## IDENTIFICATION

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

PRODUCT NAME: DEC/X11 DECNET LIBRARY

DATE: MARCH 1977

MAINTAINER: DIAGNOSTIC GROUP

CONTENTS
---


| | |
|---|---|
| N1AB | DEC/X11 DECNET SERVICE MODULE #1 |
| NUAA | DEC/X11 DECNET DU-11 MODULE |
| NQAA | DEC/X11 DECNET DQ-11 MODULE |
| NPBA | DEC/X11 DECNET DUP-11 MODULE |
| NLAA | DEC/X11 DECNET DL-11-E MODULE |
| N2AA | DEC/X11 DECNET SERVICE MODULE #2 |

```
                .REM %
```

## IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DXN1A-B-D

PRODUCT NAME: DECNET DEC/X11 SERVICE EXERCISER MODULE #1

DATE: MARCH 77

MAINTAINER: DEC/X11 DIAGNOSTIC GROUP

```
39
40                              N1A? IS A SERVICE MODULE USED BY DECNET DEC/X11
41      MODULES.  IT DOES NOT TEST ANY DEVICES.  IT MUST BE INCLUDED AS THE
42      FIRST MODULE, DIRECTLY AFTER THE MONITOR, IN THE CONFIGURING OF THE
43      RUN TIME EXERCISER.  ALL DECNET DEC/X11 MODULES MUST FOLLOW
44      IT AND THEN THE N2A? MODULE MUST FOLLOW THEM.  NEXT THE CLOCK MODULE
45      SHOULD FOLLOW AND THEN ALL NORMAL DEC/X11 MODULES IN ANY ORDER.
46      THIS MODULE IS USED TO HOLD SEVERAL COMMON SUBROUTINES AND ONE COMMON
47      TRANSMIT BUFFER WHICH ARE USED BY ALL DECNET DEC/X11 MODULES.
48      WHEN THE EXERCISER IS RUN, THIS MODULE WILL ASK QUESTIONS
49      REGARDING THE TYPES OF LINES TO BE RUN AND WHICH MESSAGE OPTION
50      YOU PREFER.
51      ************************************************************************
52                              IMPORTANT
53      EVERY NODE IN THE NETWORK BEING EXERCISED MUST USE THE EXACT SAME
54      MESSAGE!
55      ************************************************************************
56      THIS MODULE IS A SPECIAL NBKMOD SO IT WILL ONLY REPORT 1 END OF PASS
57      AND THEN NEVER RUN AGAIN.  IF THE OPERATOR DOES NOT WANT TO ANSWER
58      ALL THESE LINE QUESTIONS EVERY TIME HE RESTARTS THE EXERCISER, HE
59      CAN CHANGE LOCATION 164 TO BE A 000001.
60      HOWEVER, IF HE THEN LATER WANTS TO MODIFY A PARAMETER, HE WILL
61      HAVE TO MOD LOCATION 164 IN THE MODULE BACK TO A 000000.
62      WHILE ANSWERING THESE LINE QUESTIONS FOR AN INDIVIDUAL MODULE
63      (I.E. DEVICE TYPE) AN  "@"  REPLY WILL SKIP THE
64      REST OF THE QUESTIONS FOR THAT MODULE, LEAVING THE OTHER PARAMETERS
65      UNCHANGED.
66      THIS MODULE IS USED BY ALL DECNET DEC/X11 MODULES TO DO
67      DATA COMPARING OF RECEIVED MESSAGES.  WHEN IT FINDS AN ERROR FOR
68      A DECNET DEC/X11 MODULE, IT REPORTS IT, THEREFORE ITS NAME IN THE
69      RUN TIME EXERCISER IS INITIALLY N1XX0, BUT THIS NAME IS CHANGED EVERY
70      TIME IT REPORTS A DATA ERROR FOR A DECNET MODULE TO SHOW WHICH
71      DECNET MODULE REALLY HAS THE ERROR.  THEREFORE,
72      IN RUN SUMMARIES AND WHEN USING THE MOD COMMAND, THIS MODULE'S NAME
73      CHANGES.  TWO FIGURE OUT WHAT NAME IT HAS, CHECK THE LAST RUN SUMMARY
74      OR USE THE MAP COMMAND.  SEE THE DECNET DEC/X11 USER'S GUIDE, DXQ8C?,
75      FOR A MORE COMPLETE EXPAINATION.
76      %
```

```
                                        NBKMOD  <N1AB >
                                        MODULE  41000,N1AB ,,,,,
                                        .TITLE  N1AB DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION 4    9/6/75
                                        .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 030516  041101      040 MODNAM: .ASCII  /N1AB / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000000                  ADDR:   +0              ;1ST DEVICE ADDR.
000010' 000000                  VECTOR: +0                      ;1ST DEVICE VECTOR.
000012'     000                 BR1:    .BYTE   PRTY+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' 041000                  STAT:   41000                   ;STATUS WORD.
000022' 000166'                 INIT:   START                   ;MODULE START ADDR.
000024' 000164'                 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'                         BASADR:                         ;ADDR OF BAD DATA, OR
000054' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000056' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000060' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000062' 000166'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
        000040                          .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000164'                         MODSP:
                                ;*****************************************************************************
                                        .GLOBL  KW11X,KW11L,KW11P,CLOCK,TIME,FILCNT,FILLER
                                        .GLOBL  TKS,TKB,TPS,TPB,TYPE,MSGHED,MSG,MSGSIZ,WAIT
                                        .GLOBL  CRC,CKMSG,CRCMSG,DEVSET,SR,PWRCNT
                                ;
                                ;       IF THIS LOCATION IS NON-ZERO, THE DEVSET ROUTINE WILL
                                ;       NOT ASK OPERATOR FOR HALF/FULL DUPLEX SETUP, ETC.
                                ;
000164' 000000                  TFLAG:  0
                                ;
000166'                         START:
000166' 116700  000000G         RESTRT: MOVB    CLOCK,R0        ;FIND OUT WHICH CLOCK WE HAVE
000172' 006000                          ROR     R0              ;IS IT KW11L?
000174' 103004                          BCC     1$              ;BR IF NOT
```



```
000176' 012767  000002G 003214          MOV     #KW11L+2,TIME   ;ELSE LET TIME POINT TO "TIME"
000204' 000423                          BR      GOPT
000206' 006000                  1$:     ROR     R0              ;IS IT KW11P?
000210' 103004                          BCC     2$              ;BR IF NOT
000212' 012767  000002G 003200          MOV     #KW11P+2,TIME   ;ELSE LET TIME POINT TO "TIME"
000220' 000415                          BR      GOPT
000222' 006000                  2$:     ROR     R0              ;IS IT A FAKE CLOCK?
