R0              ;GO TO ROUTINE, RESTORE R0.
 1395                                    1$:     TST     1               ;ERROR INVALID TRAP CALL, CRASH SYSTEM.
 1396
 1397          011000                            TRP2=11000
 1398          000000                            TRAPX=0
 1399  005340                            TRPTAB:
 1400  005340  003266                            EXIT.                   ;POINTER FOR TRAP CALL EXIT
 1401  005342  003554                            QUE.                    ;POINTER FOR TRAP CALL QUE
 1402  005344  003444                            TYPQ.                   ;POINTER FOR TRAP CALL ENDPAS
 1403  005346  003444                            TYPQ.                   ;POINTER FOR TRAP CALL END
 1404  005350  003362                            TYPQ2.                  ;POINTER FOR TRAP CALL ERROR
 1405  005352  003362                            TYPQ2.                  ;POINTER FOR TRAP CALL DATERR
 1406  005354  003454                            TYPQ1.                  ;POINTER FOR TRAP CALL MSG
 1407  005356  003444                            TYPQ.                   ;POINTER FOR TRAP CALL BREAK
 1408  005360  003362                            TYPQ2.                  ;POINTER FOR TRAP CALL ERRORN
 1409  005362  003444                            TYPQ.                   ;POINTER FOR TRAP CALL MSGN
 1410  005364                            TRPLIM:
 1411
 1412
 1413  005364  012746  000004            ;BUS ERROR AND RESERVED INSTRUCTION TRAP ROUTINES
 1414  005370  000402                    BUSERR: MOV     #4,-(6)         ;INDICATE BUS ERROR TRAP.
 1415  005372  012746  000010                    BR      RESIA
 1416  005376  010605                    RESINT: MOV     #10,-(6)        ;INDICATE RESERVED INSTRUCTION TRAP.
 1417  005400  005725                    RESIA:  MOV     R6,R5           ;SAVE SP POINTER.
 1418  005402  010546                            TST     (5)+            ;SET TO VALUE AT TRAP TIME.
 1419  005404  000240                            MOV     R5,-(6)         ;SAVE IN STACK.
 1420  005406  004767  177560                    NOP                     ;CLEAR THE WORLD.
 1421  005412  104406  005714                    JSR     PC,CLRQUS       ;CLEAR QUEUES.
 1422  005416  012705  004610                    MSG,SYSERR              ;TYPE SYS ERROR FAILURE.
 1423  005422  004715                            MOV     #ITOA,R5
 1424  005424  004715                            JSR     PC,(5)          ;TYPE SP AT TIME OF ERROR.
 1425  005426  004715                            JSR     PC,(5)          ;TYPE TRAP ADDR.
 1426  005430  004715                            JSR     PC,(5)          ;TYPE ERR PC.
 1427  005432  000167  176254                    JSR     PC,(5)          ;TYPE ERR PSW.
 1428                                            JMP     CTRLCC          ;GO TYPE SUMMARY.
```

```
1436                                            ;POWER DOWN AND POWER UP ROUTINES.
1437  005436  012767  005454  172360  PWRDN:  MOV     #PWRUP,PWRFV    ;SET UP POWER UP VECTOR.
1438  005444  116767  173342  173443          MOVB    RMODE,PWRFI     ;SAVE RUN MODE INDICATOR.
1439  005452  000000                          HALT
1440  005454  012767  005436  172342  PWRUP:  MOV     #PWRDN,PWRFV    ;SET UP POWER DOWN VECTOR.
1441  005462  012706  007134                  MOV     #SPBOT,R6       ;RESET STACK.
1442  005466  004767  177500                  JSR     PC,CLRQUS       ;CLEAR QUEUES.
1443  005472  104406  005730                  MSG,PWRFAI              ;TYPE POWER FAILURE MESSAGE.
1444  005476  105767  173413                  TSTB    PWRFI           ;WERE WE IN RUN MODE?
1445  005502  001402                          BEQ     CHNOUT          ;BR IF NOT, WAIT FOR USER.
1446  005504  000167  000026                  JMP     LOGICA          ;GO RUN AGAIN.
1447
1448
1449  005510  105767  173402          CHNOUT: TSTB    CHN             ;IN CHAIN MODE?
1450  005514  001002                          BNE     1$              ;BR IF YES.
1451  005516  000167  176414                  JMP     COMCON          ;BACK TO KYBD SERVICE.
1452  005522  016700  172314          1$:     MOV     42,R0
1453  005526  004710                  LOGIC:  JSR     PC,(0)          ;RETURN TO MONITOR.
1454  005530  000240  000240  000240          .WORD   NOP,NOP,NOP
1455  005536  000167  176426          LOGICA: JMP     RUN             ;GO START ANOTHER PASS.
```



```
1482                                            ;PURE MESSAGES
1483  005542  042045  041505  054057  TITLE:  .ASCIZ  '%DEC/X11 EXERCISER%'
      005550  030461  042440  042530
      005556  041522  051511  051105
      005564  000045
1484  005566  027045     000          DOT:    .ASCIZ  '%.'
1485  005571     045  052522  020116  SUMARY: .ASCIZ  '%RUN SUMMARY%'
      005576  052523  046515  051101
      005604  022531     000
1486  005607     136  022503     000  CTRLC:  .ASCIZ  '^C%'
1487  005613     045  047111  040526  INVCMD: .ASCIZ  '%INVALID COMMAND'
      005620  044514  020104  047503
      005626  046515  047101  000104
1488  005634  044445  053116  046101  INVADR: .ASCIZ  '%INVALID ADDR/DATA'
      005642  042111  040440  042104
      005650  027522  040504  040524
      005656     000
1489  005657     045  047111  040526  INVNAM: .ASCIZ  '%INVALID/NEX NAME'
      005664  044514  027504  042516
      005672  020130  040516  042515
      005700     000
1490  005701     045  041113  043125  KBOFLO: .ASCIZ  '%KBUF OFLO'
      005706  047440  046106  000117
1491  005714  051445  051531  042440  SYSERR: .ASCIZ  '%SYS ERROR '
      005722  051122  051117  000040
1492  005730  050045  051127  043040  PWRFAI: .ASCIZ  '%PWR FAILURE'
      005736  044501  052514  042522
      005744     000
1493  005745     045  051127  052111  ROTENB: .ASCIZ  '%WRITE BUFFER ROTATION ENABLED, RANGE: '
      005752  020105  052502  043106
      005760  051105  051040  052117
      005766  052101  047511  020116
      005774  047105  041101  042514
      006002  027104  051040  047101
      006010  042507  020072     000
```

```
 1495
 1497  006015      040  020040  020040  ;IMPURE MESSAGES.
       006022  020040  040440  020124  AMODNM: .ASCII  '       AT '
 1502  006030  020040  020040  020040  APC:    .ASCII  '       STAT '
       006036  051440  040524  020124
 1503  006044  020040  020040  020040  AMDSTA: .ASCII  '       '
 1504  006052  020040  040520  041523  MDIRE:  .ASCII  '  PASCNT '
       006060  052116      040
 1505  006063      040  020040  020040  APSCNT: .ASCII  '       ERRCNT '
       006070  020056  042440  051122
       006076  047103  020124
 1506  006102  020040  020040  027040  AERR8:  .ASCIZ  '       .'
       006110      000
 1507  006111      040  020040  020040  AOCTAL: .ASCIZ  '       '
       006116  020040  020040      000
 1508  006123      045  020045  020040  AEND:   .ASCII  '%%      PC '
       006130  020040  020040  041520
       006136      040
 1509  006137      040  020040  020040  AEND1:  .ASCII  '       APC '
       006144  020040  040440  041520
       006152      040
 1510  006153      040  020040  020040  AEND2:  .ASCIZ  '       '
       006160  020040      000
 1511  006163      040  051104  050117  MODEND: .ASCIZ  ' DROPPED%'
       006170  042520  022504      000
 1512  006175      040  047105  050104  APSEND: .ASCII  ' ENDPAS '
       006202  051501      040
 1513  006205      040  020040  020040  BPSCNT: .ASCIZ  '      .%'
       006212  022456      000
 1514  006215      045  051503  040522  AERROR: .ASCII  '%CSRA '
       006222      040
 1515  006223      040  020040  020040  ACSRAC: .ASCII  '       CSRC '
       006230  020040  051503  041522
       006236      040
 1516  006237      040  020040  020040  ACSRC:  .ASCII  '       STATC '
       006244  020040  052123  052101
       006252  020103
 1517  006254  020040  020040  020040  ASTATC: .ASCIZ  '       %'
       006262  022440      000
 1518  006265      045  051503  040522  ADTERR: .ASCII  '%CSRA '
       006272      040
 1519  006273      040  020040  020040  ADTE6:  .ASCII  '       S/B '
       006300  020040  027523  020102
 1520  006306  020040  020040  020040  ADTE3:  .ASCII  '       WAS '
       006314  053440  051501      040
 1521  006321      040  020040  020040  ADTE2:  .ASCII  '       SBADR '
       006326  020040  041123  042101
       006334  020122
 1522  006336  020040  020040  020040  ADTE5:  .ASCII  '       WASADR '
       006344  053440  051501  042101
       006352  020122
 1523  006354  020040  020040  020040  ADTE4:  .ASCIZ  '       DATA ERROR%'
       006362  020040  040504  040524
       006370  042440  051122  051117
```



```
       006376  000045
 1524  006400  042440  051122  020043  ERRNMB: .ASCII  ' ERR# '
 1525  006406  020040  020040  027040  AERNMB: .ASCIZ  '      .'
       006414      000
 1526  006415      040      040      040  DECVAL: .BYTE   40,40,40,40,40,40
       006420      040      040      040
 1527
 1528
 1529
 1530
 1531              006424                ;BUFFER AREAS.
 1532  006424                            .EVEN
 1541  006734  000100                    MODQ:
 1542  007134                            .BLKW   64.             ;ROOM FOR 100 MODULE POINTERS.
 1543  007134  000310                    SPBOT:
 1544  007444  000310                    IOQ:    .BLKB   IOQL
 1545                                    TYPEQ:  .BLKB   TYPQL
 1546              007754
 1547              004657                .=.
                                         BUFSIZ=AMODNM-START
```

```
                                        ;ROUTINE TO DETERMINE WHETHER WRITE BUFFER ROTATION SHOULD TAKE PLACE,
                                        ;AND TO DETERMINE CORE LIMITS OF BUFFER ROTATION, ALSO TO DETERMINE USE OF
                                        ;RTI OR RTT INSTRUCTION.
                    007134                      .=IOQ
                                        ;CHECK FOR USE OF RTT OR RTI INSTRUCTION.
007134                                  SETBUF:
007134  011605                                  MOV     (6),R5          ;SAVE RETURN POINTER.
007136  012767  007156  170644                  MOV     #2$,RESIV       ;SET UP RESERVED INSTRUCTION TRAP.
007144  005046                                  CLR     -(6)
007146  012746  007154                          MOV     #1$,-(6)        ;SET UP TO EXIT WITH RTT INSTRUCTION.
007152  000006                                  RTT                     ;IF RTT NOT VALID IT WILL TRAP OUT.
007154  000406                          1$:     BR      3$              ;OK IF NO TRAP OCCURS.
007156  012767  000002  174062          2$:     MOV     #RTI,TRCIA      ;TRAP COMES HERE, CHANGE RTT'S TO RTI'S.
007164  012767  000002  172610                  MOV     #RTI,RTT1
007172  012767  005372  170610          3$:     MOV     #RESINT,RESIV   ;RESTORE RES INST VECTOR.
007200  005267  170644                  10$:    INC     LOCORE          ;SET LOCORE NON-ZERO.
007204  005067  170570                  9$:     CLR     0               ;CLEAR LOC 0.
007210  000205                                  RTS     R5              ;EXIT.
007212  177776  000001  177775          WCASE1: .WORD   177776,1,177775,2,177773,4,177767,10
007220  000002  177773  000004
007226  177767  000010
007232  177757  000020  177737                  .WORD   177757,20,177737,40,177677,100,177577,200
007240  000040  177677  000100
007246  177577  000200
007252  177377  000400  176777                  .WORD   177377,400,176777,1000,175777,2000,173777,4000
007260  001000  175777  002000
007266  173777  004000
007272  167777  010000  157777                  .WORD   167777,10000,157777,20000,137777,40000,77777,100000
007300  020000  137777  040000
007306  077777  100000
007312                                  WCASEE:
                    000001                      .END
```



```
ACRLF  = 005611        ACSR   = 000052        ACSRAC  006223        ACSRC   006237
ADDR     001024        ADTERR   006265        ADTE2   006321        ADTE3   006306
ADTE4    006354        ADTES    006336        ADTE6   006273        AEND    006123
AEND1    006137        AEND2    006153        AERNMB  006406        AERROR  006215
AERRS    006102        AMDSTA   006044        AMODNM  006015        AOCTAL  006111
APC      006030        APSCNT   006063        APSEND  006175        ASB    = 000056
ASTAT  = 000054        ASTATC   006254        AWAS   = 000060        BDCNV   005014 G
BIT0   = 000001        BIT1   = 000002        BIT10  = 002000        BIT11  = 004000
BIT12  = 010000        BIT13  = 020000        BIT14  = 040000        BIT15  = 100000
BIT2   = 000004        BIT3   = 000010        BIT4   = 000020        BIT5   = 000040
BIT6   = 000100        BIT7   = 000200        BIT8   = 000400        BIT9   = 001000
BKQSVC   002004        BKQUE    001010        BPSCNT   006205        BRAKE   001011
BREAK  = 104407        BREAK,   002256        BUFSIZ = 004657        BUSERR  005364
BUSEV    000004        CADDR    001132        CHN      001116        CHNOUT  005510
CLRQUS   005172        COMCON   004136        COMCO1   004132        COMCO2  004136
COMCO3   004146        COMCO4   004152        COMQUE   001130        CSRA   = 000050
CSTART   001134        CTRLC    005607        CTRLCA   003652        CTRLCB  003656
CTRLCC   003712        CTRLCD   003722        CTRLX    003726        DATERR = 104405
DDPPTR   000042        DECVAL   006415        DES      004530        DIRA    004344
DIRIND   001007        DOT      005566        DRCSR2   001124        DRIDB1  001122
DRIDR2   001126        DSTADR   001030        EABITS   000054 G      EMTV    000030
END    = 104403        ENDCOM   002720        ENDPAS = 104402        ENDSVA  003024
ENDSVB   003052        ENDSVC   003016        ENDSVD   003110        ENDSVE  003064
ERCNT  = 000030        ERRIND   001016        ERRLIM   001106        ERRNMB  006400
ERROR  = 104404        ERRORN = 104410        ERRSVC   002312        ERSVCA  002320
ERSVCB   002366        ERSVCC   002372        ERSVC1   002304        ERSVC2  002274
EXIT   = 104400        EXIT,    003266        EXIT1,   003270        EXIT2,  003350
FILCNT   001112        FILCTR   001017        FILLER   001113        FILLID  001120
FILLNM   004632        GETNAM   004660        HICORE   000052 G      IE     = 000100
INIT   = 000022        INPUT    001274        INVADR   005634        INVCMD  005613
INVNAM   005657        IOBKID   001034        IOQ      007134        IOQL   = 000310
IOQLIM = 007444        IOQSVA   001704        IOQSVB   001772        IOQSVC  001614
IOQUE    001004        IOQ1     001050        IOQ2     001052        IOTV    000020
ITOA     004610        KBOFLO   005701        KBPTR    001020        KBUFL  = 000040
LDIOQ    003560        LDTYPQ   003464        LINE1  = ****** U      LINE2  = ****** U
LOCORE   000050 G      LOGIC    005526        LOGICA   005536        MAP     004326
MDIRF    006052        MODCNT   001014        MODCTR   001015        MODEND  006163
MODPTR   001022        MODQ     006424 G      MONR0    001060        MONR1   001062
MONR2    001064        MONR3    001066        MONR4    001070        MONR5   001072
MSG    = 104406        MSGN   = 104411        MSGN,    002134        MSG,    002222
MSG1,    002240        NUMBER   001026        OACNV    004740 G      OPEN   = 000000
PASEND   003114        PC     =%000007        PIRQ   = 000004        PIRQ,   003560
POPSP  = 005726        POPSP2 = 022626        PRTY4  = 000200        PRTY5  = 000240
PRTY6  = 000300        PRTY7  = 000340        PS     = 177776        PSCNT  = 000026
PSENDA   003200        PSENDB   003234        PSW    = 177776        PUSH   = 005746
PUSH2  = 024646        PWRDN    005436        PWRFAI   005730        PWRFI   001115
PWRFV    000024        PWRUP    005454        QTSTB    001416        QTSTC   001430
QUE    = 104401        QUETST   001356        QUE,     003554        RESIA   005376
RESINT   005372        RESIV    000010        RMODE    001012        ROTENB  005745
ROTI     001117        RSTAT    001042        RSTO4    005156        RTT1    002002
RUN      004170        RUNB     004324        RUNC     004264        RUNSVC  001436
R0     =%000000        R1     =%000001        R2     =%000002        R3     =%000003
R4     =%000004        R5     =%000005        R6     =%000006        SAVO4   005144
SBADR  = 000052        SEL      004512        SETBUF   007134        SINPUT  004156
```

```
SLDS     004544        SLDSA    004544        SLDSB    004560        SLDSC    004570
SLDSD    004574        SLDSE    004600        SP     =%000006        SPBOT    007134
SPCFLG   001006        SPOINT = 000024        SPSAV    001074        SR     = 177570
SRETRN   001032        SR1OCK   005126        START    001136        STAT   = 000020
STAT1  = 000021        SUMARY   005571        SVR0   = 000032        SVR1   = 000034
SVR2   = 000036        SVR3   = 000040        SVR4   = 000042        SVR5   = 000044
SVR6   = 000046        SYSERI   001114        SYSERR   005714        TABADR   001040
TENPWR   005114        TITLF    005542        TKB      001100        TKS      001076
TPB      001104        TPS      001102        TRAPX  = 000012        TRCI     003240
TRCIA    003246        TRCIB    003250        TRCPC    001044        TRCPSW   001046
TRCV     000014        TRPINT   005306        TRPLIM   005364        TRPTAB   005340
TRPV     000034        TRP2   = 011000        TTYBSY   001013        TTYBYT   001110
TTYOUT   004106        TYPDAT   002754        TYPE     003770        TYPEA    004014
TYPEB    004020        TYPEC    004026        TYPED    004066        TYPEE    004062
TYPEO    007444        TYPLIM = 007754        TYPQL  = 000310        TYPQUE   001005
TYPQ.    003444        TYPQ1    001054        TYPQ1.   003454        TYPQ2    001056
TYPQ2.   003362        TYPSVC   002032        TYPTAB   002114        UC15   = 000000
WASADR = 000054        WBUF     000056 G      WCASE    007212        WCASEE   007312
XX       005104        YES      001036        .      = 007312

         000000
```

ERRORS DETECTED: 0


```
*,XQACB,PRT/N_DCXMON,P11/EQ:UC15
RUN-TIME:  7 12 0 SECONDS
CORE USED:  4K
```

```
    1
  213
                        .REM    !
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DXUCA-B-D
PRODUCT NAME:	XUCAB-DEC/X11 UC15 MODULE
DATE:	JUNE 15, 1973
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR(S):	R. CHRISTOPHER

1. ABSTRACT

THIS MODULE IN COMBINATION WITH THE PDP-15 SYSTEM EXERCISER MODULE UC15, EXERCISES THE UNICHANNEL15 HARDWARE WHICH CONSISTS OF:

A. MX15-B
B. DR15
C. 2 DR11-C'S

2. REQUIREMENTS

HARDWARE: UNICHANNEL15
STORAGE: THE XUCA MODULE REQUIRES 1400 OCTAL WORDS OF STORAGE

3. PASS DEFINITION

ONE PASS OF THE XUCA MODULE IS DEFINED AS RUNNING EACH OF THE MODULE'S FIVE ROUTINES ONCE.

4. EXECUTION TIME

THE XUCA MODULE RUNNING ALONE TAKES APPROXIMATELY ONE MINUTE TO COMPLETE A PASS.

5. CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 167760, VECTOR: 300, BR1:5, BR2:7, DEVCNT:1

REQUIRED PARAMETERS: NONE

6. DEVICE/OPTION SETUP

NONE

7. MODULE OPERATION

TEST SEQUENCE:

ROUTINE 0 - CHECKS A 125252 PATTERN WHICH HAS BEEN WRITTEN BY THE PDP-15 EXERCISER MODULE.

ROUTINE 1 - CHECKS A 52525 PATTERN WHICH HAS BEEN WRITTEN BY THE PDP-15 EXERCISER MODULE.

ROUTINE 2 - WRITES A 125252 PATTERN TO BE CHECKED BY THE PDP-15 EXERCISER MODULE.

ROUTINE 3 - WRITES A 52525 PATTERN TO BE CHECKED BY THE PDP-15 EXERCISER MODULE.

ROUTINE 4 - CHECKS FOR CORRECT TCBP'S SENT FROM THE PDP-15 EXERCISER MODULE.