000224' 103004                          BCC     3$              ;BR IF NOT
000226' 012767  000002G 003164          MOV     #KW11X+2,TIME   ;ELSE LET TIME POINT TO "TIME"
000234' 000407                          BR      GOPT
000236' 012767  004241' 000004  3$:     MOV     #NOCLKM,4$      ;PUT MSG ADDR ON STACK
000244' 004567  000714                  JSR     R5,TYPE         ;CALL TYPE ROUTINE
000250' 000000                  4$:     OPEN
000252' 000000                          HALT                    ;CAN'T RUN WITH NO CLOCK
                                ;
                                ;       GET MESSAGE OPTION FROM THE USER.
                                ;       NOTE: ONLY THE LAST CHARACTER TYPED BEFORE "CR"
                                ;       IS SIGNIFICANT.  ALL PREVIOUS CHARACTERS (IF ANY)
                                ;       ARE IGNORED.
                                ;
000254' 005767  004210          GOPT:   TST     INFLAG          ;FIRST TIME THRU THIS MODULE?
000260' 001011                          BNE     2$              ;BR IF NOT.
000262' 032777  000001  000000G         BIT     #1,@SR          ;HAS THE OPERATOR DISABLED MSG'S?
000270' 001003                          BNE     1$              ;BR IF YES.
000272' 004567  000666                  JSR     R5,TYPE         ;TYPE ERROR CODES.
000276' 003445'                         INSG
000300' 005267  004164          1$:     INC     INFLAG          ;DON'T TYPE IT AGAIN.
000304' 005077  003110          2$:     CLR     @TIME           ;SET TIME TO ZERO
000310' 005067  001240                  CLR     CKMSG           ;SHOW THAT CKMSG ROUTINE ISN'T BUSY
000314' 005067  001570                  CLR     R4SAV   ;CLR R4 COUNTER
000320' 005067  001566                  CLR     R4SAV+2 ;CLEAR SAME FOR LINE 1
000324' 012767  004475' 000004          MOV     #GETMG,6$       ;PUT MSG ADR ON STACK
000332' 004567  000626                  JSR     R5,TYPE         ;CALL TYPE-OUT ROUTINE
000336' 000000                  6$:     OPEN
000340' 012700  003445'         GETMSG: MOV     #MSG+1,R0       ;GET BUFFER ADDR
000344' 017767  000000G 000720          MOV     @TKS,TKSSAV     ;SAVE TTY STATUS
000352' 005077  000000G                 CLR     @TKS            ;DIABLE INTERRUPTS
000356' 005001                          CLR     R1              ;SET MESSAGE SIZE TO ZERO
000360' 105777  000000G         1$:     TSTB    @TKS            ;WAIT FOR A CHARACTER
000364' 100375                          BPL     .-4
000366' 117702  000000G                 MOVB    @TKB,R2         ;BRING CHARACTER IN
000372' 042702  177600          2$:     BIC     #177600,R2      ;GET RID OF PARITY BIT
000376' 110267  004034                  MOVB    R2,OUTPUT       ;ONE CHARACTER BUFFER.
000402' 012767  004436' 000004          MOV     #OUTPUT,21$     ;USED IN THIS CASE TO ECHO
000410' 004567  000550                  JSR     R5,TYPE         ;THE INPUT CHARACTER.
000414' 000000                  21$:    OPEN
000416' 122702  000015                  CMPB    #15,R2          ;WAS IT A CR?
000422' 001402                          BEQ     3$              ;BR IF YES.
000424' 010203                          MOV     R2,R3           ;SAVE IT, DESTROYING PREVIOUS CHARACTER.
000426' 000754                          BR      1$              ;GET NEXT CHARACTER.
000430' 112767  000012  004000  3$:     MOVB    #12,OUTPUT      ;LINE FEED TO BUFFER.
000436' 012767  004436' 000004          MOV     #OUTPUT,4$      ;TYPE IT OUT.
000444' 004567  000514                  JSR     R5,TYPE
000450' 000000                  4$:     OPEN
000452' 120327  000060                  CMPB    R3,#60          ;MUST BE BETWEEN '0' AND '6'.
000456' 002411                          BLT     HELP            ;BR IF BELOW ASCII '0'.
```

```
000460' 120327  000066          CMPB    R3,#66          ;COMPARE TO ASCII '6'.
000464' 003006                  BGT     HELP            ;SEE IF IT'S A RUBOUT.
000466' 162703  000060          SUB     #60,R3          ;MAKE WORD OFFSET FROM ASCII.
000472' 000257                  CCC                     ;ASSURE CLEAR CARRY BIT.
000474' 006103                  ROL     R3              ;SO IT CAN BE MADE A BYTE OFFSET.
000476' 000173  004452'         JMP     @ODISP(R3)      ;USE IT TO DISPATCH TO PROPER OPTION ROUTINE.
000502' 012767  004574' 000004  HELP:   MOV     #HELPM,5$       ;ERROR IN INPUT, TYPE SOME INSTRUCTIONS.
000510' 004567  000450          JSR     R5,TYPE
000514' 000000                  5$:     OPEN
000516' 000656                  BR      GOPT            ;GO BACK AND TRY AGAIN.
                                ;
                                ;       FILL BUFFER WITH CHARACTERS SELECTED BY USER.
                                ;
000520' 005001                  ZEROS:  CLR     R1              ;FILL CHARACTER = 000.
000522' 000405                  BR      FILL
000524' 005001                  ONES:   CLR     R1              ;FILL CHARACTER = 377.
000526' 005101                  COM     R1
000530' 000402                  BR      FILL
000532' 112701  000252          ALTER:  MOVB    #252,R1         ;FILL CHARACTER = 252.
000536' 012700  003444'         FILL:   MOV     #MSG,R0         ;ADDRESS OF MESSAGE BUFFER.
000542' 105720                  TSTB    (R0)+           ;MOVE AROUND OSOP PROTOCOL CHARACTER.
000544' 022700  004434'         1$:     CMP     #CRCMSG,R0      ;FIRST CRC BYTE (END OF MSG BUFFER)?
000550' 001517                  BEQ     .DONE           ;BR IF YES.
000552' 110120                  MOVB    R1,(R0)+        ;STORE CHARACTER IN BUFFER.
000554' 000773                  BR      1$              ;CONTINUE UNTIL FULL.
                                ;
                                ;       WORST CASE BIT PATTERN.
                                ;
000556' 005000                  WORST:  CLR     R0              ;START WITH A ZERO CHARACTER.
000560' 005001                  CLR     R1              ;INITIALIZE HALF-BYTE TO ZERO.
000562' 012702  003445'         MOV     #MSG+1,R2       ;SET UP POINTER TO MESSAGE AREA.
000566' 022702  004434'         1$:     CMP     #CRCMSG,R2      ;END OF BUFFER AREA?
000572' 001506                  BEQ     DONE            ;BR IF YES.
000574' 110022                  MOVB    R0,(R2)+        ;MOVE BYTE INTO BUFFER.
000576' 005201                  INC     R1              ;UPDATE HALF-BYTE.
000600' 042701  177760          BIC     #177760,R1      ;CLEAR ALL BUT HALF-BYTE.
000604' 010100                  MOV     R1,R0           ;PUT IT INTO R0...
000606' 006300                  ASL     R0              ;AND SHIFT IT OVER.