8. OPERATION OPTIONS

NONE

9. NON STARDARD PRINTOUTS

A. A DATA ERROR OCCURRING IN ROUTINE 4 WILL TYPE OUT MEANINGFULL INFORMATION ONLY FOR LOC'S ASB AND AWAS.

B. A DATA ERROR BEING OUTPUT FROM SUBROUTINE RPTERR WILL ALWAYS HAVE THE SAME ADDRESS IN LOC'S SBADR AND WASADR AND WILL OUTPUT THE DATA EXPECTED FROM ONE OF THE COMMON MEMORY TESTS AND THE DATA ACTUALLY READ. THE CSRA IS MEANINGLESS.


```
                1
000000'                 BKMOD <XUCAB >,167760,300,5,7
000000'                 MODULE  40020,XUCAB ,167760,300,5,7,
                        .TITLE  XUCAB DEC/X11 SYSTEM EXERCISER MODULE
                                .LIST   BIN
000000'                         ;*****************************************************************************
000000' 052530  040503  020102  BEGIN:
000006' 167760                  MODNAM: .ASCII  /XUCAB /                ;MODULE NAME.
000010' 000300                  ADDR:   167760+0                        ;1ST DEVICE ADDR.
000012'    240                  VECTOR: 300+0                           ;1ST DEVICE VECTOR.
000013'    340                  BR1:    .BYTE   PRTY5+0                 ;1ST BR LEVEL.
000014' 000001                  BR2:    .BYTE   PRTY7+0                 ;2ND BR LEVEL.
000016' 000000                  DVID1:  +1                              ;DEVICE INDICATOR 1.
                                SR1:    OPEN                            ;SWITCH REGISTER 1
000020' 040020                  ;*****************************************************************************
000022' 000206'                 STAT:   40020                           ;STATUS WORD.
000024' 000162'                 INIT:   START                           ;MODULE START ADDR.
000026' 000000                  SPOINT: MODSP                           ;MODULE STACK POINTER.
000030' 000000                  PASCNT: 0                               ;PASS COUNTER.
000032' 000000                  ERRCNT: 0                               ;ERROR COUNTER.
000034' 000000                  SVR0:   OPEN                            ;LOC TO SAVE R0.
000036' 000000                  SVR1:   OPEN                            ;LOC TO SAVE R1.
000040' 000000                  SVR2:   OPEN                            ;LOC TO SAVE R2.
000042' 000000                  SVR3:   OPEN                            ;LOC TO SAVE R3.
000044' 000000                  SVR4:   OPEN                            ;LOC TO SAVE R4.
000046' 000000                  SVR5:   OPEN                            ;LOC TO SAVE R5.
000050' 000000                  SVR6:   OPEN                            ;LOC TO SAVE R6.
000052'                         CSRA:   OPEN                            ;ADDR OF CURRENT CSR.
000052' 000000                  SBADR:                                  ;ADDR OF GOOD DATA, OR
000054'                         ACSR:   OPEN                            ;CONTENTS OF CSR.
000054' 000000                  WASADR:                                 ;ADDR OF BAD DATA, OR
000056' 000000                  ASTAT:  OPEN                            ;STATUS REG CONTENTS.
000060' 000000                  ASB:    OPEN                            ;EXPECTED DATA.
                                AWAS:   OPEN                            ;ACTUAL DATA.
                                        .REPT   SPSIZ                   ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000162'                         MODSP:
334                             ;*****************************************************************************
335  000162' 000000                     ;
336  000164' 000000             TCBP:   OPEN
337  000166' 000000             TEMP:   OPEN
338  000170' 000000             TMP:    OPEN
339  000172' 000137  000414'    COMM:   OPEN
340  000176' 000000             TCBPFL: JMP @#STA,3
341  000200' 000177  177772     APIDN:  OPEN
342  000204' 000000             APIDNE: JMP @APIDN
343                             CMEMBF: OPEN
344  000206' 012704  000001             ;
345  000212' 016705  177570     START:  MOV #1,R4
346  000216' 016700  177566     STA,A:  MOV ADDR,R5                     ;GET DEVICE'S 1ST REG ADDR.
                                        MOV VECTOR,R0                   ;GET 1ST DEVICE VECTOR
```

```
  347  000222' 012720  000200'                MOV #APIDNE,(R0)+       ;INIT 1ST VECTOR
  348  000226' 116720  177560                 MOVB BR1,(R0)+
  349  000232' 016700  177552                 MOV VECTOR,R0
  350  000236' 012760  000172' 000010         MOV #TCBPFL,10(R0)      ;INIT 2ND VECTOR
  351  000244' 116760  177543  000012         MOVB BR2,12(R0)
  352  000252' 012737  000276' 000176'        MOV #STA,B,@#APIDN      ;INIT FOR CORRECT VECTOR
  353  000260' 004737  001344'                 JSR PC,@#PEND
  354  000264' 112765  000047  000012         MOVB #47,12(R5)         ;API L0 TO 47
  355  000272' 000137  001140'                 JMP @#WA.1
  356  000276'                         STA.B:
  (1)                                          ;-----------------------------------------------------------
  (1)  000276' 000004  000304' 000000'         PIRQ.,STA.C,BEGIN       ;QUEUE REQUEST TO CONTINUE AT STA.C
  (1)                                          ;-----------------------------------------------------------
  357  000304' 042765  000100  000010  STA.C: BIC #100,10(R5)         ;DISABLE APIDNE INT
  358  000312' 004737  000406'                 JSR PC,@#STA.2
  359  000316' 001431                          BEQ STA.D
  360  000320' 016565  000004  000004  ST.A:  MOV 4(R5),4(R5)
  361  000326' 004737  000406'                 JSR PC,@#STA.2
  362  000332' 001372                          BNE ST.A
  363  000334' 016501  000014                  MOV 14(R5),R1
  364  000340' 006101                          ROL R1
  365  000342' 006101                          ROL R1
  366  000344' 006101                          ROL R1
  367  000346' 006101                          ROL R1
  368  000350' 006101                          ROL R1
  369  000352' 042701  117777                  BIC #117777,R1
  370  000356' 066501  000004                  ADD 4(R5),R1
  371  000362' 010137  000204'                 MOV R1,@#CMEMBF
  372  000366' 162701  000002                  SUB #2,R1
  373  000372' 010137  000170'                 MOV R1,@#COMM
  374  000376' 000137  000444'                 JMP @#ROU0
  375  000402'                         STA.D:
  (1)  000402' 104403  000000'                 END,,BEGIN              ;TCBP FLG SET UNEXPECTEDLY
  376  000406' 012715  000100          STA.2:  MOV #100,(R5)           ;ENABLE TCBP INT
  377  000412' 104400                          EXIT.                   ;RETURN TO MONITOR.
  378  000414'                         STA.3:
  (1)                                          ;-----------------------------------------------------------
  (1)  000414' 000004  000422' 000000'         PIRQ.,STA.4,BEGIN       ;QUEUE REQUEST TO CONTINUE AT STA.4
  (1)                                          ;-----------------------------------------------------------
  379  000422' 042765  000100  000010  STA.4: BIC #100,10(R5)         ;DISABLE APIDNE INT
  380  000430' 042715  000100                  BIC #100,(R5)           ;DISABLE TCBP INT
  381  000434' 032765  000002  000014          BIT #2,14(R5)           ;INITIAL ROUTINE?
  382  000442' 000207                          RTS PC                  ;RETURN TO SEQUENCE
  383                                          ;
  384  000444' 004737  000406'         ROU0:   JSR PC,@#STA.2
  385  000450' 001032                          BNE ROU0.7
  386  000452' 012702  125252                  MOV #125252,R2
  387  000456' 004737  001226'                 JSR PC,@#CKDATA
  388  000462' 012737  000506' 000176'         MOV #ROU0.5,@#APIDN     ;INIT FOR CORRECT VECTOR
  389  000470' 004737  001344'                 JSR PC,@#PEND
  390  000474' 112765  000047  000012          MOVB #47,12(R5)         ;API L0 TO 47
  391  000502' 000137  001114'                 JMP @#WAIT
  392  000506' 005277  177456          ROU0.5: INC @COMM
  393                                          ;-----------------------------------------------------------
```



```
  (1)  000512' 000004  000520' 000000'         PIRQ.,ROU0,6,BEGIN      ;QUEUE REQUEST TO CONTINUE AT ROU0.6
  (1)                                          ;-----------------------------------------------------------
  394  000520' 042715  000100          ROU0.6: BIC #100,(R5)           ;DISABLE TCBP INT
  395  000524' 042765  000100  000010          BIC #100,10(R5)         ;DISABLE APIDNE INT
  396  000532' 000137  000542'                 JMP @#ROU1
  397  000536' 000137  000320'         ROU0.7: JMP @#ST.A
  398                                          ;
  399  000542' 004737  000406'         ROU1:   JSR PC,@#STA.2
  400  000546' 001373                          BNE ROU0.7
  401  000550' 012702  052525                  MOV #52525,R2
  402  000554' 004737  001226'                 JSR PC,@#CKDATA
  403  000560' 012737  000604' 000176'         MOV #ROU1.5,@#APIDN     ;INIT FOR CORRECT VECTOR
  404  000566' 004737  001344'                 JSR PC,@#PEND
  405  000572' 112765  000047  000013          MOVB #47,13(R5)         ;API L1 TO 47
  406  000600' 000137  001114'                 JMP @#WAIT
  407  000604' 005277  177360          ROU1.5: INC @COMM
  408                                          ;-----------------------------------------------------------
  (1)  000610' 000004  000616' 000000'         PIRQ.,ROU1.6,BEGIN      ;QUEUE REQUEST TO CONTINUE AT ROU1.6
  (1)                                          ;-----------------------------------------------------------
  409  000616' 042715  000100          ROU1.6: BIC #100,(R5)           ;DISABLE TCBP INT
  410  000622' 042765  000100  000010          BIC #100,10(R5)         ;DISABLE APIDNE INT
  411                                          ;
  412  000630' 004737  000406'         ROU2:   JSR PC,@#STA.2
  413  000634' 001032                          BNE ROU2.6
  414  000636' 012702  125252                  MOV #125252,R2
  415  000642' 004737  001316'                 JSR PC,@#WRDATA
  416  000646' 012737  000672' 000176'         MOV #ROU2.4,@#APIDN     ;INIT FOR CORRECT VECTOR
  417  000654' 004737  001344'                 JSR PC,@#PEND
  418  000660' 112765  000047  000002          MOVB #47,2(R5)          ;API L2 TO 47
  419  000666' 000137  001114'                 JMP @#WAIT
  420  000672' 005277  177272          ROU2.4: INC @COMM
  421                                          ;-----------------------------------------------------------
  (1)  000676' 000004  000704' 000000'         PIRQ.,ROU2.5,BEGIN      ;QUEUE REQUEST TO CONTINUE AT ROU2.5
  (1)                                          ;-----------------------------------------------------------
  422  000704' 042715  000100          ROU2.5: BIC #100,(R5)           ;DISABLE TCBP INT
  423  000710' 042765  000100  000010          BIC #100,10(R5)         ;DISABLE APIDNE INT
  424  000716' 000137  000726'                 JMP @#ROU3
  425  000722' 000137  000320'         ROU2.6: JMP @#ST.A
  426                                          ;
  427  000726' 004737  000406'         ROU3:   JSR PC,@#STA.2
  428  000732' 001373                          BNE ROU2.6
  429  000734' 012702  052525                  MOV #52525,R2
  430  000740' 004737  001316'                 JSR PC,@#WRDATA
  431  000744' 012737  000770' 000176'         MOV #ROU3.4,@#APIDN     ;INIT FOR CORRECT VECTOR
  432  000752' 004737  001344'                 JSR PC,@#PEND
  433  000756' 112765  000047  000003          MOVB #47,3(R5)          ;API L3 TO 47
  434  000764' 000137  001114'                 JMP @#WAIT
  435  000770' 005277  177174          ROU3.4: INC @COMM
  436                                          ;-----------------------------------------------------------
  (1)  000774' 000004  001002' 000000'         PIRQ.,ROU3.5,BEGIN      ;QUEUE REQUEST TO CONTINUE AT ROU3.5
  (1)                                          ;-----------------------------------------------------------
  437  001002' 042715  000100          ROU3.5: BIC #100,(R5)           ;DISABLE TCBP INT
  438  001006' 042765  000100  000010          BIC #100,10(R5)         ;DISABLE APIDNE INT
  439  001014' 000137  001024'                 JMP @#ROU4
```

```
  440  001020' 000137  000320'          ROU3.6: JMP @#ST.A
  441                                    ;
  442  001024' 005037  000162'          ROU4:   CLR @#TCBP
  443  001030' 004737  000406'          ROU4.1: JSR PC,@#STA.2
  444  001034' 001371                           BNE ROU3.6
  445  001036' 016537  000004  000164'          MOV 4(R5),@#TEMP
  446  001044' 023737  000162' 000164'          CMP @#TCBP,@#TEMP       ;TCBP CORRECT?
  447  001052' 001410                           BEQ ROU4.2              ;BR IF YES.
  448  001054' 013737  000162' 000056'          MOV @#TCBP,@#ASB
  449  001062' 013737  000164' 000060'          MOV @#TEMP,@#AWAS
  450                                           ;*****************************************************************
  (1)  001070' 104405  000000'                  DATER.,BEGIN            ;DATA ERROR!!!
  (1)                                           ;*****************************************************************
  451  001074' 012737  001154' 000176' ROU4.2: MOV #ROU4.3,@#APIDN     ;INIT FOR CORRECT VECTOR
  452  001102' 004737  001344'                  JSR PC,@#PEND
  453  001106' 112765  000047  000012          MOVB #47,12(R5)         ;API LO TO 47
  454  001114' 105765  000010           WAIT:  TSTB 10(R5)             ;APIDNE FLG SET?
  455  001120' 100407                           BMI WA.1                ;BR IF YES.
  456  001122' 105715                           TSTB (R5)               ;TCBP FLG SET?
  457  001124' 100373                           BPL WAIT                ;BR IF NO.
  458  001126' 004737  000406'                  JSR PC,@#STA.2
  459  001132' 001006                           BNE WA.2
  460  001134' 000137  000402'                  JMP     @#STA,D                 ;BETTER BRANCH FOR INITIAL ROUTINE
  461  001140' 012765  000100  000010  WA.1:   MOV #100,10(R5)         ;ENABLE APIDNE INT.
  462  001146' 104400                           EXIT.                   ;RETURN TO MONITOR.
  463  001150' 000137  000320'          WA.2:  JMP @#ST.A
  464  001154' 005277  177010           ROU4.3: INC @COMM
  465                                           ;-----------------------------------------------------------------
  (1)  001160' 000004  001166' 000000'          PIRQ.,ROU4.4,BEGIN      ;QUEUE REQUEST TO CONTINUE AT ROU4.4
  (1)                                           ;-----------------------------------------------------------------
  466  001166' 042715  000100           ROU4.4: BIC #100,(R5)           ;DISABLE TCBP INT
  467  001172' 042765  000100  000010          BIC #100,10(R5)         ;DISABLE APIDNE INT
  468  001200' 062737  000002  000162'         ADD #2,@#TCBP           ;FINISHED?
  469  001206' 001310                           BNE ROU4.1              ;BR IF NO.
  470  001210' 005304                           DEC R4                  ;END OF PASS?
  471  001212' 001003                           BNE ROU4.5              ;BR IF NO.
  472  001214' 104402  000206' 000000'          ENDPS.,START,BEGIN      ;SIGNAL END OF PASS, RESUME AT START
  473  001222' 000137  000212'          ROU4.5: JMP @#STA.A
  474                                    ;
  475  001226' 016565  000004  000004  CKDATA: MOV 4(R5),4(R5)
  476  001234' 012703  000200                   MOV #200,R3
  477  001240' 013701  000204'                  MOV @#CMEMBF,R1         ;INIT PNT
  478  001244' 020221                    1$:    CMP R2,(R1)+            ;DATA CORRECT?
  479  001246' 001402                           BEQ 2$                  ;BR IF YES.
  480  001250' 004767  000006                   JSR PC,RPTERR           ;ERROR.
  481  001254' 005303                    2$:    DEC R3                  ;DONE?
  482  001256' 001372                           BNE 1$                  ;BR IF NO.
  483  001260' 000207                           RTS PC
  484                                    ;
  485  001262' 005037  000050'          RPTERR: CLR @#CSRA
  486  001266' 005741                           TST -(R1)
  487  001270' 010137  000052'                  MOV R1,@#BADR           ;ADDR OF LOC TESTED.
  488  001274' 010137  000054'                  MOV R1,@#WASADR         ;  "   "  "   "
  489  001300' 010237  000056'                  MOV R2,@#ASB            ;STORE GOOD DATA
```



```
  490  001304' 012137  000060'                  MOV (R1)+,@#AWAS        ;STORE BAD DATA
  491                                           ;*****************************************************************
  (1)  001310' 104405  000000'                  DATER.,BEGIN            ;DATA ERROR!!!
  (1)                                           ;*****************************************************************
  492  001314' 000207                           RTS PC
  493                                    ;
  494  001316' 016565  000004  000004  WRDATA: MOV 4(R5),4(R5)
  495  001324' 012703  000200                   MOV #200,R3
  496  001330' 013701  000204'                  MOV @#CMEMBF,R1         ;INIT PNT
  497  001334' 010221                    1$:    MOV R2,(R1)+            ;WRITE DATA.
  498  001336' 005303                           DEC R3                  ;DONE?
  499  001340' 001375                           BNE 1$                  ;BR IF NO.
  500  001342' 000207                           RTS PC
  501  001344' 016537  000014  000166' PEND:   MOV 14(R5),@#TMP
  502  001352' 042737  037477  000166'          BIC #37477,@#TMP
  503  001360' 022737  140300  000166'          CMP #140300,@#TMP       ;ANY API BREAKS PENDING?
  504  001366' 001402                           BEQ PEN.1               ;BR IF NO.
  505  001370' 104403  000000'                  END.,BEGIN              ;UNEXPECTED API BREAK PENDING
  506  001374' 000207                    PEN.1: RTS PC
  507          000001                           .END
```

```
ACSR      000052R     ADDR      000006R     APIDN     000176R     APIDNE    000200R
ASB       000056R     ASTAT     000054R     AWAS      000060R     BDCNV   = ****** G
BEGIN     000000R     BIT0    = 000001      BIT1    = 000002      BIT10   = 002000
BIT11   = 004000      BIT12   = 010000      BIT13   = 020000      BIT14   = 040000
BIT15   = 100000      BIT2    = 000004      BIT3    = 000010      BIT4    = 000020
BIT5    = 000040      BIT6    = 000100      BIT7    = 000200      BIT8    = 000400
BIT9    = 001000      BREAK,  = 104407      BR1       000012R     BR2       000013R
CKDATA    001226R     CMEMBF    000204R     COMM      000170R     CSRA      000050R
DATER,  = 104405      DVID1     000014R     EABITS  = ****** G    ENDPS,  = 104402
END,    = 104403      ERRCNT    000030R     ERRN,   = 104410      ERROR,  = 104404
EXIT,   = 104400      HICORE  = ****** G    INIT      000022R     LOCORE  = ****** G
MODNAM    000000R     MODSP     000162R     MSGN,   = 104411      MSG,    = 104406
OACNV   = ****** G    OPEN    = 000000      PASCNT    000026R     PC      =%000007
PEND      001344R     PEN,1     001374R     PIRQ,   = 000004      POPSP   = 005726
POPSP2  = 022626      PRTY    = 000000      PRTY0   = 000000      PRTY1   = 000040
PRTY2   = 000100      PRTY3   = 000140      PRTY4   = 000200      PRTY5   = 000240
PRTY6   = 000300      PRTY7   = 000340      PS      = 177776      PSW     = 177776
PUSH    = 005746      PUSH2   = 024646      QUE,    = 104401      ROU0      000444R
ROU0,5    000506R     ROU0,6    000520R     ROU0,7    000536R     ROU1      000542R
ROU1,5    000604R     ROU1,6    000616R     ROU2      000630R     ROU2,4    000672R
ROU2,5    000704R     ROU2,6    000722R     ROU3      000726R     ROU3,4    000770R
ROU3,5    001002R     ROU3,6    001020R     ROU4      001024R     ROU4,1    001030R
ROU4,2    001074R     ROU4,3    001154R     ROU4,4    001166R     ROU4,5    001222R
RPTERR    001262R     R0      =%000000      R1      =%000001      R2      =%000002
R3      =%000003      R4      =%000004      R5      =%000005      R6      =%000006
R7      =%000007      SBADR     000052R     SP      =%000006      SPOINT    000024R
SPSIZ   = 000040      SR1       000016R     START     000206R     STAT      000020R
STA,A     000212R     STA,B     000276R     STA,C     000304R     STA,D     000402R
STA,2     000406R     STA,3     000414R     STA,4     000422R     ST,A      000320R
SVR0      000032R     SVR1      000034R     SVR2      000036R     SVR3      000040R
SVR4      000042R     SVR5      000044R     SVR6      000046R     TCBP      000162R
TCBPFL    000172R     TEMP      000164R     TMP       000166R     TPX     = 000000
TRAPX   = 000012      VECTOR    000010R     WAIT      001114R     WASADR    000054R
WA.1      001140R     WA.2      001150R     WBUF    = ****** G    WRDATA    001316R
.       = 001376R

          001376
```