000610' 006300                  ASL     R0
000612' 006300                  ASL     R0
000614' 006300                  ASL     R0
000616' 060100                  ADD     R1,R0           ;SECOND HALF-BYTE.
000620' 000762                  BR      1$              ;CONTINUE PROCESS UNTIL BUFFER FULL.
                                ;
                                ;       READ IN THE USER DEFINED MESSAGE.
                                ;
000622' 012767  005162' 000004  USER:   MOV     #USERM,1$       ;TYPE INSTRUCTIONS.
000630' 004567  000330          JSR     R5,TYPE
000634' 000000                  1$:     OPEN
000636' 012700  003445'         MOV     #MSG+1,R0       ;SET POINTER TO TOP OF MESSAGE BUFFER.
000642' 005077  000000G         CLR     @TKS            ;DISABLE TTY INTERRUPTS.
000646' 012701  000001          MOV     #1,R1           ;CHARACTER COUNT TO ONE (OSOP PROTOCOL).
000652' 105777  000000G         2$:     TSTB    @TKS            ;INPUT READY?
000656' 100375                  BPL     2$              ;NO.  WAIT FOR IT.
000660' 117702  000000G         MOVB    @TKB,R2         ;GET THE TYPED CHARACTER.
000664' 142702  000200          BICB    #200,R2         ;STRIP PARITY.
```



```
000670' 122702  000177          CMPB    #177,R2         ;IS IT A RUBOUT?
000674' 001006                  BNE     3$              ;SKIP IF NOT
000676' 005701                  TST     R1              ;ANY CHAR'S LEFT TO RUB OUT?
000700' 001755                  BEQ     1$              ;BR IF NOT, IGNORE RUBOUT
000702' 005301                  DEC     R1              ;MOVE CHAR COUNT BACK
000704' 005300                  DEC     R0              ;MOVE BUFFER POINTER BACK
000706' 111002                  MOVB    (R0),R2         ;SET UP TO TYPE OUT OLD CHARACTER
000710' 000405                  BR      4$
000712' 122702  000100          3$:     CMPB    #100,R2         ;WAS THE CHARACTER AN '@'?
000716' 001423                  BEQ     51$             ;BR IF YES.
000720' 110220                  MOVB    R2,(R0)+        ;STORE CHAR IN MSG
000722' 005201                  INC     R1              ;UP MSG SIZE
000724' 110267  003506          4$:     MOVB    R2,OUTPUT       ;MOV CHAR TO BUFFER
000730' 012767  004434' 000004          MOV     #OUTPUT,41$     ;PUT ADDR ON STACK
000736' 004567  000222          JSR     R5,TYPE         ;GO ECHO IT
000742' 000000                  41$:    OPEN
000744' 020127  000765          CMP     R1,#501.        ;ANY ROOM LEFT IN MSG?
000750' 001411                  BEQ     6$              ;BR IF NOT
000752' 120227  000015          CMPB    R2,#15          ;WAS CHAR A "CR"?
000756' 001335                  BNE     2$              ;BR IF NOT BACK TO PICK UP NEXT
000760' 012702  000012          MOV     #12,R2          ;PUT A LINE FEED INTO THE BUFFER.
000764' 000752                  BR      3$              ;PROCESS THE LF JUST LIKE ALL CHARACTERS.
000766' 010167  003446          51$:    MOV     R1,MSGSIZ       ;COUNT IS THE MESSAGE SIZE.
000772' 000411                  BR      FINISH          ;COMPLETE THE MODULES TASKS.
000774' 012767  004472' 000004  6$:     MOV     #CRLF,7$        ;FULL BUFFER, FORCE A CR AND LF.
001002' 004567  000156          JSR     R5,TYPE         ;TYPE 'EM.
001006' 000000                  7$:     OPEN
001010' 012767  000766  003422  DONE:   MOV     #502.,MSGSIZ    ;SET MESSAGE SIZE.
001016' 012767  000001  003416  FINISH: MOV     #1,WAIT         ;TELLS MODULES TO WAIT FOR NT2
001024' 016777  000242  000000G         MOV     TKSSAV,@TKS     ;RESTORE TTY STATUS TO WHAT IT WAS
001032' 116767  003402  002375          MOVB    MSGSIZ,COUNT    ;PUT LOW BITS OF COUNT IN HEADER
001040' 116767  003375  002370          MOVB    MSGSIZ+1,COUNT+1 ;HIGH BITS
001046' 152767  000300  002362          BISB    #300,COUNT+1    ;SET FLAG BITS IN HEADER
001054' 012767  003434' 000014          MOV     #MSGHED,8$+2    ;PUT HEADER ADDR ON STACK
001062' 012767  000006  000004          MOV     #6,8$   ;HEADER LENGTH
001070' 004567  000202          JSR     R5,CRC          ;GO CALC CRC
001074' 000000  000000          8$:     OPEN,OPEN
001100' 016767  177772  002334          MOV     8$+2,CRCHED     ;PUT CRC IN HEADER
001106' 012767  003444' 000014          MOV     #MSG,9$+2       ;PUT MSG ADDR ON STACK
001114' 016767  003320  000004          MOV     MSGSIZ,9$       ;MSG LENGTH
001122' 004567  000150          JSR     R5,CRC          ;GO CALCULATE CRC
001126' 000000  000000          9$:     OPEN,OPEN
001132' 016767  177772  003274          MOV     9$+2,CRCMSG     ;PUT MSG CRC IN GLOBAL LOC.
001140' 016701  003274          MOV     MSGSIZ,R1       ;GET SIZE OF MSG
001144' 116761  177760  003444'         MOVB    9$+2,MSG(R1)    ;PUT CRC AT END OF MSG
001152' 116761  177753  003445'         MOVB    9$+3,MSG+1(R1)
001160' 104402  000000'         ENDPS$,BEGIN            ;SIGNAL END OF PASS, CONTINUE AT RESTRT
                                ;
                                ;
                                ;
                                ; TYPE  -CALLED BY (A) PUTTING ADDR OF ASCIZ MSG AFTER JSR CALL
                                ;               (B) DO A JSR R5,TYPE
                                ;       DOES NOT DESTROY ANY REGISTERS
                                ;
001164' 010446                  TYPE:   MOV     R4,-(R6)        ;SAVE R4
001166' 010346                  MOV     R3,-(R6)        ;SAVE R3
```

```
001170' 012503                  MOV     (R5)+,R3        ;GET ADDR TO BE TYPED
001172' 017767  000000G 000074  MOV     @TPS,TPSSAV     ;SAVE TTY STATUS
001200' 005077  000000G         CLR     @TPS            ;DISABLE INTERRUPTS
001204' 105713          1$:     TSTB    (R3)            ;TERMINATOR?
001206' 001423          BEQ     4$              ;EXIT IF SO
001210' 122713  000012          CMPB    #12,(R3)        ;IS THE CHAR A LINE FEED?