ERRORS DETECTED: 0


```
*XUCAB,XUCAB,PRT_DCXCOM,P11,XUCAB,P11
RUN-TIME:  2 3 0 SECONDS
CORE USED:  4K
```

```
                                .REM_
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DXRKA-A-D
PRODUCT NAME:	XRKAA-DEC/X11 RK11 MODULE
DATE:	FEB 15, 1973
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR(S):	A. COSSETTE

1. ABSTRACT

THIS ROUTINE PERFORMS A WRITE FOUR SECTORS,READ ONE,AND
A WRITE CHECK ON FOUR SECTORS,LOOKS FOR MORE DRIVES
SELECTED AND IF ANY PERFORMS THE SAME ROUTINE AS ABOVE
THEN ADVANCES TO THE NEXT FOUR SECTORS AND SO ON UNTILL
THE DRIVES SELECTED HAVE ALL BEEN EXERCISED.

2. REQUIREMENTS

HARDWARE: RK11 DISK CONTROL AND ONE RK02 OR ONE RK03
STORAGE: XRKA MODULE REQUIRES 754 WORDS OF STORAGE

3. PASS DEFINITION

ONE PASS OF XRKA MODULE CONSIST OF WRITE,
READ 1/4TH OF WRITTEN DATA
AND WRITE CHECK FOR ALL SELECTED RK02 OR RK03
ON LINE

4. EXECUTION TIME

XRKA RUNNING ALONE WITH ONE RK03 ON THE PDP-11/05
TAKES APPROXIMATELY 2 MIN. 38 SEC.FOR ONE PASS

5. CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 177400, VECTOR: 220, BR1:5, DEVCNT:1

```
                                        REQUIRED PARAMETERS:  NONE


                                6.      DEVICE/OPTION SETUP
                                        ------------------

                                        A.  EACH DISK DRIVE MUST HAVE A SCRATCH PACK.
                                        B.  SWITCH SLECTED DRIVES ON LINE


                                7.      MODULE OPERATION
                                        ----------------

                                        TEST SEQUENCE:

                                        A.  SELECT A DISK DRIVE
                                        B.  WRITE 1024 READ 256 AND WRITE CHECK
                                            1024 WORDS
                                        C.  DO OTHER DRIVES ON LINE
                                        D.  ADVNCE SECTOR COUNTER BY FOUR
                                            AND CONTINUE WITH A&C&D
                                        E.  CHECK FOR HIGH CYLINDER AND HIGH DRIVE
                                            SELECTED
                                        F.  CLEAR DEVICE, END PASSES

                                        NOTES:  XRKA DOES NOT USE DRIVE 0 IF LOAD MEDIUM


                                8.      OPERATION OPTIONS
                                        -----------------

                                        MODULE LOCATION DVID1 MAY BE CONFIGURED OR CHANGED TO
                                        INDICATE ANY DRIVE OR DRIVES OR ALL DISK DRIVES
                                        UP TO FOUR (4).


                                9.      NON STARDARD PRINTOUTS
                                        ----------------------

                                        ALL PRINTOUTS STANDARD, REFENCE DEC/X11 DOCUMENTATION
```



```
                                ;
                                ;RK11 DISK CONTROLLER INITIALIZATION

                                        IOMOD   <XRKAA >,177400,220,5
000000'                                 MODULE  140000,XRKAA ,177400,220,5,,
000000'                                 ,TITLE  XRKAA DEC/X11 SYSTEM EXERCISER MODULE
                                        ,LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 051130  040513  020101  MODNAM: .ASCII  /XRKAA /          ;MODULE NAME.
000006' 177400                  ADDR:   177400+0                ;1ST DEVICE ADDR.
000010' 000220                  VECTOR: 220+0                   ;1ST DEVICE VECTOR.
000012'    240                  BR1:    .BYTE   PRTY5+0         ;1ST BR LEVEL.
000013'    000                  BR2:    .BYTE   PRTY+0          ;2ND BR LEVEL.
000014' 000001                  DVID1:  +1                      ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
                                ;*****************************************************************************
000020' 140000                  STAT:   140000                  ;STATUS WORD.
000022' 000162'                 INIT:   START                   ;MODULE START ADDR.
000024' 000162'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000026' 000000                  PASCNT: 0                       ;PASS COUNTER.
000030' 000000                  ERRCNT: 0                       ;ERROR COUNTER.
000032' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000034' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000036' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000040' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000042' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000044' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000046' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000050' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000052'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000052' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000054'                         WASADR:                         ;ADDR OF BAD DATA, OR
000054' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000056' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000060' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000162'                         MODSP:
                                ;*****************************************************************************
000162' 016700  177620          START:  MOV     ADDR,R0         ;R0=177400
000166' 010067  001374                  MOV     R0,RKDSR                ;RKDSR=177400
000172' 005720                          TST     (R0)+                   ;+2=>R0
000174' 010067  001346                  MOV     R0,RKDER                ;RKDER=177402
000200' 005720                          TST     (R0)+                   ;+2=>R0
000202' 010067  177642                  MOV     R0,CSRA                 ;CSRA CONTAINS 177404
000206' 010067  001350                  MOV     R0,RKCSR                ;RKCSR=177404
000212' 105720                          TSTB    (R0)+                   ;INCR. BY 1
000214' 010067  001344                  MOV     R0,RKCSRH               ;RKCSRH=177405
000220' 105720                          TSTB    (R0)+                   ;INCR. BY 1
000222' 010067  001330                  MOV     R0,RKWC                 ;RKWC=177406
```

```
377  000226' 005720                  TST     (R0)+                   ;+2=>R0
378  000230' 010067  001324          MOV     R0,RKBAR                ;RKBAR=177410
379  000234' 005720                  TST     (R0)+                   ;+2=>R0
380  000236' 010067  001310          MOV     R0,RKDAR                ;RKDAR=177412
381  000242' 105720                  TSTB    (R0)+                   ;INCR. BY 1
382  000244' 010067  001304          MOV     R0,RKDAH                ;RKDAH=177413
383  000250' 105720                  TSTB    (R0)+                   ;INCR. BY 1
384  000252' 005720                  TST     (R0)+                   ;+2=>R0
385  000254' 010067  001270          MOV     R0,RKDBR                ;RKDBR=177416
386  000260' 000257                  CCC                             ;CLEAR CONDITION CODES
387  000262' 016700  177522          MOV     VECTOR,R0
388  000266' 012720  000610'         MOV     #RK11,(0)+
389  000272' 016720  177514          MOV     BR1,(0)+
390  000276' 016767  000000G 001270  MOV     WBUF,TWBUF              ;DATA BUFFER STORAGE
391  000304' 005067  001324          CLR     FSTFLG                  ;
392  000310' 005067  001314          CLR     DRVS                    ;
393  000314' 005067  001306          CLR     DSKONL                  ;
394  000320' 005067  001244          CLR     RKDRV
395  000324' 005067  001242          CLR     DVIDA
396  000330' 005067  001276          CLR     DRVCNT                  ;
397  000334' 012767  000031  001240  MOV     #31,RKLMT               ;
398  000342' 005000          DRV:    CLR     R0                      ;OFFSET TO ACTIVE DISK DRIVE
399                                                                  ;SELECTED
400  000344' 132737  000002  000041  BITB    #2,@#41                 ;LOAD MEDIUM TEST
401  000352' 001410                  BEQ     1$                      ;BRANCH IF NOT LOAD MEDIUM
402  000354' 042767  000001  177432  BIC     #1,DVID1                ;CLR DEVICE 0 IF SELECTED
403  000362' 005767  177426          TST     DVID1                   ;ARE THERE MORE DEVICES
404  000366' 001002                  BNE     1$                      ;BRANCH IF THERE IS MORE
405  000370' 104403  000000'         END,,BEGIN                      ;
406  000374' 162767  020000  001226  1$:     SUB     #20000,DRVS     ;
407  000402' 016767  177406  001162          MOV     DVID1,DVIDA     ;GET WORK REG, FOR DEVICE(S)
408  000410' 042767  177760  001154          BIC     #177760,DVIDA   ;MASK OFF TO 4 DEVICE SELECTIONS
409  000416' 106267  001150          4$:     ASRB    DVIDA           ;SHIFT RIGHT FOR TEST
410  000422' 103410                  BCS     2$                      ;
411  000424' 005720                  TST     (R0)+                   ;UPDATE OFFSET
412  000426' 062767  020000  001174  ADD     #20000,DRVS             ;UPDATE DRIVE SELECT ADDRESS
413  000434' 005767  001132          TST     DVIDA                   ;ARE THERE MORE DEVICES SELECTED
414  000440' 001426                  BEQ     3$                      ;
415  000442' 000765                  BR      4$                      ;
416  000444' 005267  001156          2$:     INC     DSKONL          ;COUNTING HOW MANY DRIVES SELECTED
417  000450' 062767  020000  001152  ADD     #20000,DRVS             ;UPDATE
418  000456' 016760  001146  001616' MOV     DRVS,ACDSK(R0)          ;LOAD DRIVE SELECTED ADDRESS INTO
419                                                                  ;STORAGE REG. (ACTIVE DISK)
420  000464' 005767  001144          TST     FSTFLG                  ;
421  000470' 001005                  BNE     5$                      ;
422  000472' 016067  001616' 001070  MOV     ACDSK(R0),RKDRV         ;
423  000500' 005167  001130          COM     FSTFLG                  ;
424  000504' 005767  001062          5$:     TST     DVIDA           ;ARE WE DONE
425  000510' 001402                  BEQ     3$                      ;YES
426  000512' 005720                  TST     (R0)+                   ;
427  000514' 000740                  BR      4$                      ;NO GO BACK & DO MORE
428  000516' 000367  001106          3$:     SWAB    DRVS            ;
429  000522' 066767  001102  001052  ADD     DRVS,RKLMT              ;GET ADDRESS OF HIGH DRIVE SELECTED
430  000530' 000367  001074          SWAB    DRVS                    ;REINSTAT DRVS.
```



```
431  000534' 016767  177254  001030          MOV     DVID1,DVIDA     ;
432  000542' 042777  000016  001012  B1$:     BIC     #16,@RKCSR              ;CLEAR DSK DRIVE
433  000550' 012767  043503  000764          MOV     #43503,RKFUNCTION ;SET UP DSK CONTROL
434  000556' 016777  001012  000774          MOV     TWBUF,@RKBAR
435  000564' 016777  000754  000764          MOV     RKWORDCT,@RKWC  ;LENGTH OF TRANSFER
436  000572' 016777  000772  000752          MOV     RKDRV,@RKDAR    ;INITIALIZE OF TRANSFER
437  000600' 012777  000103  000754          MOV     #103,@RKCSR     ;GO MAN
438  000606' 104400                          EXIT.                   ;RETURN TO MONITOR.
439
440                                  ;RK11 DISK TEST INTERRUPT LEVEL 5, 1024 WORD TRANSFERS
441
442
443  000610'                         RK11:
(1)                                      ;-------------------------------------------------------------------
(1)  000610' 000004  000616' 000000'     PIRQ,,SUBSER,BEGIN      ;QUEUE REQUEST TO CONTINUE AT SUBSER
(1)                                      ;-------------------------------------------------------------------
444
445  000616' 042777  000100  000736  SUBSER: BIC     #100,@RKCSR     ;CLEAR INTR ENABLE
446  000624' 105777  000732          TSTB    @RKCSR          ;INTERRUPT VECTOR POINTS HERE
447  000630' 100402                  BMI     2$              ;
448  000632' 000167  000510          JMP     RKER1           ;
449  000636' 005777  000720          2$:     TST     @RKCSR
450  000642' 100002                  BPL     1$              ;
451  000644' 000167  000530          JMP     RKER2           ;
452  000650' 122777  000120  000674  1$:     CMPB    #120,@RKDAR     ;DISK AT UPPER LIMIT?
453  000656' 001020                  BNE     WHO             ;NO
454  000660' 126777  000716  000666          CMPB    RKLMT,@RKDAH
455  000666' 001014                  BNE     WHO
456  000670' 000167  000632          JMP     ENPASS
457  000674' 016777  000670  000650  RKSTART:MOV     RKDRV,@RKDAR    ;INITIALIZE DISK-DAR,DAE
458  000702' 016777  000636  000646          MOV     RKWORDCT,@RKWC  ;LENGTH OF TRANSFER
459  000710' 116777  000626  000644          MOVB    RKFUNCTION,@RKCSR ;WRITE OR WRITE CHECK TO DISK
460  000716' 104400                          EXIT.                   ;RETURN TO MONITOR.
461
462
463
464  000720' 017767  000636  000656  WHO:     MOV     @RKCSR,FNCWAS   ;GET LAST FUNCTION CMMD.
465  000726' 042767  177761  000650          BIC     #177761,FNCWAS  ;MASK OF OTHER CONTROL BITS
466  000734' 026767  000644  000644          CMP     FNCWAS,WFNC     ;WAS LAST XFER A WRITE CMMD.
467  000742' 001410                          BEQ     RSTRT           ;YES BRANCH OFF & DO A READ
468  000744' 026767  000634  000636          CMP     FNCWAS,RFNC     ;WAS LAST XFER A READ CMMD.
469  000752' 001422                          BEQ     RCHCK           ;YES GO CHECK READ DATA
470  000754' 026767  000624  000630          CMP     FNCWAS,WCFNC    ;WAS LAS XFER A WRITE CHCK
471  000762' 001446                          BEQ     WSTRT           ;YES BRANCH OFF & SET
472                                                                  ;UP FOR NEXT PASS
473  000764' 042767  000016  000550  RSTRT:  BIC     #16,RKFUNCTION  ;GET OLD FUNCTION CMMD. OUT
474  000772' 012767  177400  000544          MOV     #-256,,RKWORDCT ;
475  001000' 056767  000604  000534          BIS     RFNC,RKFUNCTION ;SET NEXT FUNCTION CMMD.
476  001006' 012777  001652' 000544          MOV     #RBUF,@RKBAR    ;SET DISK CURRENT ADDRESS
477  001014' 000167  177654                  JMP     RKSTART         ;RETURN TO KICKOFF NEXT PASS
478
479
480  001020' 012700  001652'  RCHCK:  MOV     #RBUF,R0        ;GET STARTING ADDRESS OF
481                                                           ;READ BUFF AREA
```

```
 482  001024' 012702  000400                    MOV     #256.,R2                ;R2=WORDS TO BE CHCKED
 483  001030' 016701  000540                    MOV     TWBUF,R1                ;R1=WRITE BUF AREA
 484  001034' 022021                    CK1:    CMP     (R0)+,(R1)+             ;COMPARE DATA WITH BUFFER
 485  001036' 001175                            BNE     DATER                   ;GO REPORT ERROR
 486  001040' 005302                            DEC     R2                      ;DEC NO. WORDS CHCK'ED.
 487  001042' 001374                            BNE     CK1                     ;NOT DONE RETURN & CHECK
 488                                                                            ;MORE DATA
 489  001044' 012767  176000  000472  WCSTRT: MOV     #176000,RKWORDCT        ;
 490  001052' 042767  000016  000462            BIC     #16,RKFUNCTION ;MASK OFF OLD FUNC. CMMD.
 491  001060' 056767  000526  000454            BIS     WCFNC,RKFUNCTION ;SET WRITE CHCK MODE
 492  001066' 016777  000502  000464            MOV     TWBUF,@RKBAR            ;DISK HAS WBUF START ADDRESS
 493  001074' 000167  177574                    JMP     RKSTART                 ;RETURN TO KICKOFF NEXT PASS
 494
 495  001100' 042767  000016  000434  WSTRT:  BIC     #16,RKFUNCTION ;MASK OFF OLD FUNC. CMMD.
 496  001106' 056767  000474  000426            BIS     WFNC,RKFUNCTION ;SET WRITE MODE
 497  001114' 016767  000000G 000452            MOV     WBUF,TWBUF              ;GET NEW WRITE BUFF AREA
 498  001122' 016777  000446  000430            MOV     TWBUF,@RKBAR            ;GIVE DISK NEW WRITE BUF
 499  001130' 000400                            BR      UPDATE                  ;GO SET UP NEW DISK
 500                                                                            ;ADDRES
 501
 502
 503
 504
 505  001132' 042777  000016  000422  UPDATE: BIC     #16,@RKCSR              ;CLEAR DRIVE CONTROL LOGIC
 506  001140' 117767  000406  000446            MOVB    @RKDAR,EXAM             ;GETTING SECTOR COUNTER
 507  001146' 142767  000360  000440            BICB    #360,EXAM               ;MASK OFF GARBARGE
 508  001154' 122767  000000  000432            CMPB    #0,EXAM                 ;ARE ALL 3 BLOCK DONE(12 SECT.)
 509  001162' 001064                            BNE     1$                      ;BRANCH IF NOT
 510  001164' 042777  000017  000360            BIC     #17,@RKDAR              ;REINITIALIZE DISK SECTION COUNTER
 511  001172' 016700  000434          4$:       MOV     DRVCNT,R0               ;GET OFFSET VALUE => R0
 512  001176' 106267  000370                    ASRB    DVIDA                   ;
 513  001202' 103045                            BCC     3$                      ;
 514  001204' 042767  160017  000356  7$:       BIC     #160017,RKDRV           ;CLEAR DRIVE SELECT BITS
 515  001212' 056067  001616' 000350            BIS     ACDSK(R0),RKDRV         ;NEW DRIVE SELECTED
 516  001220' 062767  000002  000404            ADD     #2,DRVCNT               ;COUNT THIS DRIVE SELECTED
 517  001226' 005767  000340                    TST     DVIDA                   ;
 518  001232' 001043                            BNE     2$                      ;RETURN TO START ANOTHER SEG.
 519  001234' 103442                            BCS     2$                      ;
 520  001236' 005067  000370                    CLR     DRVCNT                  ;UPDATE
 521  001242' 016767  176546  000322            MOV     DVID1,DVIDA             ;
 522  001250' 016700  000356                    MOV     DRVCNT,R0               ;
 523  001254' 106267  000312          5$:       ASRB    DVIDA                   ;
 524  001260' 103402                            BCS     6$                      ;
 525  001262' 122020                            CMPB    (R0)+,(R0)+             ;
 526  001264' 000773                            BR      5$                      ;
 527  001266' 056067  001616' 000274  6$:       BIS     ACDSK(R0),RKDRV         ;
 528  001274' 016767  176514  000270            MOV     DVID1,DVIDA             ;
 529  001302' 005067  000324                    CLR     DRVCNT                  ;
 530  001306' 062767  000020  000254            ADD     #20,RKDRV               ;INCR. TRACK ADDR.
 531  001314' 000412                            BR      2$                      ;
 532  001316' 062767  000002  000306  3$:       ADD     #2,DRVCNT               ;UPDATE
 533  001324' 005767  000242                    TST     DVIDA                   ;
 534  001330' 001725                            BEQ     7$                      ;
 535  001332' 000717                            BR      4$                      ;GO BACK FOR MORE DISK
```



```
 536  001334' 062767  000004  000226  1$:       ADD     #4,RKDRV                ;
 537  001342' 000167  177326          2$:       JMP     RKSTART                 ;
 538
 539
 540
 541
 542
 543
 544
 545
 546
 547
 548
 549
 550  001346' 017767  000210  176476  RKER1:  MOV     @RKCSR,ACSR
 551  001354' 017767  000166  176472            MOV     @RKDER,ASTAT
 552                                            ;*****************************************************************
 (1)  001362' 104404  000000'                   ERROR,,BEGIN            ;RK11 READY NOT UP
 (1)                                            ;*****************************************************************
 553  001366' 042777  000016  000166            BIC     #16,@RKCSR              ;
 554  001374' 000167  177274                    JMP     RKSTART                 ;
 555
 556
 557
 558  001400' 017767  000156  176444  RKER2:  MOV     @RKCSR,ACSR
 559  001406' 017767  000134  176440            MOV     @RKDER,ASTAT
 560                                            ;*****************************************************************
 (1)  001414' 104404  000000'                   ERROR,,BEGIN            ;ERROR FLAG IS UP
 (1)                                            ;*****************************************************************
 561  001420' 042777  000016  000134            BIC     #16,@RKCSR              ;
 562  001426' 000167  177242                    JMP     RKSTART                 ;
 563
 564
 565  001432' 017767  000124  176410  DATER:  MOV     @RKCSR,CSRA             ;CONTROL AND STATUS REG.
 566  001440' 014167  176412                    MOV     -(R1),ASB               ;DATE SHOULD BE
 567  001444' 014067  176410                    MOV     -(R0),AWAS              ;DATA READ WAS
 568  001450' 010167  176376                    MOV     R1,SBADR                ;
 569  001454' 010067  176374                    MOV     R0,WASADR               ;
 570  001460' 004767  001166                    JSR     PC,RSAV                 ;GO SAVE ALL THE REG.
 571                                            ;*****************************************************************
 (1)  001464' 104405  000000'                   DATER,,BEGIN            ;DATA ERROR!!!
 (1)                                            ;*****************************************************************
 572  001470' 004767  001204                    JSR     PC,RGET                 ;RESTORE REG.
 573  001474' 022021                            CMP     (R0)+,(R1)+             ;UPDATE R'S
 574  001476' 005267  000134                    INC     ERCNT                   ;
 575  001502' 022767  000003  000126            CMP     #3,ERCNT                ;
 576  001510' 001402                            BEQ     1$                      ;
 577  001512' 000167  177316                    JMP     CK1                     ;RETURN TO CHCK'ING DATER
 578  001516' 005067  000114          1$:       CLR     ERCNT                   ;
 579  001522' 000167  177316                    JMP     WCSTRT                  ;
 580
 581
 582
 583
```

```
  584  001526' 042777  000016  000026  ENPASS: BIC     #16,@#RKCSR      ;CLEAR DSK DRIVE
  585  001534' 104402  000162' 000000'         ENDPS,,START,BEGIN      ;SIGNAL END OF PASS, RESUME AT START
  586
  587
  588  001542' 000000                  RKFUNCTION:     0               ;DISK COMMAND
  589  001544' 176000                  RKWORDCT:       -1024.          ;LENGTH OF TRANSFER
  590  001546' 000000                  RKDER:  0
  591  001550' 000000                  RKDBR:  0                       ;RK11 DATA BUFFER
  592  001552' 000000                  RKDAR:  0                       ;RK11 DISK ADDRESS REGISTER
  593  001554' 000000                  RKDAH:  0                       ;RK11 HIGH BYTE OF DISK ADDRESS
  594  001556' 000000                  RKWC:   0                       ;RK11 WORD COUNT REGISTER
  595  001560' 000000                  RKBAR:  0                       ;RK11 CURRENT ADDRESS REGISTER
  596  001562' 000000                  RKCSR:  0                       ;RK11 STATUS REGISTER
  597  001564' 000000                  RKCSRH: 0                       ;RK11 HIGH BYTE ADDRESS OF CSR
  598  001566' 000000                  RKDSR:  0                       ;RK11 DRIVE STATUS REGISTER
  599  001570' 000000                  RKDRV:  0                       ;RK11 DRIVE SELECTED FOR TEST
  600  001572' 000000                  DVIDA:  0
  601  001574' 000000                  TWBUF:  0
  602  001576' 000107                  PASSEN: 107
  603  001600' 000000                  DEVAD:  0
  604  001602' 000031                  RKLMT:  31              ;DSK ADDRESS LIMET
  605  001604' 000000                  FNCWAS: OPEN            ;
  606  001606' 000002                  WFNC:   2                       ;
  607  001610' 000004                  RFNC:   4                       ;
  608  001612' 000006                  WCFNC:  6                       ;
  609  001614' 000000                  EXAM:   OPEN                    ;
  610  001616' 000000                  ACDSK:  0                       ;DRIVE SELECTED ADDRESS STORAGE
  611  001620' 000000                          0
  612  001622' 000000                          0
  613  001624' 000000                          0
  614  001626' 000000                  DSKONL: 0                       ;NO. OF ACTIVE DRIVES SELECTED
  615  001630' 000000                  DRVS:   0                       ;HIGHEST DRIVE SELECTED
  616  001632' 000000                  DRVCNT: 0                       ;DRIVE COUNTER FOR COMPARASIONS
  617  001634' 000000                  FSTFLG: 0                       ;FIRST DRIVE SELECTED FLAG
  618  001636' 000000                  ERCNT:  0                       ;
  619
  620  001640' 000000                  XSR0:   0                       ;
  621  001642' 000000                  XSR1:   0                       ;
  622  001644' 000000                  XSR2:   0                       ;
  623  001646' 000000                  XSR3:   0                       ;
  624  001650' 000000                  XSR4:   0                       ;
  625
  626
  627
  628
  629  001652' 000400                  RBUF:   .BLKW   256.            ;
  630
  631
  632
  633
  634  002652' 010067  176762          RSAV:   MOV     R0,XSR0         ;SAVE REG.
  635  002656' 010167  176760                  MOV     R1,XSR1         ;
  636  002662' 010267  176756                  MOV     R2,XSR2         ;
  637  002666' 010367  176754                  MOV     R3,XSR3         ;
```



```
  638  002672' 010467  176752                  MOV     R4,XSR4         ;
  639  002676' 000207                          RTS     PC              ;
  640
  641
  642
  643
  644  002700' 016700  176734          RGET:   MOV     XSR0,R0         ;RESTORE REG.
  645  002704' 016701  176732                  MOV     XSR1,R1         ;
  646  002710' 016702  176730                  MOV     XSR2,R2         ;
  647  002714' 016703  176726                  MOV     XSR3,R3         ;
  648  002720' 016704  176724                  MOV     XSR4,R4         ;
  649  002724' 000207                          RTS     PC              ;
  650
  651
  652          000001                          .END
```

```
ACDSK   001616R     ACSR    000052R     ADDR    000006R     ASR     000056R
ASTAT   000054R     AWAS    000060P     BDCNV = ****** G    BEGIN   000000R
BIT0  = 000001      BIT1  = 000002      BIT10 = 002000      BIT11 = 004000
BIT12 = 010000      BIT13 = 020000      BIT14 = 040000      BIT15 = 100000
BIT2  = 000004      BIT3  = 000010      BIT4  = 000020      BIT5  = 000040
BIT6  = 000100      BIT7  = 000200      BIT8  = 000400      BIT9  = 001000
BREAK, = 104407     BR1     000012R     BR2     000013R     BUF     000542R
CK1     001034R     CSRA    000050R     DATER   001432R     DATER, = 104405
DEVAD   001600R     DRV     000342R     DRVCNT  001632R     DRVS    001630R
DSKCNL  001626R     DVIDA   001572R     DVID1   000014R     FABITS = ****** G
ENDPS, = 104402     END,  = 104403      ENPASS  001526R     ERCNT   001636R
ERRCNT  000030R     ERRN,  = 104410     ERROR, = 104404     EXAD    001614R
EXIT,  = 104400     FNCWAS  001604R     FSTFLG  001634R     HICORE = ****** G
INIT    000022R     LOCORE = ****** G   MODNAM  000000R     MODSP   000162R
MSGN,  = 104411     MSG,   = 104406     OACNV = ****** G    OPEN    000000
PASCNT  000026R     PASSEN  001576R     PC    =%000007      PIRQ,  = 000004
POPSP  = 005726     POPSP2 = 022626     PRTY  = 000000      PRTY0  = 000000
PRTY1  = 000040     PRTY2  = 000100     PRTY3  = 000140     PRTY4  = 000200
PRTY5  = 000240     PRTY6  = 000300     PRTY7  = 000340     PS     = 177776
PSW    = 177776     PUSH   = 005746     PUSH2  = 024646     QUE,   = 104401
RBUF    001652R     RCHCK   001020R     RFNC    001610R     RGET    002700R
RKBAR   001560R     RKCSR   001562R     RKCSRH  001564R     RKDAH   001554R
RKDAR   001552R     RKDBR   001550R     RKDER   001546R     RKDRV   001570R
RKDSR   001566R     RKER1   001346R     RKER2   001400R     RKFUNC  001542R
RKLMT   001602R     RKSTAR  000674R     RKWC    001556R     RKWORD  001544R
RK11    000610R     RSAV    002652R     RSTRT   000764R     R0     =%000000
R1     =%000001     R2     =%000002     R3     =%000003     R4     =%000004
R5     =%000005     R6     =%000006     R7     =%000007     SBADR   000052R
SP     =%000006     SPOINT  000024R     SPSIZ  = 000040     SR1     000016R
START   000162R     STAT    000020R     SUBSER  000616R     SVR0    000032R
SVR1    000034R     SVR2    000036R     SVR3    000040R     SVR4    000042R
SVR5    000044R     SVR6    000046R     TPX    = 000000     TRAPX  = 000012
TWBUF   001574R     UPDATE  001132R     VECTOR  000010R     WASADR  000054R
WBUF   = ****** G    WCFNC   001612R     WCSTRT  001044R     WFNC    001606R
WHO     000720R     WSTRT   001100R     XSR0    001640R     XSR1    001642R
XSR2    001644R     XSR3    001646R     XSR4    001650R     .      = 002726R

        002726
```

ERRORS DETECTED: 0


```
*XRKAB,XRKAB,PRT=DCXCOM.P11,XRKAB.P11
RUN-TIME:  2  4  0 SECONDS
CORE USED:  4K
```

```
    1
  213
                .REM_
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DXLPA-B-D
PRODUCT NAME:	XLPAR-DEC/X11 LP11 MODULE
DATE:	JUL. 27, 1973
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR(S):	R. E. UNDERWOOD

1. ABSTRACT

XLPA EXERCISES THE LP11 PRINTER CONTROL AND AN LP11 PRINTER OF 4 POSSIBLE MODELS. THE BASIC TEST RUNS AN INCREMENTAL TEST PATTERN FILLING 500 LINES WITH ALL POSSIBLE PRINTING CHARACTERS AND THE SPACE.

2. REQUIREMENTS

HARDWARE: LP11 LINE PRINTER CONTROL AND ONE LP11 LINE PRINTER
STORAGE: XLPA MODULE REQUIRES 232 WORDS OF STORAGE

3. PASS DEFINITION

ONE PASS OF XLPA MODULE WRITES AN INCREMENTAL TEST PATTERN FOR 500 FULL LINES.

4. EXECUTION TIME

XLPA RUNNING ALONE WITH A LP11-FA ON THE PDP-11/05 TAKES APPROXMATELY 1.5 MINUTES FOR ONE PASS.

5. CONFIGURATION REQUIRMENTS

DEFAULT PARAMETERS:

DEVADR: 177514, VECTOR:200, BR:4, DEVCNT:1, SR: 0

REQUIRED PARAMETERS: NONE

6. DEVICE/OPTION SETUP

A, LOAD LINE PRINTER PAPER.
B, SWITCH ON LINE

7. MODULE OPERATION

TEST SEQUENCE;

A, INITIALIZE LP11 CONTROL
B, GENERATE INCREMENTING CHARACTER PATTERN AND PRINT.
C, DOING STEP B UNTIL ALL 500 LINES HAVE BEEN PRINTED.

8. OPERATION OPTIONS

THIS MODULE MAY BE USED FOR 4 MODELS OF THE LP11 BY SETTING THE SR1 OPTION IN THE CONFIGURE MODE.

A,	SR1: 0	LP11-FA	80 COL, 64 CHAR.
B,	SR1: 1	LP11-HA	80 COL, 96 CHAR.
C,	SR1: 2	LP11-JA	132 COL, 64 CHAR.
D,	SR1: 3	LP11-KA	132 COL, 96 CHAR.

9. NON STANDARD PRINTOUT

ALL PRINTOUTS STANDARD, REFERENCE DEC/X11 DOCUMENTATION


```
000000' IOMOD<XLPAB >,177514,200,4
000000'         MODULE  140000,XLPAB ,177514,200,4,,
                .TITLE  XLPAB DEC/X11 SYSTEM EXERCISER MODULE
                        .LIST   BIN
                ;*****************************************************************
000000'         BEGIN:
000000' 046130  040520  020102  MODNAM: .ASCII  /XLPAB /          ;MODULE NAME.
000006' 177514                  ADDR:   177514+0                 ;1ST DEVICE ADDR.
000010' 000200                  VECTOR: 200+0                    ;1ST DEVICE VECTOR.
000012'    200                  BR1:    .BYTE   PRTY4+0          ;1ST BR LEVEL.
000013'    000                  BR2:    .BYTE   PRTY+0           ;2ND BR LEVEL.
000014' 000001                  DVID1:  +1                       ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                     ;SWITCH REGISTER 1
                ;*****************************************************************
000020' 140000                  STAT:   140000                   ;STATUS WORD.
000022' 000162'                 INIT:   START                    ;MODULE START ADDR.
000024' 000162'                 SPOINT: MODSP                    ;MODULE STACK POINTER.
000026' 000000                  PASCNT: 0                        ;PASS COUNTER.
000030' 000000                  ERRCNT: 0                        ;ERROR COUNTER.
000032' 000000                  SVR0:   OPEN                     ;LOC TO SAVE R0.
000034' 000000                  SVR1:   OPEN                     ;LOC TO SAVE R1.
000036' 000000                  SVR2:   OPEN                     ;LOC TO SAVE R2.
000040' 000000                  SVR3:   OPEN                     ;LOC TO SAVE R3.
000042' 000000                  SVR4:   OPEN                     ;LOC TO SAVE R4.
000044' 000000                  SVR5:   OPEN                     ;LOC TO SAVE R5.
000046' 000000                  SVR6:   OPEN                     ;LOC TO SAVE R6.
000050' 000000                  CSRA:   OPEN                     ;ADDR OF CURRENT CSR.
000052'                         SBADR:                           ;ADDR OF GOOD DATA, OR
000052' 000000                  ACSR:   OPEN                     ;CONTENTS OF CSR.
000054'                         WASADR:                          ;ADDR OF BAD DATA, OR
000054' 000000                  ASTAT:  OPEN                     ;STATUS REG CONTENTS.
000056' 000000                  ASR:    OPEN                     ;EXPECTED DATA.
000060' 000000                  AWAS:   OPEN                     ;ACTUAL DATA.
                                .REPT   SPSIZ                    ;MODULE STACK STARTS HERE.
                                .NLIST
                                .WORD   0
                                .LIST
                                .ENDR
000162'                         MODSP:
                ;*****************************************************************
352 000162' 016700  177620      START:  MOV     ADDR,R0
353 000166' 010067  177656              MOV     R0,CSRA
354 000172' 010067  000500              MOV     R0,LPCS
355 000176' 005720                      TST     (0)+
356 000200' 010067  000474              MOV     R0,LPDB
357 000204' 016700  177600              MOV     VECTOR,R0
358 000210' 012720  000340'             MOV     #INTER,(0)+
359 000214' 016710  177572              MOV     BR1,(0)
360 000220' 005067  000460              CLR     LINCNT          ; INITIALIZE
361 000224' 005067  000456              CLR     CHACNT
362 000230' 012767  000036  000454      MOV     #36,FRST
363 000236' 012767  000120  000450      MOV     #80.,COLUMN
```

```
  364  000244' 012767  000140  000444          MOV     #140,TOP
  365  000252' 012767  000137  000440          MOV     #137,TOP1
  366  000260' 032767  000001  177530          BIT     #1,SR1          ; 25 CHARACTER SET ?
  367  000266' 001406                          BEQ     1$              ; NO, CONTINUE
  368  000270' 012767  000177  000422          MOV     #177,TOP1       ; YES, FIX COUNTERS
  369  000276' 012767  000200  000412          MOV     #200,TOP
  370  000304' 032767  000002  177504  1$:     BIT     #2,SR1          ; 132 COLUMNS ?
  371  000312' 001403                          BEQ     2$              ; NO, CONTINUE
  372  000314' 012767  000204  000372          MOV     #132.,COLUMN    ; YES, FIX COUNTER
  373  000322' 012767  000384' 000352  2$:     MOV     #PAGE,NEXT      ; PICK FIRST ENTRY POINT
  374  000330' 012777  000100  000340          MOV     #100,@LPCS      ; SET INTERUPT ENABLE
  375  000336' 104400                          EXIT.                   ;RETURN TO MONITOR.
  376  000340'                         INTER:
  (1)                                         ;-------------------------------------------------------------
  (1)  000340' 000004  000346' 000000'        PIRQ,,PROCED,BEGIN      ;QUEUE REQUEST TO CONTINUE AT PROCED
  (1)                                         ;-------------------------------------------------------------
  377                                     ;
  378  000346' 005777  000324          PROCED: TST     @LPCS           ; IS ERROR BIT SET ?
  379  000352' 100002                          BPL     1$              ; NO, CONTINUE
  380  000354' 000167  000146                  JMP     WAIT            ; YES, GO TO WAIT
  381  000360' 000177  000316          1$:     JMP     @NEXT           ; GO TO NEXT ENTRY POINT
  382                                     ;
  383  000364' 012767  000402' 000310  PAGE:   MOV     #LINE,NEXT      ; PICK NEXT ENTRY POINT
  384  000372' 012777  000014  000300          MOV     #14,@LPDB       ; TOP OF FORM
  385  000400' 104400                          EXIT.                   ;RETURN TO MONITOR.
  386                                     ;   ------------------------
  387  000402' 026727  000276  000764  LINE:   CMP     LINCNT,#500.    ; DONE THIS PASS ?
  388  000410' 001524                          BEQ     FINI            ; YES, GO TO FINI
  389  000412' 005267  000266                  INC     LINCNT          ; NO, PRINT ANOTHER LINE
  390  000416' 005267  000270                  INC     FRST            ; GET STARTING CHARACTER
  391  000422' 026767  000272  000262          CMP     TOP1,FRST       ; OUT OF CHARACTERS ?
  392  000430' 001003                          BNE     1$              ; NO, CONTINUE
  393  000432' 012767  000037  000252          MOV     #37,FRST        ; YES, START OVER
  394  000440' 016767  000246  000242  1$:     MOV     FRST,CHAR       ; PICK FIRST CHARACTER
  395  000446' 012767  000454' 000226          MOV     #PRNT,NEXT      ; PICK NEXT ENTRY POINT
  396                                     ;   ------------------------
  397  000454' 026767  000226  000232  PRNT:   CMP     CHACNT,COLUMN   ; DONE THIS LINE ?
  398  000462' 001410                          BEQ     DUN             ; YES, GO TO DUN
  399  000464' 005267  000216                  INC     CHACNT          ; NO, GO AGAIN
  400  000470' 004567  000142                  JSR     5,CARE          ; GET NEXT CHARACTER
  401  000474' 016777  000210  000176          MOV     CHAR,@LPDB      ; PRINT A CHARACTER
  402  000502' 104400                          EXIT.                   ;RETURN TO MONITOR.
  403                                     ;   ------------------------
  404  000504' 012767  000402' 000170  DUN:    MOV     #LINE,NEXT      ; PICK NEXT ENTRY POINT
  405  000512' 005067  000170                  CLR     CHACNT          ; CLEAR CHARACTER COUNTER
  406  000516' 012777  000012  000154          MOV     #12,@LPDB       ; <CR> <LF>
  407  000524' 104400                          EXIT.                   ;RETURN TO MONITOR.
  408                                     ;   ------------------------
  409  000526' 042777  000100  000142  WAIT:   BIC     #100,@LPCS      ; CLEAR INTERUPT ENABLE
  410  000534' 016767  000136  177306          MOV     LPCS,CSRA       ; SAVE LPCS
  411  000542' 017767  000130  177302          MOV     @LPCS,ACSR      ; SAVE (LPCS)
  412                                         ;*************************************************************
  (1)  000550' 104404  000000'                ERROR,,BEGIN            ;PRINTER ERROR CONDITION ENCOUNTERED ... WAITING
  (1)                                         ;*************************************************************
```



```
  413  000554' 012701  177777                  MOV     #177777,R1      ; SET CLOCK COUNTER
  414  000560'                         1$:
  (1)  000560' 104407  000000'                BREAK,,BEGIN            ;TEMPORARY RETURN TO MONITOR.
  415  000564' 005777  000106                  TST     @LPCS           ; ERROR BIT STILL SET ?
  416  000570' 100016                          BPL     4$              ; NO, GET OUT
  417  000572' 005701                          TST     R1              ; YES, WAIT SOME MORE ?
  418  000574' 001402                          BEQ     2$              ; NO, TIME OUT
  419  000576' 005301                          DEC     R1              ; YES, MORE TIME LEFT
  420  000600' 000767                          BR      1$              ; GO WAIT SOME MORE
  421  000602' 016767  000070  177240  2$:     MOV     LPCS,CSRA       ; SAVE LPCS
  422  000610' 017767  000062  177234          MOV     @LPCS,ACSR      ; SAVE (LPCS)
  423                                         ;*************************************************************
  (1)  000616' 104404  000000'                ERROR,,BEGIN            ;PRINTER ERROR CONDITION STILL EXISTS ... BYE
  (1)                                         ;*************************************************************
  424  000622' 104403  000000'                END,,BEGIN              ;
  425  000626' 052777  000100  000042  4$:     BIS     #100,@LPCS      ; SET INTERUPT ENABLE
  426  000634' 104400                          EXIT.                   ;RETURN TO MONITOR.
  427                                     ;   ------------------------
  428  000636' 005267  000046          CARE:   INC     CHAR            ; GET NEXT CHARACTER TO BE PRINTED
  429  000642' 026767  000050  000040          CMP     TOP,CHAR        ; OUT OF CHARACTERS ?
  430  000650' 001003                          BNE     1$              ; NO, CONTINUE
  431  000652' 012767  000040  000030          MOV     #40,CHAR        ; YES, START OVER WITH FIRST CHARACTER
  432  000660' 000205                  1$:     RTS     R5              ; RETURN
  433                                     ;   ------------------------
  434  000662' 042777  000100  000006  FINI:   BIC     #100,@LPCS      ; CLEAR INTERUPT ENABLE
  435  000670' 104402  000162' 000000'         ENDPS,,START,BEGIN      ;SIGNAL END OF PASS, RESUME AT START
  436                                     ;   ------------------------
  437  000676' 000000                  LPCS:   0                       ; LINE PRINTER CONTROL STATUS REG.
  438  000700' 000000                  LPDB:   0                       ; LINE PRINTER DATA BUFFER REG.
  439  000702' 000000                  NEXT:   0                       ; CONTAINS NEXT ENTRY POINT
  440  000704' 000000                  LINCNT: 0                       ; CONTAINS NUMBER OF LINES PRINTED
  441  000706' 000000                  CHACNT: 0                       ; CONTAINS NUMBER OF CHARACTERS PRINTED
  442  000710' 000000                  CHAR:   0                       ; CONTAINS CHARACTER TO BE PRINTED
  443  000712' 000000                  FRST:   0                       ; 1ST CHARACTER OF LINE TO BE PRINTED
  444  000714' 000000                  COLUMN: 0                       ; NUMBER OF COLUMNS TO BE PRINTED
  445  000716' 000000                  TOP:    0                       ; LAST CHARACTER TO BE PRINTED
  446  000720' 000000                  TOP1:   0                       ; LAST CHARACTER TO BE PRINTED
  447
  448          000001                          .END
```

```
ACSR     000052R    ADDR     000006R    ASB      000056R    ASTAT    000054R
AWAS     000060R    BDCNV  = ****** G   BEGIN    000000R    BIT0   = 000001
BIT1   = 000002     BIT10  = 002000     BIT11  = 004000     BIT12  = 010000
BIT13  = 020000     BIT14  = 040000     BIT15  = 100000     BIT2   = 000004
BIT3   = 000010     BIT4   = 000020     BIT5   = 000040     BIT6   = 000100
BIT7   = 000200     BIT8   = 000400     BIT9   = 001000     BREAK, = 104407
BR1      000012R    BR2      000013R    CARE     000636R    CHACNT   000706R
CHAR     000710R    COLUMN   000714R    CSRA     000050R    DATER, = 104405
DUN      000504R    DVID1    000014R    FABITS = ****** G   ENDPS, = 104402
END,   = 104403     ERRCNT   000030R    ERRN,  = 104410     ERROR, = 104404
EXIT,  = 104400     FINI     000662R    FRST     000712R    HICORE = ****** G
INIT     000022R    INTER    000340R    LINCNT   000704R    LINE     000402R
LOCORE = ****** G   LPCS     000676R    LPDB     000700R    MODNAM   000000R
MODSP    000162R    MSGN,  = 104411     MSG,   = 104406     NEXT     000702R
OACNV  = ****** G   OPEN   = 000000     PAGE     000364R    PASCNT   000026R
PC     =%000007     PIRQ,  = 000004     POPSP  = 005726     POPSP2 = 022626
PRNT     000454R    PROCED   000346R    PRTY   = 000000     PRTY0  = 000000
PRTY1  = 000040     PRTY2  = 000100     PRTY3  = 000140     PRTY4  = 000200
PRTY5  = 000240     PRTY6  = 000300     PRTY7  = 000340     PS     = 177776
PSW    = 177776     PUSH   = 005746     PUSH2  = 024646     QUE,   = 104401
R0     =%000000     R1     =%000001     R2     =%000002     R3     =%000003
R4     =%000004     R5     =%000005     R6     =%000006     R7     =%000007
SBADR    000052R    SP     =%000006     SPOINT   000024R    SPSIZ  = 000040
SR1      000016R    START    000162R    STAT     000020R    SVR0     000032R
SVR1     000034R    SVR2     000036R    SVR3     000040R    SVR4     000042R
SVR5     000044R    SVR6     000046R    TOP      000716R    TOP1     000720R
TPX    = 000000     TRAPX  = 000012     VECTOR   000010R    WAIT     000526R
WASADR   000054R    WBUF   = ****** G   .      = 000722R

         000722
```