001214' 001012          BNE     3$              ;NO, SKIP FILL CHAR'S
001216' 116704  000000G         MOVB    FILCNT,R4       ;GET THE FILL COUNT
001222' 116777  000000G 000000G 2$:     MOVB    FILLER,@TPB     ;TYPE A FILLER
001230' 105777  000000G         TSTB    @TPS            ;DONE YET?
001234' 100375          BPL     .-4             ;NO, WAIT
001236' 005304          DEC     R4              ;DEC COUNT
001240' 100370          BPL     2$              ;LOOP UNTIL 0
001242' 112377  000000G 3$:     MOVB    (R3)+,@TPB      ;LOAD AND TYPE CHAR
001246' 105777  000000G         TSTB    @TPS            ;IS PRINTER READY?
001252' 100375          BPL     .-4             ;WAIT TILL IT IS
001254' 000753          BR      1$              ;GET NEXT CHAR
001256' 012603          4$:     MOV     (R6)+,R3        ;RESTORE R3
001260' 012604          MOV     (R6)+,R4        ;RESTORE R4
001262' 016777  000006  000000G MOV     TPSSAV,@TPS     ;RESTORE TTY
001270' 000205          RTS     R5              ;RETURN
                        ;
001272' 000000          TKSSAV: 0
001274' 000000          TPSSAV: 0
                        ;
                        ;
                        ;CRC - SUBROUTINE
                        ;TO CALL PUT LENGTH, IN BYTES, AFTER JSR CALL
                        ;PUT ADDRESS OF FIRST BYTE NEXT
                        ;DO JSR R5,CRC
                        ;RETURNS WITH ONE WORD OF CRC-16 WHERE ADDRESS WAS
                        ;
                        ;
                        ;FOR EXAMPLE
                        ;       MOV     LENGHT,1$       ;MOVE LENGTH IN
                        ;       MOV     ADDR,1$+2       ;MAVE ADDRESS IN
                        ;       JSR     R5,CRC          ;GO CALCULATE
                        ;1$:    OPEN,OPEN
                        ;       MOV     1$+2,CRCLOC     ;PUT CRC WHERE YOU WANT IT
001276' 010067  000576  CRC:    MOV     R0,R0SAV        ;SAVE   R0
001302' 010167  000574          MOV     R1,R1SAV        ;       R1
001306' 010267  000572          MOV     R2,R2SAV        ;       R2
001312' 010367  000570          MOV     R3,R3SAV        ;       R3
001316' 010467  000566          MOV     R4,R4SAV        ;       R4
001322' 012501          MOV     (R5)+,R1        ;GET LENGTH IN REGISTER
001324' 011500          MOV     (R5),R0         ;       ADDR
001326' 005002          CLR     R2              ;CRC VALUE REGISTER
001330' 112003          1$:     MOVB    (R0)+,R3        ;CHAR TO ADD
001332' 010246          MOV     R2,-(SP)        ;XOR OLD CRC AND DATA CHAR
001334' 040316          BIC     R3,(SP)
001336' 040203          BIC     R2,R3
001340' 052603          BIS     (SP)+,R3
001342' 010304          MOV     R3,R4           ;SAVE VALUE
001344' 042703  177760          BIC     #177760,R3      ;CLEAR HIGH BITS
001350' 006303          ASL     R3              ;MAKE WORD INDEX
```



```
001352' 016303  001454'         MOV     CRCTAB(R3),R3   ;MODIFIER WORD
001356' 042704  177417          BIC     #177417,R4      ;LEAVE HIGH 4 BITS
001362' 006204          ASR     R4              ;MAKE WORD INDEX
001364' 006204          ASR     R4
001366' 006204          ASR     R4
001370' 016404  001514'         MOV     CRCTAB+32.(R4),R4 ;MODIFIER WORD
001374' 010446          MOV     R4,-(SP)        ;TOTAL MODIFIER
001376' 040316          BIC     R3,(SP)
001400' 040403          BIC     R4,R3
001402' 052603          BIS     (SP)+,R3
001404' 105002          CLRB    R2              ;CLEAR LOW BYTE OLD CRC
001406' 000302          SWAB    R2              ;MOVE HIGH TO LOW
001410' 010346          MOV     R3,-(SP)        ;XOR TOTAL MODIFIER AND OLD HIGH
001412' 040216          BIC     R2,(SP)
001414' 040302          BIC     R3,R2
001416' 052602          BIS     (SP)+,R2
001420' 005301          DEC     R1              ;DEC CHAR COUNT
001422' 001342          BNE     1$              ;AND GO BACK
001424' 010225          MOV     R2,(R5)+        ;SAVE CRC
001426' 016704  000456          MOV     R4SAV,R4        ;RESTORE  R4
001432' 016703  000450          MOV     R3SAV,R3        ;         R3
001436' 016702  000442          MOV     R2SAV,R2        ;         R2
001442' 016701  000434          MOV     R1SAV,R1        ;         R1
001446' 016700  000426          MOV     R0SAV,R0        ;         R0
001452' 000205          RTS     R5
                        ;
                        ;CRC TABLE OF MODIFIERS
001454' 000000          CRCTAB:     0
                                140301
                                140601
                                   500
                                141401
                                  1700
                                  1200
                                141101
                                143001
                                  3300
                                  3600
                                143501
                                  2400
                                142701
                                142201
                                  2100
                                     0
                                146001
                                154001
                                 12000
                                170001
                                 36000
                                 24000
                                162001
                                120001
                                 66000
                                 74000
                                132001
                                 50000
```

```
001546' 116001                          116001
001550' 104001                          104001
001552' 042000                           42000

                                ;CKMSG  - CHECK MSG GLOBAL SUBROUTINE
                                ;ONLY CHECKS MSG, NO HEADER AND NO CRC'S
                                ;PUT MODULE'S "BEGIN" AFTER JSR CALL
                                ;PUT LOGICAL LINE # NEXT, IF BIT 15 IS SET DATA IS CHECKED
                                ;BUT NO DATA ERR MSG'S ARE TYPED
                                ;PUT ADDR OF MSG TO BE CHECKED NEXT
                                ;DO JSR R5,CKMSG+2
                                ;RETURNS 2 BYTES PAST CALL IF ERROR
                                ;RETURNS 4 BYTES PAST CALL IF NO ERRORS
                                ;YOU MUST FIRST MAKE SURE LOCATION CKMSG IS ZERO
                                ;BEFOR JUMPING INTO THIS ROUTINE BECAUSE IT IS NOT RE-ENTREANT
                                ;IF IT IS NON ZERO YOU MUST DO BREAKS UNTIL ITS CLEARS
                                ;
                                ;
001554' 000000                  CKMSG:  0               ;FLAG LOCATION, = TO 0 IF ROUTINE IS NOT IN
                                                        ;USE, ELSE = -1
001556' 012767  177777  177770          MOV     #-1,CKMSG       ;SET FLAF TO BUSY
001564' 012567  002654                  MOV     (R5)+,CKBEG     ;GET MODULE POINTER
001570' 012567  002652                  MOV     (R5)+,CKLINE    ;GET LINE #
001574' 010046                          MOV     R0,-(R6)        ;SAVE   R0
001576' 010146                          MOV     R1,-(R6)        ;       R1
001600' 010246                          MOV     R2,-(R6)        ;       R2
001602' 005067  002642                  CLR     ERCNT           ;CLEAR ERRCOUNT
001606' 012700  003444'                 MOV     #MSG,R0         ;GET ADDR OF GOOD MSG
001612' 012501                          MOV     (R5)+,R1        ;GET ADDR OF MSG TO BE CK'ED
001614' 016702  002620                  MOV     MSGSIZ,R2       ;SET UP COUNTER
001620' 122021                  1$:     CMPB    (R0)+,(R1)+     ;COMPARE A BYTE
001622' 001016                          BNE     4$              ;BR IF NOT EQUAL
001624' 005302                  11$:    DEC     R2              ;ARE WE DONE?
001626' 001374                          BNE     1$              ;BR IF NOT
001630' 012602                  2$:     MOV     (R6)+,R2        ;       R2
001632' 012601                          MOV     (R6)+,R1        ;       R1
001634' 012600                          MOV     (R6)+,R0        ;       R0
001636' 005067  176166                  CLR     ERRCNT          ;CLEAR FOR NEXT CALL
001642' 005767  002602                  TST     ERCNT           ;ANY ERROR?
001646' 001001                          BNE     3$              ;BR IF SO
001650' 005725                          TST     (R5)+           ;BUMP RETURN ADDR
001652' 005067  177676          3$:     CLR     CKMSG
001656' 000205                          RTS     R5              ;RETURN
001660' 005267  002564          4$:     INC     ERCNT           ;UP ERR COUNT
001664' 005767  002556                  TST     CKLINE          ;TYPE ERRORS?
001670' 100757                          BMI     2$              ;BR IF NOT, EXIT
001672' 032767  002000  177570          BIT     #2000,177570            ;PRINT ALL ERRORS?
001700' 001004                          BNE     5$              ;BR IF YES
001702' 022767  000004  002540          CMP     #4,ERCNT        ;HAVE WE DONE 3 ALREADY?
001710' 003747                          BLE     2$              ;YES, GO EXIT
001712' 124041                  5$:     CMPB    -(R0),-(R1)     ;SET POINTERS BACK
001714' 016767  002524  176130          MOV     CKBEG,ACSR      ;GET MODULES NAME
001722' 016767  002520  176120          MOV     CKLINE,CSRA
001730' 016767  002504  176116          MOV     MSGSIZ,WASADR   ;TOTAL NUMBER OF CHARACTERS.
001736' 160267  176112                  SUB     R2,WASADR       ;MAKE IT WHICH CHARACTER IN MESSAGE.
001742' 005267  176106                  INC     WASADR          ;MAKE IT START WITH A ONE.
```



```
001746' 112067  176104                  MOVB    (R0)+,ASB
001752' 112167  176102                  MOVB    (R1)+,AWAS
001756' 010067  000116                  MOV     R0,R0SAV        ;SAVE R0
001762' 010167  000114                  MOV     R1,R1SAV        ;R1
001766' 010267  000112                  MOV     R2,R2SAV        ;R2
001772' 010367  000110                  MOV     R3,R3SAV
001776' 010467  000106                  MOV     R4,R4SAV
002002' 010567  000104                  MOV     R5,R5SAV
002006' 010667  000102                  MOV     R6,R6SAV
002012' 017767  002426  175760          MOV     @CKBEG,BEGIN    ;MAKE FIRST TWO LETTERS OF ERR CALL MATCH
                                                                ;CALLING MODULE'S
002020' 042767  020000  175772          BIC     #020000,STAT    ;WAKE THIS MODULE UP FOR CALL
                                        ;***************************************************************
002026' 104405  000000'                 DATER$,BEGIN            ;DATA ERROR!!!
                                        ;***************************************************************
002032' 052767  020000  175760          BIS     #020000,STAT    ;PUT IT BACK TO SLEEP
002040' 016706  000050                  MOV     R6SAV,R6
002044' 016705  000042                  MOV     R5SAV,R5
002050' 016704  000034                  MOV     R4SAV,R4
002054' 016703  000026                  MOV     R3SAV,R3
002060' 016702  000020                  MOV     R2SAV,R2        ;RESTORE R2
002064' 016701  000012                  MOV     R1SAV,R1        ;R1
002070' 016700  000004                  MOV     R0SAV,R0        ;R0

002074' 000167  177524                  JMP     11$             ;GO CHECK SOME MORE

002100' 000000                  R0SAV:  0               ;SAVE PLACE FOR REGISTERS
002102' 000000                  R1SAV:  0
002104' 000000                  R2SAV:  0
002106' 000000                  R3SAV:  0
002110' 000000                  R4SAV:  0
002112' 000000                  R5SAV:  0
002114' 000000                  R6SAV:  0

                                ;
                                ;;      THIS ROUTINE MAY BE CALLED BY NET MODULES DURING
                                ;       START-UP TO INITIALIZE THE HALF/FULL DUPLEX AND
                                ;       MULTI-DROP INFORMATION IN THE PHYTAB TABLE AND
                                ;       SET UP MULTI-DROP STATION ADDRESSES (IF REQUIRED).
                                ;
                                ;       CALL:   MOV     #BEGIN,R5
                                ;               JSR     R5,DEVSET
                                ;       NOTE: DEVSET SET MUST BE DEFINED AS A GLOBAL AND ALL
                                ;               REGISTERS ARE PRESERVED.
                                ;
002116' 005767  176042          DEVSET: TST     TFLAG                   ;IS THIS ROUTINE INHIBITED?
002122' 001401                          BEQ     1$                      ;BR IF NOT.
002124' 000205                          RTS     R5                      ;RETURN IMMEDIATELY.
002126' 010567  001230          1$:     MOV     R5,RET                  ;SAVE THE RETURN ADDRESS.
002132' 011605                          MOV     (SP),R5                 ;GET THE ARGUMENT.
002134' 010046                          MOV     R0,-(SP)                ;SAVE ALL REGISTERS.
002136' 010146                          MOV     R1,-(SP)
002140' 010246                          MOV     R2,-(SP)
002142' 010346                          MOV     R3,-(SP)
002144' 010446                          MOV     R4,-(SP)
002146' 005067  001214                  CLR     DFLAG                   ;FLAG TO CONTROL FLOW - SEE BELOW.
```

```
                                ;
                                ;       TYPE MODULE NAME.
                                ;
002170' 011567  001214          MOV     (R5),NAMEM              ;FIRST TWO CHARACTERS.
002174' 016567  000002  001210  MOV     2(R5),NAMEM+2           ;SECOND TWO CHARACTERS.
002202' 004567  176756          JSR     R5,TYPE                 ;TYPE IT.
002206' 003374'                 NAMEMG
                                ;
                                ;       REQUEST HALF/FULL DUPLEX INFORMATION. (IF DFLAG = 0)
                                ;       REQUEST MULTI-DROP INFORMATION. (IF DFLAG = 1)
                                ;       NOTE: IF '@' IS TYPED, THE SUBROUTINE IS IMMEDIATELY
                                ;       TERMINATED.