ERRORS DETECTED: 0


```
*XLPAB,XLPAB.PRT_DCXCOM.P11,XLPAB.P11
RUN-TIME:  2  3  0 SECONDS
CORE USED:  4K
```

```
    1
  213
                .REM_
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DXCRA-B-D
PRODUCT NAME:	CRA-DEC/X11 CR11 MODULE
DATE:	15 JUN 1973
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR(S):	S. MALLICK

1. ABSTRACT

CRA IS AN IOMOD THAT EXERCISES THE CR-11 CARD READER. IT READS A PRE-PUNCHED ALPHANUMERIC DECK FORMING A CHECKSUM FOR EACH CARD READ. THE CALCULATED CHECKSUM IS COMPARED AGAINST A KNOWN CKSUM AND ANY ERRORS REPORTED ON THE TTY. THE MODULE TESTS BOTH THE DIRECT AND ENCODED DATA.

2. REQUIREMENTS

HARDWARE: ONE CR11 CARD READER WITH CONTROLLER
ONE PRE-PUNCHED ALPHANUMERIC DECK
STORAGE: CRA REQUIRES 229 WORDS OF STORAGE

3. PASS DEFINITION

ONE PASS OF THE CRA MODULE CONSISTS OF READING CARDS UNTIL THE INPUT HOPPER IS EMPTY WHICH RESULTS IN READING 80N WORDS WHERE N=NO. OF CARDS.

4. EXECUTION TIME

ONE PASS OF CRA RUNNING ALONE ON A PDP11/05 PROCESSOR TAKES APPROXIMATELY --- MINUTES (80 CARD DECK)

5. CONFIGUTATION REQUIREMENTS

DEFAULT PARAMETERS:

DEVADR: 177160, VECTOR:230, BR:6, DEVCNT: 1

REQUIRED PARAMETERS:

NONE

6. DEVICE/OPTION SET-UP

A. POWER UP THE READER
B. LOAD THE ALPHA DECK
C. DEPRESS RESET

7. MODULE OPERATION

TEST SEQUENCE:

A. SET UP VECTORS AND INITIALIZE MODULE VAVIABLES
B. READ A CARD - ENABLE INTERRUPT
C. INTERRUPT SERVICE:
    1.)COUNT COLUMN
    2.)FORM CHECKSUMS (DIRECT AND ENCODED)
    3.)IF 80 COLUMNS READ: CHECK DATA - REPORT ERRORS
D. IF HOPPER NOT EMPTY REPEAT B-C
E. AT HOPPER EMPTY (OFF-LINE) REPORT END OF PASS AND RESTART AT A.

OTHER ERROR CONDITIONS TESTED FOR AND REPORTED:

A. COLUMN COUNT
B. COLUMN DONE RESET BY READING DATA

IF OFF-LINE CONDITION CAN NOT BE CORRECTED MODULE WILL LOOP CONTINUOUSLY - NO END PASS PRINTOUT.

8. OPERATION OPTIONS

NONE

9. NON-STANDARD PRINTOUTS

NONE; ALL PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN THE DEC-X11 DOCUMENT.


```
338                                     .LIST SEQ,BIN
339                                     ;CR11 DEC/X11 EXERCISER
340  000000'                                    IOMOD   <XCRAB >,177160,230,6
(2)  000000'                                    MODULE  140000,XCRAB ,177160,230,6,,
(2)                                             ,TITLE  XCRAB DEC/X11 SYSTEM EXERCISER MODULE
(2)                                             .LIST   BIN
(2)                                     ;*****************************************************************
(2)  000000'                            BEGIN:
(2)  000000' 041530  040522  020102     MODNAM: .ASCII  /XCRAB /                ;MODULE NAME.
(2)  000006' 177160                     ADDR:   177160+0                        ;1ST DEVICE ADDR.
(2)  000010' 000230                     VECTOR: 230+0                           ;1ST DEVICE VECTOR.
(2)  000012'    300                     BR1:    .BYTE   PRTY6+0                 ;1ST BR LEVEL.
(2)  000013'    000                     BR2:    .BYTE   PRTY+0                  ;2ND BR LEVEL.
(2)  000014' 000001                     DVID1:  +1                              ;DEVICE INDICATOR 1.
(2)  000016' 000000                     SR1:    OPEN                            ;SWITCH REGISTER 1
(2)                                     ;*****************************************************************
(2)  000020' 140000                     STAT:   140000                          ;STATUS WORD.
(2)  000022' 000162'                    INIT:   START                           ;MODULE START ADDR.
(2)  000024' 000162'                    SPOINT: MODSP                           ;MODULE STACK POINTER.
(2)  000026' 000000                     PASCNT: 0                               ;PASS COUNTER.
(2)  000030' 000000                     ERRCNT: 0                               ;ERROR COUNTER.
(2)  000032' 000000                     SVR0:   OPEN                            ;LOC TO SAVE R0.
(2)  000034' 000000                     SVR1:   OPEN                            ;LOC TO SAVE R1.
(2)  000036' 000000                     SVR2:   OPEN                            ;LOC TO SAVE R2.
(2)  000040' 000000                     SVR3:   OPEN                            ;LOC TO SAVE R3.
(2)  000042' 000000                     SVR4:   OPEN                            ;LOC TO SAVE P4.
(2)  000044' 000000                     SVR5:   OPEN                            ;LOC TO SAVE R5.
(2)  000046' 000000                     SVR6:   OPEN                            ;LOC TO SAVE R6.
(2)  000050' 000000                     CSRA:   OPEN                            ;ADDR OF CURRENT CSR.
(2)  000052'                            SBADR:                                  ;ADDR OF GOOD DATA, OR
(2)  000052' 000000                     ACSR:   OPEN                            ;CONTENTS OF CSR.
(2)  000054'                            WASADR:                                 ;ADDR OF BAD DATA, OR
(2)  000054' 000000                     ASTAT:  OPEN                            ;STATUS REG CONTENTS.
(2)  000056' 000000                     ASB:    OPEN                            ;EXPECTED DATA.
(2)  000060' 000000                     AWAS:   OPEN                            ;ACTUAL DATA.
(2)                                             .REPT   SPSIZ                   ;MODULE STACK STARTS HERE.
(2)                                             .NLIST
(2)                                             .WORD   0
(3)                                             .LIST
(2)                                             .ENDR
(2)  000162'                            MODSP:
(2)                                     ;*****************************************************************
341                                     ;THIS MODULE TESTS THE CR-11 CARD READER
342                                     ;
343                                     ;
344                                     ;
345                                     ;
346                                     ;
347                                     ;
348                                     ;
349                                     ;
350                                     ;
351                                     ;
```

```
                                        ;INITIALIZATION FOR CARD READER CR11
                                        ;VECTOR ADDRESS 230
000162' 016705  177620          START:  MOV     ADDR,R5                 ;GET DEVICE ADDRESS
000166' 010567  177656                  MOV     R5,CSRA                 ;R5 IS CR11 OR CM11 STATUS REGISTER
000172' 016700  177612                  MOV     VECTOR,R0               ;LOAD DEVICE VECTOR
000176' 012720  000252'                 MOV     #CRCM11,(0)+            ;SET VECTOR TO SERVICE ROUTINE
000202' 016720  177604                  MOV     BR1,(0)+                ;SET PRIORITY
000206' 005067  000476                  CLR     CRCK1                   ;CLEAR CHECKSUM-1
000212' 005067  000474                  CLR     CRCK2                   ;CLEAR CHECKSUM-2
000216' 005067  000472                  CLR     CRCLCT                  ;CLEAR READER COLUMN COUNT
000222'                         RDYWT:
(1) 000222' 104407  000000'             BREAK,,BEGIN            ;TEMPORARY RETURN TO MONITOR,
000226' 032715  000400                  BIT     #400,(R5)               ;TEST CR11 READY BIT
000232' 001404                          BEQ     1$                      ;NO GO TRY AGAIN
000234' 005301                          DEC     R1
000236' 001371                          BNE     RDYWT
(1) 000240' 104403  000000'             END,,BEGIN              ;
000244' 012715  000101          1$:     MOV     #101,(R5)               ;START READ, ENABLE INTERRUPT
000250' 104400                          EXIT.                   ;RETURN TO MONITOR,


                                        ;INTERRUPT SERVICE ROUTINE FOR CARD READER
000252' 010546                  CRCM11: MOV     R5,-(R6)                ;SAVE R5 HERE USED AS CONTROL REG
000254' 016705  177564                  MOV     SVR5,R5                 ;MOVE SAVED R5 INTO R5
000260' 105715                          TSTB    (R5)                    ;CHECK COLUMN READY FOR READING
000262' 100026                          BPL     CRCHK1                  ;BRANCH IF ALREADY READ
000264' 005267  000424                  INC     CRCLCT                  ;COUNT NUMBER OF COLUMNS
000270' 066567  000002  000412          ADD     2(R5),CRCK1             ;ADD DATA TO CHECKSUM 1
000276' 066567  000004  000406          ADD     4(R5),CRCK2             ;ADD ENCODED DATA TO CHECKSUM 2
000304' 105715                          TSTB    (R5)                    ;CHECK COLUMN DONE
000306' 100012                          BPL     NXCOL                   ;BRANCH IF DONE
000310' 012605                          MOV     (R6)+,R5                ;RESTORE R5
                                        ;*****************************************************************
(1) 000312' 000004  000320' 000000'     PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
(1)                                     ;*****************************************************************
000320' 004767  000066          1$:     JSR     R7,ERSUB                ;READING DATA DIDN'T CLEAR COLUMN DONE
(1) 000324' 104404  000000'             ERROR,,BEGIN                    ;COLUMN DONE
(1)                                     ;*****************************************************************
000330' 000167  000256                  JMP     CRCHK4                  ;START NEW CARD
000334' 012605                  NXCOL:  MOV     (R6)+,R5                ;RESTORE R5
000336' 000002                          RTI                             ;RETURN
000340' 012605                  CRCHK1: MOV     (R6)+,R5                ;RESTORE R5
(1) 000342' 000004  000350' 000000'     PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
(1)                                     ;*****************************************************************
000350' 032715  040400          1$:     BIT     #40400,(R5)             ;CHECK FOR CARD DONE
000354' 001527                          BEQ     CRCHK2                  ;BRANCH IF CARD NOT DONE
000356' 022767  000120  000330          CMP     #80.,CRCLCT             ;CHECK FOR EIGHTY COLUMNS
000364' 001422                          BEQ     CR1                     ;BRANCH IF OK
000366' 022767  000050  000320          CMP     #40.,CRCLCT             ;40 COLUMNS
000374' 001413                          BEQ     CR0                     ;BRANCH TO 40 COLUMN CHECK
```



```
000376' 004767  000010                  JSR     R7,ERSUB
(1) 000402' 104404  000000'             ERROR,,BEGIN                    ;COLUMN COUNT
                                        ;*****************************************************************
                                                                        ;TOTAL NUMBER OF COLUMNS READ NOT 80 OR 40
000406' 000167  000200                  JMP     CRCHK4                  ;MOST PROBABLY NO CARDS IN READER
000412' 010567  177432          ERSUB:  MOV     R5,CSRA                 ;START NEW CARD
000416' 011567  177430                  MOV     (R5),ACSR               ;MOVE STATUS REG, ADDRESS
000422' 000207                          RTS     R7                      ;MOVE STATUS REG, CONTECTS

000424' 012715  000101          CR0:    MOV     #101,   (R5)
000430' 104400                          EXIT.                           ;RESTART READER FOR SECOND CARD
                                                                        ;RETURN TO MONITOR,

000432' 026767  000246  000250  CR1:    CMP     CRSUM1,CRCK1            ;CHECK FOR CORRECT ALPHA CARD IMAGE SUM
000440' 001420                          BEQ     CRCONT                  ;BRANCH IF CORRECT
000442' 012767  000704' 177402          MOV     #CRSUM1,$BADR           ;GOOD CHECKSUM ADDRESS
000450' 016767  000230  177400          MOV     CRSUM1, ASB             ;GOOD CHECKSUM
000456' 012767  000710' 177370          MOV     #CRCK1, WASADR          ;ADDRESS OF BAD CHECKSUM
000464' 016767  000220  177366          MOV     CRCK1,  AWAS            ;CHECKSUM AS WAS
(1) 000472' 104405  000000'             ;*****************************************************************
(1)                                     DATER,,BEGIN            ;DATA ERROR!!!
                                        ;*****************************************************************

000476' 000167  000110                  JMP     CRCHK4                  ;CARD IMAGE CHECKSUM INCORRECT
000502' 026767  000200  000202  CRCONT: CMP     CRSUM2,CRCK2            ;START NEW CARD
000510' 001420                          BEQ     RIG1                    ;CHECK FOR CORRECT ALPHA ENCODED SUM
000512' 012767  000706' 177332          MOV     #CRSUM2,$BADR           ;BRANCH IF CORRECT
000520' 016767  000162  177330          MOV     CRSUM2, ASB             ;GOOD CHECKSUM ADDRESS
000526' 012767  000712' 177320          MOV     #CRCK2, WASADR          ;GOOD CHECKSUM
000534' 016767  000152  177316          MOV     CRCK2,  AWAS            ;BAD CHECKSUM ADDRESS
                                        ;BAD CHECKSUM
(1) 000542' 104405  000000'             ;*****************************************************************
(1)                                     DATER,,BEGIN            ;DATA ERROR!!!
                                        ;*****************************************************************

000546' 000167  000040                  JMP     CRCHK4                  ;ENCODED CHECKSUM INCORRECT
000552' 032715  000400          RIG1:   BIT     #400,(R5)               ;GET NEW CARD
000556' 001415                          BEQ     CRCHK4                  ;CHECK OFF-LINE (BIT 8)
000560' 042715  000100                  BIC     #100,   (R5)            ;BRANCH IF NOT SET
000564' 104402  000162' 000000'         ENDPS,,START,BEGIN      ;SIGNAL ;CLEAR ENABLE INTERRUPT
                                                                        ;END OF PASS, RESUME AT START
                                                                        ;IF READER OFF-LINE, DON'T START UP AGAIN
000572' 032715  000400          RIG2:   BIT     #400,(R5)               ;END OF DECK
000576' 001002                          BNE     1$                      ;TEST FOR ON-LINE
000600' 000167  177356                  JMP     START                   ;REPORT ERROR
000604'                                                                 ;RESTART IF ON LINE
(1)                             1$:
(1) 000604' 104404  000000'             ;*****************************************************************
(1)                                     ERROR,,BEGIN                    ;RELOAD DECK - DEPRESS RESET
000610' 000770                          ;*****************************************************************
                                        RR      RIG2                    ;LOOP TIL ON LINE

000612' 005067  000076          CRCHK4: CLR     CRCLCT                  ;CLEAR COLUMN COUNT
000616' 005067  000066                  CLR     CRCK1                   ;CLEAR CHECKSUMS
```

```
   444  000622' 005067  000064                CLR     CRCK2
   445  000626' 012715  000101                MOV     #101,(R5)               ;CLEAR INTERRUPTING CONDITION, RESTART READER
   446  000632' 104400                        EXIT.                   ;RETURN TO MONITOR.
   447  000634' 005715                CRCHK2: TST     (R5)                    ;CHECK BIT 15
   448  000636' 100011                        BPL     CRCHK3                  ;BRANCH IF NOT SET
   449  000640' 032715  000400                BIT     #400,(R5)               ;CHECK OFF-LINE (BIT 8)
   450  000644' 001362                        BNE     CRCHK4                  ;BR IF SET
   451  000646' 004767  177540                JSR     R7,ERSUB
   452                                        ;*****************************************************************
(1) 000652' 104404  000000'                ERROR.,BEGIN            ;FALSE ERROR
(1)                                         ;*****************************************************************
   453                                                                ;ERROR BIT WAS SET, OTHERS WEREN'T
   454  000656' 000167  177730                JMP     CRCHK4                  ;GET NEW CARD
   455  000662' 032715  002000        CRCHK3: BIT     #2000,(R5)              ;CHECK BIT 10
   456  000666' 001351                        BNE     CRCHK4                  ;BRANCH IF SET
   457  000670' 004767  177516                JSR     R7,ERSUB
   458                                        ;*****************************************************************
(1) 000674' 104404  000000'                ERROP.,BEGIN            ;FALSE INTERRUPT
(1)                                         ;*****************************************************************
   459                                                                ;NO INTERRUPTING BITS WERE SET
   460  000700' 000167  177706                JMP     CRCHK4                  ;GET NEW CARD
   461  000704' 067443                CRSUM1: 67443                   ;DESIRED TOTAL FOR ALPHANUMERIC CARD-IMAGE DATA
   462  000706' 014173                CRSUM2: 14173                   ;DESIRED TOTAL FOR ALPHANUMERIC ENCODED DATA
   463  000710' 000000                CRCK1:  0                       ;RUNNING CHECKSUM FOR CARD IMAGE
   464  000712' 000000                CRCK2:  0                       ;RUNNING CHECKSUM FOR ENCODED DATA
   465  000714' 000000                CRCLCT: 0                       ;CARD READER COLUMN COUNT
   466          000001                .END
```