                                ;
002210' 004567  176750  DSET1:  JSR     R5,TYPE
002214' 005233'                 $ETM1
002216' 005767  001144          TST     DFLAG                   ;MULTI-DROP PASS?
002222' 001004                  BNE     MTYP                    ;BRANCH IF YES.
002224' 004567  176734          JSR     R5,TYPE                 ;TYPE "HALF DUPLEX"
002230' 005276'                 $ETM2
002232' 000403                  BR      CTYP                    ;CONTINUE.
002234' 004567  176724  MTYP:   JSR     R5,TYPE                 ;TYPE "MULTI-DROP"
002240' 005314'                 $ETM3
002242' 004567  176716  CTYP:   JSR     R5,TYPE
002246' 005331'                 $ETM4
002250' 016500  000014          MOV     14(R5),R0               ;GET DVID1.
002254' 016503  000016          MOV     16(R5),R3               ;GET SR1 IN CASE DOING MULTI-DROP.
002260' 005001                  CLR     R1                      ;INIT TO LINE ZERO.
002262' 112767  000060  003071  MOVB    #60,$ETM6               ;INIT TYPE OUT.
002270' 032700  000001  DSET2:  BIT     #1,R0                   ;THIS LINE TO BE TESTED?
002274' 001007                  BNE     DSET4                   ;BR IF YES.
002276' 006200          DSET3:  ASR     R0                      ;MOVE ONTO NEXT LINE.
002300' 006203                  ASR     R3
002302' 005201                  INC     R1                      ;LINE NO. = LINE NO. + 1.
002304' 022701  000020          CMP     #16.,R1                 ;DONE ALL SIXTEEN?
002310' 001522                  BEQ     DEXT                    ;BR IF YES.
002312' 000766                  BR      DSET2                   ;CHECK THIS LINE.
002314' 022701  000010  DSET4:  CMP     #10,R1                  ;AT LINE TEN(OCTAL)?
002320' 001003                  BNE     3$                      ;BR IF YES.
002322' 112767  000061  003031  MOVB    #61,$ETM6               ;NOW DOING LINES WITH TWO DIGIT NOS.
002330' 010102          3$:     MOV     R1,R2                   ;LINE NO.
002332' 042702  000010          BIC     #10,R2                  ;ISOLATE LOW ORDER DIGIT.
002336' 062702  000060          ADD     #60,R2                  ;MAKE IT ASCII.
002342' 110267  003014          MOVB    R2,$ETM6+1              ;PUT IT INTO THE BUFFER.
002346' 004567  176612          JSR     R5,TYPE                 ;ASK THE OPERATOR ABOUT THIS LINE.
002352' 005353'                 $ETM5
002354' 105777  000000G 4$:     TSTB    @TKS                    ;CHARACTER?
002360' 100375                  BPL     4$                      ;WAIT FOR IT.
002362' 117702  000000G         MOVB    @TKB,R2                 ;GET IT.
002366' 042702  177600          BIC     #177600,R2              ;ISOLATE IT.
002372' 110267  002040          MOVB    R2,OUTPUT               ;ECHO IT.
002376' 004567  176562          JSR     R5,TYPE
002402' 004436'                 OUTPUT
002404' 010267  000754          MOV     R2,FCHAR                ;SAVE IT.
```



```
002410' 022702  000015          CMP     #15,R2                  ;CARRIAGE RETURN?
002414' 001423                  BEQ     6$                      ;BR IF IT IS.
002416' 105777  000000G 5$:     TSTB    @TKS                    ;WAIT FOR CARRIAGE RETURN.
002422' 100375                  BPL     5$
002424' 117702  000000G         MOVB    @TKB,R2                 ;GET IT.
002430' 042702  177600          BIC     #177600,R2              ;ISOLATE IT.
002434' 110267  001776          MOVB    R2,OUTPUT               ;ECHO IT.
002440' 004567  176520          JSR     R5,TYPE
002444' 004436'                 OUTPUT
002446' 122702  000015          CMPB    #15,R2                  ;CARRIAGE RETURN?
002452' 001404                  BEQ     6$                      ;;BR IF YES.
002454' 122702  000177          CMPB    #177,R2                 ;RUBOUT?
002460' 001356                  BNE     5$                      ;BR IF NOT.
002462' 000734                  BR      4$                      ;RESTART.
002464' 112767  000012  001744  6$:     MOVB    #12,OUTPUT      ;TYPE A LINE FEED.
002472' 004567  176466          JSR     R5,TYPE
002476' 004436'                 OUTPUT
002500' 010502                  MOV     R5,R2                   ;TOP OF MODULE.
002502' 060102                  ADD     R1,R2                   ;OFFSET TO LINE ENTRY.
002504' 005767  000656          TST     DFLAG                   ;DOING MULTI-DROP?
002510' 001402                  BEQ     7$                      ;BR IF NOT.
002512' 000167  000102          JMP     MDROP                   ;HANDLE MULTI-DROP.
002516' 022767  000131  000640  7$:     CMP     #131,FCHAR      ;FIRST CHARACTER A 'Y'?
002524' 001004                  BNE     8$                      ;BR IF NOT.
002526' 152762  000100  000212          BISB    #100,212(R2)    ;FULL DUPLEX.
002534' 000660                  BR      DSET3                   ;CONTINUE CHECKING.
002536' 022767  000100  000620  8$:     CMP     #100,FCHAR      ;IS IT AN '@'?
002544' 001404                  BEQ     DEXT                    ;BR IF YES.
002546' 142762  000100  000212          BICB    #100,212(R2)    ;HALF DUPLEX.
002554' 000653                  BR      DSET3                   ;CONTINUE.
002556' 005767  000604  DEXT:   TST     DFLAG                   ;DONE MULTI-DROP?
002562' 001003                  BNE     1$                      ;BR IF YES.
002564' 005267  000576          INC     DFLAG                   ;SET FOR MULTI-DROP.
002570' 000607                  BR      DSET1                   ;DO IT.
002572' 016777  176474  000000G 1$:     MOV     TKSSAV,@TKS     ;RESTORE TTY STATUS.
002600' 016705  000556          MOV     RET,R5                  ;RETURN ADDRESS BACK IN R5.
002604' 012604                  MOV     (SP)+,R4                ;RESTORE REGISTERS.
002606' 012603                  MOV     (SP)+,R3
002610' 012602                  MOV     (SP)+,R2
002612' 012601                  MOV     (SP)+,R1
002614' 012600                  MOV     (SP)+,R0
002616' 000205                  RTS     R5
002620' 022767  000131  000536  MDROP:  CMP     #131,FCHAR      ;WAS ANSWER YES?
002626' 001410                  BEQ     1$                      ;BR IF NOT.
002630' 122767  000100  000526          CMPB    #100,FCHAR      ;IS IT A "@"?
002636' 001747                  BEQ     DEXT                    ;BR IF YES.