```
ACSR     000052R        ADDR     000006R        ASB      000056R        ASTAT    000054R
AWAS     000060R        BDCNV  = ****** G       BEGIN    000000R        BIT0   = 000001
BIT1   = 000002         BIT10  = 002000         BIT11  = 004000         BIT12  = 010000
BIT13  = 020000         BIT14  = 040000         BIT15  = 100000         BIT2   = 000004
BIT3   = 000010         BIT4   = 000020         BIT5   = 000040         BIT6   = 000100
BIT7   = 000200         BIT8   = 000400         BIT9   = 001000         BREAK, = 104407
BR1      000012R        BR2      000013R        CRCHK1   000340R        CRCHK2   000634R
CRCHK3   000662R        CRCHK4   000612R        CRCK1    000710R        CRCK2    000712R
CRCLCT   000714R        CRCM11   000252R        CRCONT   000502R        CRSUM1   000704R
CRSUM2   000706R        CR0      000424R        CR1      000432R        CSRA     000050R
DATEP, = 104405         DVID1    000014R        EABITS = ****** G       ENDPS, = 104402
END,   = 104403         ERRCNT   000030R        ERRN,  = 104410         ERROR, = 104404
ERSUB    000412R        EXIT,  = 104400         HICORE = ****** G       INIT     000022R
LOCORE = ****** G       MODNAM   000000R        MODSP    000162R        MSGN,  = 104411
MSG,   = 104406         NXCOL    000334R        OACNV  = ****** G       OPEN   = 000000
PASCNT   000026R        PC     =%000007         PIRQ,  = 000004         POPSP  = 005726
POPSP2 = 022626         PRTY   = 000000         PRTY0  = 000000         PRTY1  = 000040
PRTY2  = 000100         PRTY3  = 000140         PRTY4  = 000200         PRTY5  = 000240
PRTY6  = 000300         PRTY7  = 000340         PS     = 177776         PSW    = 177776
PUSH   = 005746         PUSH2  = 024646         QUE,   = 104401         RDYWT    000222R
RIG1     000552R        RIG2     000572R        R0     =%000000         R1     =%000001
R2     =%000002         R3     =%000003         R4     =%000004         R5     =%000005
R6     =%000006         R7     =%000007         SBADR    000052R        SP     =%000006
SPOINT   000024R        SPSIZ  = 000040         SR1      000016R        START    000162R
STAT     000020R        SVR0     000032R        SVR1     000034R        SVR2     000036R
SVR3     000040R        SVR4     000042R        SVR5     000044R        SVR6     000046R
TPX    = 000000         TRAPX  = 000012         VECTOR   000010R        WASADR   000054R
WBUF   = ****** G       .      = 000716R

         000716
```

ERRORS DETECTED: 0

```
*XCRAB,XCRAB,PRT_DCXCOM,P11,XCRAB,P11
RUN-TIME:  2  3  0 SECONDS
CORE USED:  4K
```



```
    1
  213
                .PEM_
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DXDPA-B-D
PRODUCT NAME:	DPA-DEC/X11 DP11
DATE:	JUN. 15, 1973
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR(S):	AL COSSETTE

1. ABSTRACT

DPA IS AN IOMOD THAT EXERCISES UP TO EIGHT 8-BIT SYNCHRONOUS LINE INTERFACES (DP11) BY TRANSMITTING A STANDARD BINARY COUNT PATTERN USING THE MAINTENANCE MODE FEATURE. THE RECEIVED DATA IS COMPARED WITH THE TRANSMITTED DATA AND ANY ERRORS ARE REPORTED VIA THE CONSOLE TTY. ALL AVAILABLE INTERFACES (UP TO 8) ARE ACTIVATED AND RUNNING SIMULTANEOUSLY.

2. REQUIREMENTS

HARDWARE: DP11 ASYNCHRONOUS INTERFACE
STORAGE: DPA REQUIRES 524 WORDS OF STORAGE

3. PASS DEFINITION

ONE PASS OF THE DPA MODULE CONSISTS OF TRANSMITTING AND RECEIVING 12800 8-BIT CHARACTERS (TOTAL)

4. EXECUTION TIME

DPA RUNNING ALONE ON A PDP11/05 PROCESSOR TAKES APPROXIMATELY --- MINUTES TO COMPLETE ONE PASS.

5. CONFIGURATION PARAMETERS

DEFAULT PARAMETERS:

DEVADR: 174770, VECTOR:440, BR1:5, BR2:5, DEVCNT:1

REQUIRED PARAMETERS: NONE

6. DEVICE/OPTION SETUP

NONE: NO DEVICE IS REQUIRED IN MAINTENANCE MODE

7. MODULE OPERATION

TEST SEQUENCE:

A. TEST UP TO 8 POSSIBLE DEVICES FOR SELECTION
B. STORE THE DEVICE NO. OF DEVICES TO BE TESTED AND SET UP THE VECTORS FOR THESE DEVICES
C. TURN ON RECEIVER INTERRUPT ENABLE, TRANSMITTER INTERRUPT ENABLE, AND MAINTENANCE MODE FOR ALL ACTIVE DEVICES.
D. INITIAL TRANSMITTER INTERRUPT SERVICE:

    1.) TEST FOR FALSE INTERRUPT (READY (0)); REPORT ERRORS
    2.) OUTPUT NEXT CHARACTER TO EACH ACTIVE DEVICE
    3.) RETURN TO MONITOR TO WAIT FOR RECEIVER INTERRUPT.

E. RECEIVER INTERRUPT SERVICE:

    1.) TEST FOR FALSE INTERRUPT (DONE (0)); REPORT ERRORS
    2.) COMPARE INPUT/OUTPUT DATA; REPORT ERRORS
    3.) RETURN TO MONITOR TO WAIT FOR TRANSMITTER INTERRUPT

F. REPEAT D AND E UNTIL 12800.(TOTAL) CHARACTERS HAVE BEEN PROCESSED
G. AT END OF PASS TURN OFF ALL ACTIVE DEVICES AND RESTART AT B

B. OPERATION OPTIONS

A. LOCATION DVID1 (DPA 14) MAY BE CHANGED TO SELECT ANY COM-
    BINATION OF DEVICES BIT0=DEV0, BIT1=DEV1 ....BIT7=DEV7
    IF DVID1 IS INITIALLY 0 DPA WILL BE DROPPED FROM TEST.

B. LOCATION STRT1+2 (DPA 1462) MAY BE MODIFIED TO INCREASE
    OR DECREASE THE TOTAL NUMBER OF CHARACTERS PROCESSED PER PASS