002640' 142762  000020  000212          BICB    #20,212(R2)     ;CLEAR THE FLAG.
002646' 000613                  BR      DSET3                   ;CONTINUE.
002650' 152762  000020  000212  1$:     BISB    #20,212(R2)     ;SET THE FLAG.
002656' 032703  000001          BIT     #1,R3                   ;IS THIS LINE MASTER?
002662' 001450                  BEQ     MDSLAV                  ;BR IF NOT.
002664' 005767  000503          TST     MFLAG                   ;ALREADY HAD A MASTER LINE?
002670' 001064                  BNE     SECMAS                  ;BR IF YES.
002672' 005267  000472          INC     MFLAG                   ;DON'T ALLOW ANOTHER MASTER.
002676' 010502                  MOV     R5,R2                   ;BASE ADDRESS OF MODULE.
002700' 010563  000466          MOV     R5,LIM
```

```
   637  002704' 062767  000331  000460          ADD     #331,LIM                ;END OF MDMTAB.
   638  002712' 062702  000312                  ADD     #312,R2                 ;BASE OF ADDRESS MENU TABLE.
   639  002716' 004567  176242                  JSR     R5,TYPE                 ;ASK OPERATOR FOR ADDRESSES.
   640  002722' 005376'                         SETM8
   641  002724' 004767  000132          2$:     JSR     PC,ROCT                 ;GET RESPONSE.
   642  002730' 022767  000055  000346          CMP     #55,CQUEUE-2            ;WAS IT MINUS?
   643  002736' 001415                          BEQ     3$                      ;BR IF YES.
   644  002740' 010167  000420                  MOV     R1,FCHAR                ;HOLD R1 TEMPORARILY.
   645  002744' 004767  000350                  JSR     PC,A$COCT               ;CONVERT STRING TO OCTAL NO.
   646  002750' 110112                          MOVB    R1,(R2)                 ;SAVE IT IN THE MENU TABLE.
   647  002752' 016701  000406                  MOV     FCHAR,R1                ;RESTORE R1.
   648  002756' 005202                          INC     R2                      ;POINT TO NEXT BYTE.
   649  002760' 020267  000406                  CMP     R2,LIM                  ;FULL TABLE?
   650  002764' 001357                          BNE     2$                      ;BR IF NOT.
   651  002766' 000167  177304                  JMP     DSET3                   ;GO ON TO NEXT LINE.
   652  002772' 112762  000377  000312  3$:     MOVB    #377,312(R2)            ;SHOW END OF TABLE ENTRIES.
   653  003000' 000167  177272                  JMP     DSET3                   ;CONTINUE.
   654  003004' 004567  176154          MDSLAV: JSR     R5,TYPE                 ;ASK FOR SLAVE STATION ADDRESS.
   655  003010' 005365'                         SETM7
   656  003012' 004767  000044                  JSR     PC,ROCT         ;READ IT.
   657  003016' 010167  000342                  MOV     R1,FCHAR                ;HOLD ONTO R1.
   658  003022' 004767  000272                  JSR     PC,ASCOCT               ;CONVERT RESPONSE TO OCTAL.
   659  003026' 110162  000272                  MOVB    R1,272(R2)              ;SAVE IT.
   660  003032' 016701  000326                  MOV     FCHAR,R1                ;RESTORE R1.
   661  003036' 000167  177234                  JMP     DSET3                   ;CONTINUE.
   662  003042' 004567  176116          SECMAS: JSR     R5,TYPE                 ;HAVE OPERATOR CORRECT PROBLEM
   663  003046' 005421'                         SMMSG                           ;OF TWO MULTI-DROP MASTERS.
   664  003050' 016777  176216  000000G         MOV     TKSSAV,@TKS             ;RESTORE TTY STATUS.
   665  003056' 000777                  1$:     BR      .
   666  003060' 000776                          BR      1$                      ;WAIT FOR ^C.
   667                                  ;
   668                                  ;       READ ASCII STRING OF OCTAL DIGITS TERMINATED BY
   669                                  ;       A CARRIAGE RETURN.  RUBOUTS AND ^U ARE RECOGNIZED.
   670                                  ;
   671  003062' 012704  003306'          ROCT:   MOV     #CQUEUE,R4              ;INIT THE QUEUE.
   672  003066' 010046                          MOV     R0,-(SP)                ;SAVE CALLER'S R0.
   673  003070' 105777  000000G         1$:     TSTB    @TKS                    ;CHARACTER READY?
   674  003074' 100375                          BPL     1$                      ;IF NOT, WAIT FOR IT.
   675  003076' 117700  000000G                 MOVB    @TKB,R0                 ;GET THE CHARACTER.
   676  003102' 042700  177600                  BIC     #177600,R0              ;STRIP IT.
   677  003106' 022700  000055                  CMP     #55,R0                  ;MINUS SIGN?
   678  003112' 001431                          BEQ     31$                     ;BR IF YES.
   679  003114' 022700  000015                  CMP     #15,R0                  ;IS IT A CR?
   680  003120' 001426                          BEQ     31$                     ;BR IF YES.
   681  003122' 022700  000177                  CMP     #177,R0                 ;RUBOUT?
   682  003126' 001004                          BNE     2$                      ;BR IF NOT.
   683  003130' 112467  001302                  MOVB    (R4)+,OUTPUT            ;GET LAST CHARACTER.
   684  003134' 105724                          TSTB    (R4)+                   ;FIX POINTER.
   685  003136' 000422                          BR      4$                      ;GO ECHO IT.
   686  003140' 022700  000025          2$:     CMP     #25,R0                  ;^U?
   687  003144' 001006                          BNE     3$                      ;BR IF NOT.
   688  003146' 004567  176012                  JSR     R5,TYPE                 ;FORCE A CR-LF.
   689  003152' 004472'                         CRLF
   690  003154' 012704  003306'                 MOV     #CQUEUE,R4              ;START WITH A FRESH QUEUE.
   691  003160' 000743                          BR      1$                      ;RESTART THIS ROUTINE.
   692  003162' 022700  000060          3$:     CMP     #60,R0                  ;MUST BE BETWEEN 60 AND 67.
```



```
   693  003166' 003024                          BGT     5$                      ;(OCTAL DIGIT).
   694  003170' 022700  000067                  CMP     #67,R0                  ;ERROR IF NOT.
   695  003174' 002421                          BLT     5$
   696  003176' 010044                  31$:    MOV     R0,-(R4)                ;CHARACTER TO QUEUE.
   697  003200' 110067  001232                  MOVB    R0,OUTPUT               ;ECHO IT.
   698  003204' 004567  175754          4$:     JSR     R5,TYPE
   699  003210' 004436'                         OUTPUT
   700  003212' 022700  000015                  CMP     #15,R0                  ;WAS IT A LINE FEED?
   701  003216' 001324                          BNE     1$                      ;BR IF NOT.
   702  003220' 112767  000012  001210          MOVB    #12,OUTPUT              ;SET UP TO TYPE...