C. NON STANDARD PRINTOUTS

NONE; ALL PRINTOUTS HAVE STANDARD FORMATS AS DESCRIBED IN THE
      DEC/X11 DOCUMENT.


```
                                        .LIST SEQ,BIN
                                        ; SET UP VECTOR (RETURN ADDRESS(PC)) PC = INTR SERV, AREA.
                                                IOMOD   <XDPAB >,174770,440,5,5
                                                MODULE  140000,XDPAB ,174770,440,5,5,
                                                .TITLE  XDPAB DEC/X11 SYSTEM EXERCISER MODULE
                                                .LIST   BIN
                                        ;*****************************************************************************
                                        BEGIN:
000000' 042130  040520  020102          MODNAM: .ASCII  /XDPAB /                ;MODULE NAME.
000006' 174770                          ADDR:   174770+0                        ;1ST DEVICE ADDR.
000010' 000440                          VECTOR: 440+0                           ;1ST DEVICE VECTOR.
000012'    240                          BR1:    .BYTE   PRTY5+0                 ;1ST BR LEVEL.
000013'    240                          BR2:    .BYTE   PRTY5+0                 ;2ND BR LEVEL.
000014' 000001                          DVID1:  +1                              ;DEVICE INDICATOR 1.
000016' 000000                          SR1:    OPEN                            ;SWITCH REGISTER 1
                                        ;*****************************************************************************
000020' 140000                          STAT:   140000                          ;STATUS WORD.
000022' 000162'                         INIT:   START                           ;MODULE START ADDR.
000024' 000162'                         SPOINT: MODSP                           ;MODULE STACK POINTER.
000026' 000000                          PASCNT: 0                               ;PASS COUNTER.
000030' 000000                          ERRCNT: 0                               ;ERROR COUNTER.
000032' 000000                          SVR0:   OPEN                            ;LOC TO SAVE R0.
000034' 000000                          SVR1:   OPEN                            ;LOC TO SAVE R1.
000036' 000000                          SVR2:   OPEN                            ;LOC TO SAVE R2.
000040' 000000                          SVR3:   OPEN                            ;LOC TO SAVE R3.
000042' 000000                          SVR4:   OPEN                            ;LOC TO SAVE R4.
000044' 000000                          SVR5:   OPEN                            ;LOC TO SAVE R5.
000046' 000000                          SVR6:   OPEN                            ;LOC TO SAVE R6.
000050' 000000                          CSRA:   OPEN                            ;ADDR OF CURRENT CSR.
000052'                                 SBADR:                                  ;ADDR OF GOOD DATA, OR
000052' 000000                          ACSR:   OPEN                            ;CONTENTS OF CSR.
000054'                                 WASADR:                                 ;ADDR OF BAD DATA, OR
000054' 000000                          ASTAT:  OPEN                            ;STATUS REG CONTENTS.
000056' 000000                          ASB:    OPEN                            ;EXPECTED DATA.
000060' 000000                          AWAS:   OPEN                            ;ACTUAL DATA.
                                                .REPT   SPSIZ                   ;MODULE STACK STARTS HERE.
                                                .NLIST
                                                .WORD   0
                                                .LIST
                                                .ENDR
000162'                                 MODSP:
                                        ;*****************************************************************************
000162' 005767  177626                  START:  TST     DVID1                   ;CHECK ANY DPN'S ON LINE
000166' 001002                                  BNE     1$                      ;YES
000170' 104403  000000'                         END,,BEGIN                      ;
000174' 016767  177614  001764          1$:     MOV     DVID1,DVIDA
000202' 016701  177602                          MOV     VECTOR,R1               ;R1 = VECTOR ADDRESS
000206' 012702  002020'                         MOV     #LINKER,R2              ;R2 = LINK; JSR TABLE WITH OFFSET
000212' 012767  000001  001740                  MOV     #1,PNTR                 ;SET UP PNTR TO TEST DEVICE ON LINE
000220' 036767  001734  177566          2$:     BIT     PNTR,DVID1              ;TEST IS THIS DP11 ON LINE
000226' 001420                                  BEQ     3$                      ;NO GO CHANGE DP ADDR & TRY AGAIN
000230' 010221                                  MOV     R2,(R1)+                ;SET UP VECTOR RETURN ADDRESS(RCV)
```

```
                                        MOVB    BR1,(R1)+       ;SET UP VECTOR PRIORITY (RCV)
                                        TSTB    (R1)+           ;INCR. POINTER
                                        ADD     #6,R2           ;UPDATE NEW LINK ADDRESS
                                        MOV     R2,(R1)+        ;SET UP VECTOR RETURN ADDRESS (XMT)
                                        MOVB    BR1,(R1)+       ;SET UP VECTOR PRIORITY )XMT)
                                        TSTB    (R1)+           ;INCR, POINTER
                                        ADD     #6,R2           ;UPDATE NEW LINK ADDRESS
                                4$:     ASLB    PNTR            ;SET UP FOR NEW DEVICE COMPARE
                                        BCS     START1          ;HAVE WE TESTED FOR ALL ON LINE
                                                                ; DEVICES
                                        BR      2$              ;NOT DONE GO DO MORE
                                3$:     ADD     #10,R1          ;UPDATE TO NEW VECTOR ADDRESS
                                        ADD     #14,R2          ;UPDATE TO NEW LINK ADDRESS
                                        BR      4$              ;GO TEST FOR NEXT DEVICE ON LINE


                                ; THIS CODE WILL CLEAR ALL OF THE WRITE BUFFER AREA

                        001654  START1: MOV     #103.,,CNT80    ;COUNT REQUIRED TO GO THRU
                                                                ;ALL 88 DATA STORAGE BUFFER
                                        MOV     #DPLIN,R3       ;STARTING ADDRESS OF
                                                                ; DATA BUFFER LOCATIONS,
                                1$:     CLR     (R3)+           ;CLEAR DATA BUFF REG
                                        DEC     CNT80           ;ARE THERE MORE TO CLEAR?
                                        BNE     1$              ;NO GO BACK & DO THE REST


                                ; THIS CODE WILL SELECT WHICH LINES (<1:8>) HAVE
                                ; BEEN SELECTED FOR TEST & TRANSMIT SYNC TO START
                                ; TESTING ALL LINES,

                        001640  INT:    MOV     #10,COUNT       ;SET COUNT VALUE
                                        MOV     ADDR,R1         ;R1=174770
                                        MOV     #DVAD1,R2       ;R2=DVAD1 ADDR,
                                        MOV     #DPLIN,R3       ;LINE BUFFER POINTER
                                        MOV     #DPLIN1,R4      ;START OF LINE BUFF
                                1$:     MOV     R1,(R2)+        ;DVADX=DEVICE ADDR, CODE
                                        MOV     R4,(R3)+        ;BUFF POINTER HAS START OF LINE
                                                                ;BUFF STORAGE
                                        ADD     #-10,R1         ;UPDATE
                                        ADD     #20,R4          ;UPDATE
                                        DEC     COUNT           ;CNT DOWN
                                        BNE     1$              ;NOT DONE GO BACK FOR MORE


                                START2: CLR     NODVTS          ;CLEAR NO, OF LINE TESTED
                                                                ;REG,
                                        MOV     ADDR,R1         ;GET DEVICE ADDRESS
                                        MOV     ADDR,R2         ;XMT CSR ADDRESS IN R2
                                        ADD     #4,R2           ;R2=XMT CSR REGISTER 174XX4
                                        MOV     #LNSYN1+1,R0    ;SET UP R0 TO POINT TO LNSYN LOC,
                                        MOV     #LNCNT1,R3
                                        MOV     #1,PNTR         ;SET PNTR REG POINTER TO
                                                                ; 1ST DEVICE ON LINE
                                D8:     BIT     PNTR,DVID1      ;TEST IS THIS DEVICE ON LINE
```



```
                                        BEQ     B$
                                        MOVB    #4,(R0)         ;NO GO UPDATE ADDRESS
                                                                ;PLACE SYNC COUNT INTO HIGH
                                        MOV     #10020,(R3)     ;BYTE LNSYN X
                                                                ;COUNT =16 HIGH FOR XMT
                                                                ;LOW FOR RCV
                                                                ; BIT 3 = TRANSMIT SYNC ON INTR,
                                C$:     ADD     #2,R0           ;UPDATE LNSYN X POINTER
                                        TST     (R3)+           ;UPDATE LNCNT X POINTER
                                        CCC                     ;CLEAR CARRY BIT (CLR FOR TEST)
                                        ASLB    PNTR            ;HAS 8 DP11'S BEEN INITIALIZED
                                        BCC     D$              ;NO GO BACK SET UP NEXT DP
                                        MOV     #1,PNTR ;NEW POINTER
                                KCKOFF: BIT     PNTR,DVID1      ;IS THIS LINE ON
                                        BEQ     UPDAT           ;GO UPDATE ADDRESS
                                        BIS     #105,(R1)       ;INTR ENABLE,MAINT, MODE,
                                                                ;& STRIP SYNC
                                        MOVB    TSYNC,3(R1)     ;LOAD SYNC BUFFER
                                        BIS     #312,(R2)       ;INITIALIZE XMT STATUS
                                                                ;7=DONE
                                                                ;6=INTR ENABLE
                                                                ;3=XMT SYNC ON INTR
                                                                ;1=IDLE SYNC
                                        CLRB    3(R2)           ;CLEAR SYNC EXT
                                UPDAT:  ADD     #-10,R1         ;INDEX RCV CSR
                                        ADD     #-10,R2         ;INDEX XMT CSR
                                        CCC                     ;CLEAR CONDITION CODES
                                        ASLB    PNTR            ;MOVE POINTER FOR NEXT TEST
                                        BCC     KCKOFF          ;GO ENABLE NEXT LINE
                                        EXIT.                   ;RETURN TO MONITOR,

                                B$:     COM     (R3)            ;SET LNCNT X FLAG (THIS DP NOT SELECTED)
                                        INC     NODVTS          ;SET UP DEVICE COUNT
                                        BR      C$              ;GO BACK & UPDATE REG,


                                ;THIS CODE WILL ANSWER THE XMT INTERRUPT REQUEST
                                ; FOR SERVICE

                                DPXMT:  MOV     R0,-(SP)        ;SAVE REG, 0 ON STACK
                                        MOV     R1,-(SP)        ;SAVE REG, 1 ON STACK
                                        MOV     (R5),R0         ;R0 HAS LINE NO, OFFSET
                                        MOV     DVAD1(R0),R1    ;R1 = R0 WITH OFFSET VALUE
                                        TSTB    4(R1)           ;TEST IF DONE BIT SET
                                        BMI     DPXMT1          ;DONE IS SET CONT, PROGRAM
                                        MOV     R0,CSRA         ;SAVE CSRA ADDR,
                                        MOV     (R0),ACSR       ;SAVE CONTENTS OF CSR
                                        MOV     (SP)+,R1        ;RESTORE STACK
                                        MOV     (SP)+,R0        ;RESTORE STACK
                                        MOV     (SP)+,R5        ;RESTORE STACK
(1)  000620' 000004  000626' 000000'    ;-----------------------------------------------------------------
(1)                                     PIRQ,,FITER,BEGIN       ;QUEUE REQUEST TO CONTINUE AT FITER
                                        ;-----------------------------------------------------------------
```

```
                                FIRER:
(1)                                     ;*****************************************************************
(1)  000626' 104404  000000'            ERROR,,BEGIN            ;FALSE INTERRUPT
(1)                                     ;*****************************************************************
463  000632' 104400                     EXIT.                   ;RETURN TO MONITOR.
464
465
466  000634' 032761  000010  000004  DPXMT1: BIT     #10,4(R1)       ;TEST FOR RESYNC
467  000642' 001021                          BNE     DPXMT2          ;BRANCH IF IN SYNC
468  000644' 116061  001540' 000006          MOVB    LNSYN1(R0),6(R1)    ;SEND DATA TO ACTIVE DP LINE NO BUFF
469  000652' 105260  001540'                 INCB    LNSYN1(R0)              ;INCRAMENT NEXT DATA WORD
470  000656' 122760  000026  001540'         CMPB    #26,LNSYN1(R0)  ;CHCK IS THIS = TO SYNC CHAR.
471  000664' 001002                          BNE     DPXMT3          ;OK CONT.
472  000666' 105260  001540'                 INCB    LNSYN1(R0)              ;INC SYNC CHAR.(THIS IS DONE
473                                                                  ;SO THAT STRIP SYNC CHAR. WILL
474                                                                  ;NOT MAKE AN ERROR)
475  000672' 105360  001561'         DPXMT3: DECB    LNCNT1+1(R0)    ;CHECK HAVE WE XMTED
476                                                                  ;ALL 16 CHAR,, THIS LINE
477  000676' 001017                          BNE     XMTRTN          ;NO RETURN TO MONITOR
478  000700' 052711  000010                  BIS     #10,(R1)        ;SET RE-SYNC BIT
479  000704' 000414                          BR      XMTRTN          ;RETURN TO MONITOR.
480  000706' 116161  000003  000006  DPXMT2: MOVB    3(R1),6(R1)     ;XMT SYNC CHAR. (TSYNC)
481  000714' 105360  001541'                 DECB    LNSYN1+1(R0)    ;DEC SYNC COUNTER
482  000720' 001006                          BNE     XMTRTN          ;EXIT IF SYNC COUNT NOT ZERO
483  000722' 112760  000004  001541'         MOVB    #4,LNSYN1+1(R0) ;RE-INITIALIZE SYNC COUNTER
484  000730' 042761  000010  000004          BIC     #10,4(R1)       ;CLEAR SYNC FLAG
485  000736' 012601                  XMTRTN: MOV     (SP)+,R1        ;RESTORE STACK
486  000740' 012600                          MOV     (SP)+,R0        ;
487  000742' 012605                          MOV     (SP)+,R5        ;
488  000744' 000002                          RTI                     ;RETURN TO INTERRUPTER CODE
489
490                                  ; THIS CODE WILL ANSWER THE RCV INTERRUPT
491                                  ; REQUEST FOR SERVICE
492
493  000746' 010246                  DPRCV1: MOV     R2,-(SP)        ;SAVE REG, 2 ON STACK
494  000750' 010346                          MOV     R3,-(SP)        ;SAVE REG, 3 ON STACK
495  000752' 010446                          MOV     R4,-(SP)        ;SAVE REG, 4 ON STACK
496  000754' 011503                          MOV     (R5),R3         ;GET OFFSET
497  000756' 016304  001520'                 MOV     DVAD1(R3),R4    ;R3 = R4 DEVICE CODE OFFSET VALUE
498  000762' 105714                          TSTB    (R4)            ;IS DONE SET
499  000764' 100416                          BMI     DPRCV1          ;DONE SET ;SERV DONE REQUEST
500  000766' 010367  177056                  MOV     R3,CSRA         ;SHOW CSR ADDR.
501  000772' 011367  177054                  MOV     (R3),ACSR       ;CONTENTS OF CSR
502  000776' 012604                          MOV     (SP)+,R4        ;RESTORE STACK
503  001000' 012603                          MOV     (SP)+,R3        ;
504  001002' 012602                          MOV     (SP)+,R2        ;
505  001004' 012605                          MOV     (SP)+,R5        ;
506                                          ;-------------------------------------------------------------
(1)  001006' 000004  001014' 000000'         PIRQ,,FIRER,BEGIN       ;QUEUE REQUEST TO CONTINUE AT FIRER
(1)                                          ;-------------------------------------------------------------
507
508  001014'                         FIRER:
(1)                                     ;*****************************************************************
(1)  001014' 104404  000000'            ERROR,,BEGIN            ;FALSE INTERRUPT
```



```
(1)                                     ;*****************************************************************
509  001020' 104400                     EXIT.                   ;RETURN TO MONITOR.
510
511  001022' 032764  040000  000004  DPRCV1: BIT     #40000,4(R4)    ;IS OVERRUN BIT SET
512  001030' 001432                          BEQ     READ            ;NO OVERRUN GO READ DATA
513  001032' 105363  001540'                 DECB    LNSYN1(R3)      ;UPDATE XMT DATA
514  001036' 105263  001561'                 INCB    LNCNT1+1(R3)    ;  "      "  ACTIVE COUNT
515  001042' 042764  160000  000004          BIC     #160000,4(R4)   ;CLEAR OVERRUN ERROR BITS
516  001050' 052763  000010  000004          BIS     #10,4(R3)       ;SET RESYNC FLAG
517  001056' 042713  004000                  BIC     #4000,(R3)      ;CLEAR RECEIVE ACTIVE
518  001062' 010367  176762                  MOV     R3,CSRA         ;CSR ADDR.
519  001066' 011367  176760                  MOV     (R3),ACSR       ;CONTENTS CSR
520  001072' 012604                          MOV     (SP)+,R4        ;RESTORE STACK
521  001074' 012603                          MOV     (SP)+,R3        ;
522  001076' 012602                          MOV     (SP)+,R2        ;
523  001100' 012605                          MOV     (SP)+,R5        ;
524                                          ;-------------------------------------------------------------
(1)  001102' 000004  001110' 000000'         PIRQ,,OVERR,BEGIN       ;QUEUE REQUEST TO CONTINUE AT OVERR
(1)                                          ;-------------------------------------------------------------
525
526
527  001110'                         OVERR:
(1)                                     ;*****************************************************************
(1)  001110' 104404  000000'            ERROR,,BEGIN            ;OVERRUN ERROR
(1)                                     ;*****************************************************************
528  001114' 104400                     EXIT.                   ;RETURN TO MONITOR.
529
530
531  001116' 032714  004000          READ:   BIT     #4000,(R4)      ;IS DEVICE ACTIVE
532  001122' 001437                          BEQ     RCVRTN          ;GET OUT DEVICE NOT READY
533  001124' 005002                          CLR     R2              ;CLEAR BYTE PNTER
534  001126' 066302  001600'                 ADD     VRFLG1(R3),R2   ;GET BYTE OFFSET
535  001132' 066302  001500'                 ADD     DPLIN(R3),R2    ;ADDR=DATA BUFF ADDR
536  001136' 116412  000002                  MOVB    2(R4),(R2)      ;DATA => DATA BUFF
537  001142' 122712  000026                  CMPB    #26,(R2)        ;SKP IF SYNC BIT
538  001146' 001425                          BEQ     RCVRTN          ;
539  001150' 105263  001600'                 INCB    VRFLG1(R3)      ;
540  001154' 105363  001560'                 DECB    LNCNT1(R3)      ;CHECK HAVE WE TRANSFERRED ALL
541                                                                  ; DATA WORDS,
542  001160' 001020                          BNE     RCVRTN          ;THIS LINE NOT DONE RECEIVING
543                                                                  ;ALL DATA TRANSFERS
544  001162' 005014                          CLR     (R4)            ;CLEAR RCV, CSR REG,
545  001164' 005064  000004                  CLR     4(R4)           ;CLEAR XMT, CSR, REG,
546  001170' 106367  000772          X:      ASLB    DVIDA
547  001174' 103375                          BCC     X
548  001176' 105767  000764                  TSTB    DVIDA
549  001202' 001007                          BNE     RCVRTN
550  001204' 012604                          MOV     (SP)+,R4        ;RESTORE STACK
551  001206' 012603                          MOV     (SP)+,R3        ;
552  001210' 012602                          MOV     (SP)+,R2        ;
553  001212' 012605                          MOV     (SP)+,R5        ;
554                                          ;-------------------------------------------------------------
(1)  001214' 000004  001234' 000000'         PIRQ,,CHCK,BEGIN        ;QUEUE REQUEST TO CONTINUE AT CHCK
(1)                                          ;-------------------------------------------------------------
```

```
                                                                        ;NO HAVE ALL LINES RCV
                                                                        ;SOME DATA WORDS YES WAIT
                                                                        ;FOR COMPLET


001222' 012604                  RCVRTN: MOV     (SP)+,R4        ;RESTORE STACK POINTER
001224' 012603                          MOV     (SP)+,R3        ;
001226' 012602                          MOV     (SP)+,R2        ;
001230' 012605                          MOV     (SP)+,R5        ;
001232' 000002                          RTI                     ;RETURN TO MAINLINE


001234' 005001                  CHCK:   CLR     R1
001236' 005002                          CLR     R2
001240' 005000                          CLR     R0              ;CLEAR R0;R0 WILL BE
                                                                ;USED AS OFFSET
001242' 012767  000020  000722  CHCK1:  MOV     #20,COUNT       ;FOR COUNTING NO OF
                                                                ;CHAR, READ
001250' 005002                          CLR     R2              ;CLR BUFF POINTER
001252' 012701  001560'                 MOV     #LNCNT1,R1
001256' 105711                  1$:     TSTB    (R1)
001260' 001402                          BEQ     CHCK2
001262' 022122                          CMP     (R1)+,(R2)+
001264' 000774                          BR      1$
001266' 010200                  CHCK2:  MOV     R2,R0           ;R0 WILL HOLD LINE NO,/2
001270' 016202  001500'                 MOV     DPLIN(R2),R2    ;R2=START ADDR, THIS LINE BUFF
001274' 111267  000700                  MOVB    (R2),CHECKR     ;GET FIRST CHAR,
001300' 126722  000674          CONTNU: CMPB    CHECKR,(R2)+    ;CHECK DATA & INCR, POINTER
001304' 001410                          BEQ     1$              ;THIS WORD GOOD GO CHECK MORE
001306' 122767  000026  000664          CMPB    #26,CHECKR      ;WAS IT STRIP CHAR,
001314' 001022                          BNE     ERRRT           ;NO GO REPORT ERROR
001316' 005267  000656                  INC     CHECKR          ;YES UPDATE CHECHR
001322' 005302                          DEC     R2              ;UPDATE DPLIN BUFFER POINTER
001324' 000765                          BR      CONTNU          ;GO BACK & CHECK REAL DATA
001326' 005267  000646          1$:     INC     CHECKR          ;SET UP FOR NEXT BYTE TEST
001332' 005367  000634                  DEC     COUNT           ;ONE MORE BYTE HAS BEEN TESTED
001336' 001360                          BNE     CONTNU          ;NOT DONE YET GO CHECK MORE
001340' 005267  000624                  INC     NODVTS          ;THIS LINE DONE ADD 1 TO
                                                                ;NO, OF DEVICES TESTED
001344' 012711  100777                  MOV     #100777,(R1)
001350' 022767  000010  000612          CMP     #10,NODVTS      ;HAVE ALL LINES BEEN TESTED
001356' 001435                          BEQ     PASS            ;GO TO END PASS CODING
001360' 000730                          BR      CHCK1

001362' 016067  001520' 176460  ERRRT:  MOV     DVAD1(R0),CSRA  ;CSRA=LINE ADDR,
001370' 005302                          DEC     R2              ;UPDATE POINTER TO DATA BUFF
001372' 111267  176462                  MOVB    (R2),AWAS       ;BAD DATA BYTE
001376' 005202                          INC     R2              ;UPDATE POINTER TO DATA BUFF
001400' 116767  000574  176450          MOVB    CHECKR,ASB      ;GOOD DATA BYTE
001406' 005267  000566                  INC     CHECKR          ;UPDATE FOR NEXT TEST
001412' 005367  000554                  DEC     COUNT           ;ONE MORE BYTE HAS BEEN TESTED
                                        ;*****************************************************************
```



```
(1)  001416' 104405  000000'            DATER,,BEGIN            ;DATA ERROR!!!
(1)                                     ;*****************************************************************


001422' 005767  000544          RESTOR: TST     COUNT           ;ARE WE DONE CHECKING DATA ON
                                                                ;ON THIS LINE
001426' 001324                          BNE     CONTNU          ;NOG GO DO THE REST OF THIS LINE
001430' 005267  000534                  INC     NODVTS          ;YES THIS LINE DONE ADD 1 TO
                                                                ;NODVTS>NO, OF LINES TESTED
001434' 012711  100777                  MOV     #100777,(R1)
001440' 022767  000010  000522          CMP     #10,NODVTS      ;HAVE ALL LINES BEEN TESTED
001446' 001401                          BEQ     PASS            ;GO TO END PASS CODE
001450' 000674                          BR      CHCK1                   ;RETURN TO MONITOR


001452' 005367  000516          PASS:   DEC     PASCT           ;IS THIS LAST PASS
001456' 001402                          BEQ     1$              ;DONE EXIT
001460' 000167  176476                  JMP     START           ;NO GO DO ONE MORE
001464' 012767  000100  000502  1$:     MOV     #100,PASCT              ;SET NO, OF PASSES
001472' 104402  000162' 000000'         ENDPS,,START,BEGIN      ;SIGNAL END OF PASS, RESUME AT START


                                ; SYNC WORD STORAGE LOCATIONS (LINE SYNC <1:8>)


001500' 000000                  DPLIN:  0
001502' 000000                          0
001504' 000000                          0
001506' 000000                          0
001510' 000000                          0
001512' 000000                          0
001514' 000000                          0
001516' 000000                          0


001520' 000000                  DVAD1:  0
001522' 000000                          0
001524' 000000                          0
001526' 000000                          0
001530' 000000                          0
001532' 000000                          0
001534' 000000                          0
001536' 000000                          0


001540' 000000                  LNSYN1: OPEN                    ;HIGH BYTE=SYNC COUNT NO,
                                                                ;LOW BYTE =BINARY WORD PATTERN
001542' 000000                          OPEN
001544' 000000                          OPEN
001546' 000000                          OPEN
001550' 000000                          OPEN
001552' 000000                          OPEN
```

```
  661  001554' 000000                        OPEN
  662  001556' 000000                        OPEN
  663
  664
  665  001560' 000000                LNCNT1: OPEN                    ;HIGH BYTE=NO. XMTED INTERRUPTS
                                                                       ;LOW BYTE = NO. RCV. INTERRUPTS
  666
  667  001562' 000000                        OPEN
  668  001564' 000000                        OPEN
  669  001566' 000000                        OPEN
  670  001570' 000000                        OPEN
  671  001572' 000000                        OPEN
  672  001574' 000000                        OPEN
  673  001576' 000000                        OPEN
  674
  675
  676  001600' 000000                VRFLG1: 0                       ;BYTE OFFSET VALUE FOR READ
  677  001602' 000000                        0                       ;
  678  001604' 000000                        0                       ;
  679  001606' 000000                        0                       ;
  680  001610' 000000                        0                       ;
  681  001612' 000000                        0                       ;
  682  001614' 000000                        0                       ;
  683  001616' 000000                        0                       ;
  684
  685
  686                                ;RECEIVE DATA 16 BYTES PER,BUFFER
  687
  688  001620' 000000                DPLIN1: OPEN                    ;DP11 LINE #1 RECEIVE
  689  001622' 000000                        OPEN                    ;  DATA BUFFER,
  690  001624' 000000                        OPEN
  691  001626' 000000                        OPEN
  692  001630' 000000                        OPEN
  693  001632' 000000                        OPEN
  694  001634' 000000                        OPEN
  695  001636' 000000                        OPEN
  696
  697  001640' 000000                DPLIN2: OPEN                    ;DP11 LINE #2 RECEIVE
  698  001642' 000000                        OPEN                    ;  DATA BUFFER,
  699  001644' 000000                        OPEN
  700  001646' 000000                        OPEN
  701  001650' 000000                        OPEN
  702  001652' 000000                        OPEN
  703  001654' 000000                        OPEN
  704  001656' 000000                        OPEN
  705  001660' 000000                DPLIN3: OPEN                    ;DP11 LINE #3 RECEIVE
  706  001662' 000000                        OPEN                    ; TRANSMIT DATA BUFFER,
  707  001664' 000000                        OPEN
  708  001666' 000000                        OPEN
  709  001670' 000000                        OPEN
  710  001672' 000000                        OPEN
  711  001674' 000000                        OPEN
  712  001676' 000000                        OPEN
  713  001700' 000000                DPLIN4: OPEN                    ;DP11 LINE #4 RECEIVE
  714  001702' 000000                        OPEN                    ;  DATA BUFFER
```



```
  715  001704' 000000                        OPEN
  716  001706' 000000                        OPEN
  717  001710' 000000                        OPEN
  718  001712' 000000                        OPEN
  719  001714' 000000                        OPEN
  720  001716' 000000                        OPEN
  721  001720' 000000                DPLIN5: OPEN                    ;DP11 LINE #5 RECEIVE
  722  001722' 000000                        OPEN                    ;  DATA BUFFER
  723  001724' 000000                        OPEN
  724  001726' 000000                        OPEN
  725  001730' 000000                        OPEN
  726  001732' 000000                        OPEN
  727  001734' 000000                        OPEN
  728  001736' 000000                        OPEN
  729  001740' 000000                DPLIN6: OPEN                    ;DP11 LINE #6 RECEIVE
  730  001742' 000000                        OPEN                    ;  DATA BUFFER
  731  001744' 000000                        OPEN
  732  001746' 000000                        OPEN
  733  001750' 000000                        OPEN
  734  001752' 000000                        OPEN
  735  001754' 000000                        OPEN
  736  001756' 000000                        OPEN
  737  001760' 000000                DPLIN7: OPEN                    ;DP11 LINE #7 RECEIVE
  738  001762' 000000                        OPEN                    ;  DATA BUFFER
  739  001764' 000000                        OPEN
  740  001766' 000000                        OPEN
  741  001770' 000000                        OPEN
  742  001772' 000000                        OPEN
  743  001774' 000000                        OPEN
  744  001776' 000000                        OPEN
  745  002000' 000000                DPLIN8: OPEN                    ;DP11 LINE #8 RECEIVE
  746  002002' 000000                        OPEN                    ;  DATA BUFFER
  747  002004' 000000                        OPEN
  748  002006' 000000                        OPEN
  749  002010' 000000                        OPEN
  750  002012' 000000                        OPEN
  751  002014' 000000                        OPEN
  752  002016' 000000                        OPEN
  753
  754
  755                                ; SERVICE CODE FOR LINKING A PARTICULER DEVICE
  756                                ; TO A COMMON XMT OR RCV SERVICE ROUTINE.
  757  002020' 004567  176722  LINKER: JSR     R5,DPRCV                ;ANSWER LINE 1 RCV INTR
  758  002024' 000000                 0                               ;OFFSET FOR LINE 1
  759  002026' 004567  176530         JSR     R5,DPXMT                ;ANSWER LINE 1 XMT INTR
  760  002032' 000000                 0                               ;OFFSET FOR LINE 1
  761  002034' 004567  176706         JSR     R5,DPRCV                ;ANSWER LINE 2 RCV INTR
  762  002040' 000002                 2                               ;OFFSET FOR LINE 2
  763  002042' 004567  176514         JSR     R5,DPXMT                ;ANSWER LINE 2 XMT INTR
  764  002046' 000002                 2                               ;OFFSET FOR LINE 2
  765  002050' 004567  176672         JSR     R5,DPRCV                ;ANSWER LINE 3 RCV INTR
  766  002054' 000004                 4                               ;OFFSET FOR LINE 3
  767  002056' 004567  176500         JSR     R5,DPXMT                ;ANSWER LINE 3 XMT INTR
  768  002062' 000004                 4                               ;OFFSET FOR LINE 3
```

```
 769  002064' 004567  176656                JSR     R5,DPRCV        ;ANSWER LINE 4 RCV INTR
 770  002070' 000006                          4                       ;OFFSET FOR LINE 4
 771  002072' 004567  176464                JSR     R5,DPXMT        ;ANSWER LINE 4 XMT INTR
 772  002076' 000006                          6                       ;OFFSET FOR LINE 4
 773  002100' 004567  176642                JSR     R5,DPRCV        ;ANSWER LINE 5 RCV INTR
 774  002104' 000010                          10                      ;OFFSET FOR LINE 5
 775  002106' 004567  176450                JSR     R5,DPXMT        ;ANSWER LINE 5 XMT INTR
 776  002112' 000010                          10                      ;OFFSET FOR LINE 5
 777  002114' 004567  176626                JSR     R5,DPRCV        ;ANSWER LINE 6 RCV INTR
 778  002120' 000012                          12                      ;OFFSET FOR LINE 6
 779  002122' 004567  176434                JSR     R5,DPXMT        ;ANSWER LINE 6 XMT INTR
 780  002126' 000012                          12                      ;OFFSET FOR LINE 6
 781  002130' 004567  176612                JSR     R5,DPRCV        ;ANSWER LINE 7 RCV INTR
 782  002134' 000014                          14                      ;OFFSET FOR LINE 7
 783  002136' 004567  176420                JSR     R5,DPXMT        ;ANSWER LINE 7 XMT INTR
 784  002142' 000014                          14                      ;OFFSET FOR LINE 7
 785  002144' 004567  176576                JSR     R5,DPRCV        ;ANSWER LINE 8 RCV INTR
 786  002150' 000016                          16                      ;OFFSET FOR LINE 8
 787  002152' 004567  176404                JSR     R5,DPXMT        ;ANSWER LINE 8 XMT INTR
 788  002156' 000016                          16                      ;OFFSET FOR LINE 8
 789
 790  002160' 000000              PNTR:   OPEN                    ;PNTR REG TO TEST DEVICE ON LINE
 791  002162' 013426              TSYNC:  13426                   ;SYNC CODE
 792  002164' 000000              CNT80:  OPEN                    ;USED FOR COUNTER OF 64,
 793  002166' 000000              DVIDA:  OPEN                    ;POINTER FLAG WHICH WILL BRANCH TO
 794                                                              ;TEST STATUS OF ALL LINES AFTER
 795                                                              ;COMPLETING ONE LINE DATA TRANSFER
 796  002170' 000000              NODVTS: OPEN                    ;WHEN =8 ALL LINES HAVE BEEN TESTED
 797  002172' 000000              COUNT:  OPEN                    ;COUNTS DOWN FROM 16 WHEN CHECKING
 798                                                              ;DATA BUFFER REG,
 799  002174' 000100              PASCT:100                       ;USED TO INCREASE NO, OF PASSES
 800  002176' 000000              RCVDAT: 0                       ;WORD USED TO INCRAMENT XMTED DATA
 801  002200' 000000              CHECKR: 0                       ;STORES WORD BEING CHECKED
 802          000001                      .END
```



```
ACSR      000052R       ADDR      000006R       ASB       000056R       ASTAT     000054R
AWAS      000060R       BDCNV   = ******  G     BEGIN     000000R       BIT0    = 000001
BIT1    = 000002        BIT10   = 002000        BIT11   = 004000        BIT12   = 010000
BIT13   = 020000        BIT14   = 040000        BIT15   = 100000        BIT2    = 000004
BIT3    = 000010        BIT4    = 000020        BIT5    = 000040        BIT6    = 000100
BIT7    = 000200        BIT8    = 000400        BIT9    = 001000        BREAK,  = 104407
BR1       000012R       BR2       000013R       B$        000552R       CHCK      001234R
CHCK1     001242R       CHCK2     001266R       CHECKR    002200R       CNT80     002164R
CONTNU    001300R       COUNT     002172R       CSRA      000050R       C$        000452R
DATER,  = 104405        DPLIN     001500R       DPLIN1    001620R       DPLIN2    001640R
DPLIN3    001660R       DPLIN4    001700R       DPLIN5    001720R       DPLIN6    001740R
DPLIN7    001760R       DPLIN8    002000R       DPRCV     000746R       DPRCV1    001022R
DPXMT     000562R       DPXMT1    000634R       DPXMT2    000706R       DPXMT3    000672R
DVAD1     001520R       DVIDA     002166R       DVID1     000014R       D$        000432R
EABITS  = ******  G     ENDPS,  = 104402        END,    = 104403        ERRCNT    000030R
ERRN,   = 104410        ERROR,  = 104404        ERRRT     001362R       EXIT,   = 104400
FIRER     001014R       FITER     000626R       HICORE  = ******  G     INIT      000022R
INT       000324R       KCKOFF    000476R       LINKER    002020R       LNCNT1    001560R
LNSYN1    001540R       LOCORE  = ******  G     MODNAM    000000R       MODSP     000162R
MSGN,   = 104411        MSG,    = 104406        NODVTS    002170R       OACNV   = ******  G
OPEN    = 000000        OVERR     001110R       PASCNT    000026R       PASCT     002174R
PASS      001452R       PC      =%000007        PIRQ,   = 000004        PNTR      002160R
POPSP   = 005726        POPSP2  = 022626        PRTY    = 000000        PRTY0   = 000000
PRTY1   = 000040        PRTY2   = 000100        PRTY3   = 000140        PRTY4   = 000200
PRTY5   = 000240        PRTY6   = 000300        PRTY7   = 000340        PS      = 177776
PSW     = 177776        PUSH    = 005746        PUSH2   = 024646        QUE,    = 104401
RCVDAT    002176R       RCVRTN    001222R       READ      001116R       RESTOR    001422R
R0      =%000000        R1      =%000001        R2      =%000002        R3      =%000003
R4      =%000004        R5      =%000005        R6      =%000006        R7      =%000007
SBADR     000052R       SP      =%000006        SPOINT    000024R       SPSIZ   = 000040
SR1       000016R       START     000162R       START1    000302R       START2    000374R
STAT      000020R       SVR0      000032R       SVR1      000034R       SVR2      000036R
SVR3      000040R       SVR4      000042R       SVR5      000044R       SVR6      000046R
TPX     = 000000        TRAPX   = 000012        TSYNC     002162R       UPDAT     000530R
VECTOR    000010R       VRFLG1    001600R       WASADR    000054R       WBUF    = ******  G
X         001170R       XMTRTN    000736R       .       = 002202R

          002202
```

ERRORS DETECTED: 0

```
*XDPAR,XDPAR,PRT_DCXCOM.P11,XDPAB.P11
RUN-TIME:  3  5  0 SECONDS
CORE USED:  4K
```



```
    1
  213
                        .REM -
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DXXYA-B-D
PRODUCT NAME:	XY11 DEC/X11 MODULE
DATE:	JUNE 15,1973
MAINTAINER:	COMPUTER SPECIAL SYSTEMS
AUTHOR:	ROBERT J. COLLINS

1. ABSTRACT:

XYA IS AN IOMOD THAT EXERCISES THE XY11 PLOTTER INTERFACE.
A SQUARE WITH CROSSED CENTER LINES IS CONTINUOUSLY DRAWN
AS THE PAPER ROLL ADVANCES.

2. REQUIREMENTS:

HARDWARE: XY11 INTERFACE WITH ITS ASSOCIATED PLOTTER.

STORAGE: XYA REQUIRES 632(10) WORDS OF STORAGE

3. PASS DEFINITION:

EACH COMPLETE FIGURE CONSTITUTES A PASS OF XYA.

4. EXECUTION TIME:

XYA RUNNING ALONE ON A PDP11/05 PROCESSOR TAKES
APPROXIMATELY---MINUTES TO COMPLETE ONE PASS.

5. CONFIGURATION REQUIREMENTS:

DEFAULT PARAMETERS:

DEVADR: 172554, VECTOR: 120, BR: 5

REQUIRED PARAMETERS:

NONE

6. DEVICE/OPTION SETUP:

A. TURN PLOTTER POWER AND DRUM DRIVE ON.
B. MANUALLY POSITION THE PEN TO THE LEFT MARGIN.

7. MODULE OPERATION:

A. SETUP THE XY11 REGISTER ADRESSES
B. RAISE THE PEN AND FIND THE LEFT MARGIN.
C. DRAW A SQUARE.
D. DRAW A CROSS WITHIN THE SQUARE
E. SPACE UP THE PAPER A DISTANCE ONE HALF THE SQUARE SIZE.
F. REPEAT FROM 7.B

8. OPERATION OPTIONS:

MODULE LOCATION STEPS (XYA 1154) MAY BE USED TO CHANGE THE
SIZE OF THE FIGURE.

9. NON-STANDARD PRINTOUTS:

NONE

```
000000'          IDMOD   <XXYAB >,172554,120,5
000000'          MODULE  140000,XXYAB ,172554,120,5,,
                 .TITLE  XXYAB DEC/X11 SYSTEM EXERCISER MODULE
                         .LIST   BIN
                         ;*****************************************************************************
000000'                  BEGIN:
000000' 054130  040531  020102  MODNAM: .ASCII  /XXYAB /          ;MODULE NAME.
000006' 172554           ADDR:   172554+0                        ;1ST DEVICE ADDR.
000010' 000120           VECTOR: 120+0                           ;1ST DEVICE VECTOR.
000012'     240          BR1:    .BYTE   PRTY5+0                 ;1ST BR LEVEL.
000013'     000          BR2:    .BYTE   PRTY+0                  ;2ND BR LEVEL.
000014' 000001           DVID1:  +1                              ;DEVICE INDICATOR 1.
000016' 000000           SR1:    OPEN                            ;SWITCH REGISTER 1
                         ;*****************************************************************************
000020' 140000           STAT:   140000                          ;STATUS WORD.
000022' 000162'          INIT:   START                           ;MODULE START ADDR.
000024' 000162'          SPOINT: MODSP                           ;MODULE STACK POINTER.
000026' 000000           PASCNT: 0                               ;PASS COUNTER.
000030' 000C00           ERRCNT: 0                               ;ERROR COUNTER.
000032' 000000           SVR0:   OPEN                            ;LOC TO SAVE R0.
000034' 000000           SVR1:   OPEN                            ;LOC TO SAVE R1.
000036' 000000           SVR2:   OPEN                            ;LOC TO SAVE R2.
000040' 000000           SVR3:   OPEN                            ;LOC TO SAVE R3.
000042' 000000           SVR4:   OPEN                            ;LOC TO SAVE R4.
000044' 000000           SVR5:   OPEN                            ;LOC TO SAVE R5.
000046' 000000           SVR6:   OPEN                            ;LOC TO SAVE R6.
000050' 000000           CSRA:   OPEN                            ;ADDR OF CURRENT CSR.
000052'                  SBADR:                                  ;ADDR OF GOOD DATA, OR
000052' 000000           ACSR:   OPEN                            ;CONTENTS OF CSR.
000054'                  WASADR:                                 ;ADDR OF BAD DATA, OR
000054' 000000           ASTAT:  OPEN                            ;STATUS REG CONTENTS.
000056' 000000           ASB:    OPEN                            ;EXPECTED DATA.
000060' 000000           AWAS:   OPEN                            ;ACTUAL DATA.
                                 .REPT   SPSIZ                   ;MODULE STACK STARTS HERE.
                                 .NLIST
                                 .WORD   0
                                 .LIST
                                 .ENDR
000162'                  MODSP:
                         ;*****************************************************************************
```



```
 316  000162' 012767  000012  001014  START:  MOV     #10.,PASCTR     ;WILL DO SEQUENCE 10 TIMES PER PASS.
 317  000170' 016767  177612  001016           MOV     ADDR,XYCS       ;LOAD XY11 CSR ADDRESS
 318  000176' 016767  177604  001012           MOV     ADDR,XYDB       ;LOAD XY11 DBR ADDRESS
 319  000204' 062767  000002  001004           ADD     #2,XYDB
 320  000212' 016700  177572                   MOV     VECTOR,R0       ;SETUP TO LOAD XY11 PI INFO
 321  000216' 012720  000242'                  MOV     #STP1,(R0)+     ;LOAD PI VECTOR
 322  000222' 016720  177564                   MOV     BR1,(R0)+       ;LOAD BR LEVEL
 323  000226' 012777  000100  000760           MOV     #100,@XYCS      ;ENABLE PI
 324  000234' 005077  000756                   CLR     @XYDB           ;RAISE DUMMY INTERRUPT
 325  000240' 104400                           EXIT.
 326
 327  000242'                          STP1:
 (1)                                       ;*****************************************************************
 (1)  000242' 000004  000250' 000000'      PIRQ.,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
 (1)                                       ;*****************************************************************
 328  000250' 012777  000274' 177532  1$:   MOV     #STP2,@VECTOR   ;CHANGE PI VECTOR
 329  000256' 012767  000062  000726        MOV     #62,COUNT       ;SET COUNTER
 330  000264' 012777  000040  000724        MOV     #40,@XYDB       ;PEN UP
 331  000272' 104400                        EXIT.
 332
 333  000274'                          STP2:
 (1)                                       ;*****************************************************************
 (1)  000274' 000004  000302' 000000'      PIRQ.,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
 (1)                                       ;*****************************************************************
 334  000302' 012777  000320' 177500  1$:   MOV     #STP2A,@VECTOR  ;CHANGE PI VECTOR
 335  000310' 012777  000010  000700        MOV     #10,@XYDB       ;PEN RIGHT
 336  000316' 104400                        EXIT.
 337
 338  000320'                          STP2A:
 (1)                                       ;*****************************************************************
 (1)  000320' 000004  000326' 000000'      PIRQ.,2$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 2$
 (1)                                       ;*****************************************************************
 339  000326' 005367  000660           2$:  DEC     COUNT           ;DONE?
 340  000332' 001404                        BEQ     STP3            ;SKIP IF YES
 341  000334' 012777  000010  000654        MOV     #10,@XYDB       ;NO= PEN RIGHT
 342  000342' 104400                        EXIT.
 343
 344  000344' 012777  000362' 177436  STP3:  MOV     #STP4,@VECTOR   ;CHANGE PI VECTOR
 345  000352' 012777  000020  000636        MOV     #20,@XYDB       ;PEN DOWN
 346  000360' 104400                        EXIT.
 347
 348  000362'                          STP4:
 (1)                                       ;*****************************************************************
 (1)  000362' 000004  000370' 000000'      PIRQ.,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
 (1)                                       ;*****************************************************************
 349  000370' 012777  000414' 177412  1$:   MOV     #STP5,@VECTOR   ;CHANGE PI VECTOR
 350  000376' 016767  000604  000606        MOV     STEP5,COUNT     ;LOAD COUNT
 351  000404' 012777  000010  000604        MOV     #10,@XYDB       ;PEN RIGHT
 352  000412' 104400                        EXIT.
 353
```

```
                                        STP5:
(1)
(1)  000414' 000004  000422' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
000422' 005367  000564          1$:     DEC     COUNT           ;DONE?
000426' 001404                          BEQ     STP6            ;SKIP IF YES
000430' 012777  000010  000560          MOV     #10,@XYDB       ;NO- PEN RIGHT
000436' 104400                          EXIT.

000440' 012777  000464' 177342  STP6:   MOV     #STP7,@VECTOR   ;CHANGE PI VECTOR
000446' 016767  000534  000536          MOV     STEPS,COUNT     ;LOAD COUNT
000454' 012777  000001  000534          MOV     #1,@XYDB        ;DRUM DOWN
000462' 104400                          EXIT.

000464'                                 STP7:
(1)
(1)  000464' 000004  000472' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
000472' 005367  000514          1$:     DEC     COUNT           ;DONE?
000476' 001404                          BEQ     STP10           ;YES- SKIP
000500' 012777  000001  000510          MOV     #1,@XYDB        ;NO- DRUM DOWN
000506' 104400                          EXIT.

000510' 012777  000534' 177272  STP10:  MOV     #STP11,@VECTOR  ;CHANGE PI VECTOR
000516' 016767  000464  000466          MOV     STEPS,COUNT     ;LOAD COUNTER
000524' 012777  000004  000464          MOV     #4,@XYDB        ;PEN LEFT
000532' 104400                          EXIT.

000534'                                 STP11:
(1)
(1)  000534' 000004  000542' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
000542' 005367  000444          1$:     DEC     COUNT           ;DONE?
000546' 001404                          BEQ     STP12           ;SKIP IF YES
000550' 012777  000004  000440          MOV     #4,@XYDB        ;NO- PEN LEFT
000556' 104400                          EXIT.

000560' 012777  000604' 177222  STP12:  MOV     #STP13,@VECTOR  ;CHANGE PI VECTOR
000566' 016767  000414  000416          MOV     STEPS,COUNT     ;LOAD COUNTER
000574' 012777  000002  000414          MOV     #2,@XYDB        ;DRUM UP
000602' 104400                          EXIT.

000604'                                 STP13:
(1)
(1)  000604' 000004  000612' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
000612' 005367  000374          1$:     DEC     COUNT           ;DONE?
000616' 001404                          BEQ     STP14           ;SKIP IF YES
000620' 012777  000002  000370          MOV     #2,@XYDB        ;NO- DRUM UP
000626' 104400                          EXIT.
```



```
000630' 012777  000654' 177152  STP14:  MOV     #STP15,@VECTOR  ;CHANGE PI VECTOR
000636' 016767  000344  000346          MOV     STEPS,COUNT     ;LOAD COUNT
000644' 012777  000011  000344          MOV     #11,@XYDB       ;DRUM DOWN AND PEN RIGHT
000652' 104400                          EXIT.

000654'                                 STP15:
(1)
(1)  000654' 000004  000662' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
000662' 005367  000324          1$:     DEC     COUNT           ;DONE?
000666' 001404                          BEQ     STP16           ;SKIP IF YES
000670' 012777  000011  000320          MOV     #11,@XYDB       ;NO- DRUM DOWN AND PEN RIGHT
000676' 104400                          EXIT.

000700' 012777  000716' 177102  STP16:  MOV     #STP17,@VECTOR  ;CHANGE PI VECTOR
000706' 012777  000040  000302          MOV     #40,@XYDB       ;PEN UP
000714' 104400                          EXIT.

000716'                                 STP17:
(1)
(1)  000716' 000004  000724' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
000724' 012777  000750' 177056  1$:     MOV     #STP20,@VECTOR  ;CHANGE PI VECTOR
000732' 016767  000250  000252          MOV     STEPS,COUNT     ;LOAD COUNTER
000740' 012777  000002  000250          MOV     #2,@XYDB        ;DRUM UP
000746' 104400                          EXIT.

000750'                                 STP20:
(1)
(1)  000750' 000004  000756' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
000756' 005367  000230          1$:     DEC     COUNT           ;DONE?
000762' 001404                          BEQ     STP21           ;SKIP IF YES
000764' 012777  000002  000224          MOV     #2,@XYDB        ;NO- DRUM UP
000772' 104400                          EXIT.

000774' 012777  001012' 177006  STP21:  MOV     #STP22,@VECTOR  ;CHANGE PI VECTOR
001002' 012777  000020  000206          MOV     #20,@XYDB       ;PEN DOWN
001010' 104400                          EXIT.

001012'                                 STP22:
(1)
(1)  001012' 000004  001020' 000000'            ;-------------------------------------------------------------
(1)                                             PIRQ,,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
                                                ;-------------------------------------------------------------
001020' 012777  001044' 176762  1$:     MOV     #STP23,@VECTOR  ;CHANGE PI VECTOR
001026' 016767  000154  000156          MOV     STEPS,COUNT     ;LOAD COUNTER
001034' 012777  000005  000154          MOV     #5,@XYDB        ;DRUM DOWN AND PEN LEFT
001042' 104400                          EXIT.
```

```
  433  001044'                          STP23:
  (1)
  (1)  001044' 000004  001052' 000000'          PIRQ.,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
  (1)
  434  001052' 005367  000134           1$:     DEC     COUNT           ;DONE?
  435  001056' 001404                           BEQ     STP24           ;SKIP IF YES
  436  001060' 012777  000005  000130           MOV     #5,@XYDB        ;NO- DRUM DOWN AND PEN LEFT
  437  001066' 104400                           EXIT.
  438
  439  001070' 012777  001106' 176712  STP24:  MOV     #STP25,@VECTOR  ;CHANGE PI VECTOR
  440  001076' 012777  000040  000112           MOV     #40,@XYDB       ;PEN UP
  441  001104' 104400                           EXIT.
  442
  443  001106'                          STP25:
  (1)
  (1)  001106' 000004  001114' 000000'          PIRQ.,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
  (1)
  444  001114' 012777  001140' 176666  1$:     MOV     #STP26,@VECTOR  ;CHANGE PI VECTOR
  445  001122' 016767  000062  000062           MOV     HALF,COUNT      ;LOAD COUNTER
  446  001130' 012777  000001  000060           MOV     #1,@XYDB        ;DRUM DOWN
  447  001136' 104400                           EXIT.
  448
  449  001140'                          STP26:
  (1)
  (1)  001140' 000004  001146' 000000'          PIRQ.,1$,BEGIN  ;QUEUE REQUEST TO CONTINUE AT 1$
  (1)
  450  001146' 005367  000040           1$:     DEC     COUNT           ;DONE?
  451  001152' 001404                           BEQ     STP27           ;YES- REDRAW PATTERN
  452  001154' 012777  000001  000034           MOV     #1,@XYDB        ;NO- DRUM DOWN
  453  001162' 104400                           EXIT.
  454
  455  001164' 005367  000014           STP27:  DEC     PASCTR          ;DONE REQUIRED TIMES?
  456  001170' 001402                           BEQ     1$              ;BR IF YES.
  457  001172' 000167  177146                   JMP     STP3            ;NO GO DO IT AGAIN.
  458  001176'                          1$:
  (1)  001176' 104402  000162' 000000'          ENDPS.,.START,BEGIN     ;SIGNAL END OF PASS, RESUME AT START
  459
  460  001204' 000000                   PASCTR: OPEN
  461  001206' 000454                   STEPS:  300.
  462  001210' 000226                   HALF:   150.
  463  001212' 000000                   COUNT:  0
  464  001214' 000000                   XYCS:   0
  465  001216' 000000                   XYDB:   0
  466
  467          000001                           .END                    ;THAT'S ALL FOLKS!
```



```
ACSR     000052R        ADDR     000006R        ASB      000056R        ASTAT    000054R
AWAS     000060R        BDCNV  = ****** G       BEGIN    000000R        BIT0   = 000001
BIT1   = 000002         BIT10  = 002000         BIT11  = 004000         BIT12  = 010000
BIT13  = 020000         BIT14  = 040000         BIT15  = 100000         BIT2   = 000004
BIT3   = 000010         BIT4   = 000020         BIT5   = 000040         BIT6   = 000100
BIT7   = 000200         BIT8   = 000400         BIT9   = 001000         BREAK, = 104407
BR1      000012R        BR2      000013R        COUNT    001212R        CSRA     000050R
DATER, = 104405         DVID1    000014R        EABITS = ****** G       ENDPS, = 104402
END,   = 104403         ERRCNT   000030R        ERRN,  = 104410         ERROR, = 104404
EXIT,  = 104400         HALF     001210R        HICORE = ****** G       INIT     000022R
LOCORE = ****** G       MODNAM   000000R        MODSP    000162R        MSGN,  = 104411
MSG,   = 104406         OACNV  = ****** G       OPEN   = 000000         PASCNT   000026R
PASCTR   001204R        PC     =%000007         PIRQ,  = 000004         POPSP  = 005726
POPSP2 = 022626         PRTY   = 000000         PRTY0  = 000080         PRTY1  = 000040
PRTY2  = 000100         PRTY3  = 000140         PRTY4  = 000200         PRTY5  = 000240
PRTY6  = 000300         PRTY7  = 000340         PS     = 177776         PSW    = 177776
PUSH   = 005746         PUSH2  = 024646         QUE,   = 104401         R0     =%000000
R1     =%000001         R2     =%000002         R3     =%000003         R4     =%000004
R5     =%000005         R6     =%000006         R7     =%000007         SBADR    000052R
SP     =%000006         SPOINT   000024R        SPSIZ  = 000040         SR1      000016R
START    000162R        STAT     000020R        STEPS    001206R        STP1     000242R
STP10    000510R        STP11    000534R        STP12    000560R        STP13    000604R
STP14    000630R        STP15    000654R        STP16    000700R        STP17    000716R
STP2     000274R        STP2A    000320R        STP20    000750R        STP21    000774R
STP22    001012R        STP23    001044R        STP24    001070R        STP25    001106R
STP26    001140R        STP27    001164R        STP3     000344R        STP4     000362R
STP5     000414R        STP6     000440R        STP7     000464R        SVR0     000032R
SVR1     000034R        SVR2     000036R        SVR3     000040R        SVR4     000042R
SVR5     000044R        SVR6     000046R        TPX    = 000000         TRAPX  = 000012
VECTOR   000010R        WASADR   000054R        WBUF   = ****** G       XYCS     001214R
XYDB     001216R        .      = 001220R

         001220
```

ERRORS DETECTED: 0