   703  003226' 004567  175732                  JSR     R5,TYPE                 ;...A LINE FEED.
   704  003232' 004436'                         OUTPUT
   705  003234' 012600                          MOV     (SP)+,R0                ;RESTORE R0.
   706  003236' 000207                          RTS     PC                      ;RETURN TO CALLER.
   707  003240' 110067  001172          5$:     MOVB    R0,OUTPUT               ;CHARACTER IN ERROR.
   708  003244' 004567  175714                  JSR     R5,TYPE                 ;ECHO IT.
   709  003250' 004436'                         OUTPUT
   710  003252' 004567  175706                  JSR     R5,TYPE                 ;WARN OPERATOR.
   711  003256' 003310'                         TERRM
   712  003260' 012704  003306'                 MOV     #CQUEUE,R4              ;TRY AGAIN WITH A FRESH QUEUE.
   713  003264' 000701                          BR      1$
   714                                  ;
   715  003266' 000010                          .BLKW   10                      ;INPUT QUEUE.
   716  003306'                         CQUEUE:
   717  003306' 000000                          0
   718  003310' 005015  047516  006441  TERRM:  .ASCIZ  <15><12>'NO!'<15><12>
   719  003316' 000012
   720                                          .EVEN
   721                                  ;
   722                                  ;       CONVERT ASCII STRING TO OCTAL.
   723                                  ;       CHARACTERS IN CQUEUE, TERMINATED BY CARRIAGE RETURN.
   724                                  ;       NOTE: ONLY THE LAST FIVE DIGITS OF A STRING ARE VALID.
   725                                  ;       CALL:   JSR     PC,ASCOCT
   726                                  ;               OCTAL RETURNED IN R1
   727                                  ;               CONTENTS OF R1 AND R4 ARE LOST.
   728                                  ;
   729  003320' 005724                  ASCOCT: TST     (R4)+                   ;BYPASS THE CARRIAGE RETURN.
   730  003322' 005714                  ASCO1:  TST     (R4)                    ;AT TOP OF INPUT STACK?
   731  003324' 001414                          BEQ     AEXIT                   ;BR IF YES.
   732  003326' 012446                          MOV     (R4)+,-(SP)             ;CHARACTER (DIGIT) TO STACK.
   733                                                                          ;TWO STACKS = ONE QUEUE!
   734  003330' 162716  000060                  SUB     #60,(SP)                ;STRIP TO OCTAL.
   735  003334' 004767  177762                  JSR     PC,ASCO1                ;RECURSE.
   736  003340' 006301                          ASL     R1                      ;MOVE WORD OVER
   737  003342' 006301                          ASL     R1
   738  003344' 006301                          ASL     R1
   739  003346' 062601                          ADD     (SP)+,R1                ;ADD NEXT DIGIT.
   740  003350' 042701  100000                  BIC     #100000,R1              ;NO SIGN ALLOWED.
   741  003354' 000207                          RTS     PC                      ;RETURN.
   742  003356' 005001                  AEXIT:  CLR     R1                      ;ALL DIGITS READ.
   743  003360' 000207                          RTS     PC                      ;TAKE THE CHARACTERS FROM THE
   744                                                                          ;QUEUE AND RETURN WITH THE OCTAL NO.
   745                                  ;
   746  003362' 000000                  RET:    0                               ;RETURN ADDRESS HELD HERE.
   747  003364' 000000                  FCHAR:  0                               ;OPERATOR RESPONSE HELD HERE.
   748  003366' 000000                  DFLAG:  0
```

```
                                MFLAG:  0
                                LIM:    0
003372' 000000                  
003374' 005015  042516  020124  NAMEMG: .ASCII  <15><12>'NET MOD:  '
003402' 047515  035104  020040  
003410' 047116  047116  005015  NAMEM:  .ASCIZ  'NNNN'<15><12>          ;MODULE NAME STORED HERE.
003416'    000
                003420'             .EVEN
                                ;
                                ;CONSTANTS AND STORAGE LOCATIONS
                                ;
003420' 000000                  TIME:   .WORD   0               ;HOLDS POINTE TO SYSTEM TIME
                                ;THE FOLLOWING 10 BYTES WILL DEFAULT TO 10 SYNC CHAR'S, 026
                                ;THESE ARE USED BY SOME DECNET MODULE'S TRANSMIT ROUTINE
                                ;THESE LOCATIONS MAY BE OVERWRITTEN WITH A NEW SYNC CHAR IF
                                ;SOME USER SPECIFIES A NON STD SYNC.  IF HE DOES HE MUST CHANGE
                                ;THE SYNC FOR ALL DECNET MODULES THAT USE THIS SYNC HEADER.
                                ;THESE WOULD BE THE NQA, NHA, ??? MODULES
003422' 013026                          .WORD   013026
003424' 013026                          .WORD   013026
003426' 013026                          .WORD   013026
003430' 013026                          .WORD   013026
003432' 013026                          .WORD   013026
003434'    220                  MSGHED: .BYTE   220             ;DLE, MEANS OFF-LINE MSG
003435'    000                  COUNT:  .BYTE   0               ;COUNT AND FLAGS
003436'    000                          .BYTE   0               ;
003437'    000                          .BYTE   0               ;FILL
003440'    000                          .BYTE   0               ;FILL
003441'    001                          .BYTE   1               ;ADDRESS
003442' 000000                  CRCHED: .WORD   0               ;HEADER CRC
003444'    024                  MSG:    .BYTE   24              ;OSOP PROTOCOL CODE FOR "LOOP BACK"
003445'    015  042412  051122  IMSG:   .ASCII  <15><12>'ERROR CODES FOR NET MODULES (ASTAT):'<15><12>
003452' 051117  041440  042117  
003460' 051505  043040  051117  
003466' 047040  052105  046440  
003474' 042117  046125  051505  
003502' 024040  051501  040524  
003510' 024524  006472     012  
003515'    060  030460  054060          .ASCII  '0010XX          REC TIME OUT'<15><12>
003522' 020130  020040  004440  
003530' 042522  020103  044524  
003536' 042515  047440  052125  
003544' 005015                  
003546' 030060  030062  054130          .ASCII  '0020XX          TMT CLOCK LOSS'<15><12>
003554' 020040  020040  052011  
003562' 052115  041440  047514  
003570' 045503  046040  051517  
003576' 006523     012          
003601'    060  031460  054060          .ASCII  '0030XX          HEADER CRC ERROR'<15><12>
003606' 020130  020040  004440  
003614' 042510  042101  051105  
003622' 041440  041522  042440  
003630' 051122  051117  005015  
003636' 030060  030064  054130          .ASCII  '0040XX          MSG CRC ERROR'<15><12>
003644' 020040  020040  046411  
003652' 043523  041440  041522  
003660' 042440  051122  051117  
```



```
003666' 005015                  
003670' 030060  030065  054130          .ASCII  '0050XX          TMT TIME OUT'<15><12>
003676' 020040  020040  052011  
003704' 052115  052040  046511  
003712' 020105  052517  006524  
003720'    012                  
003721'    060  033060  054060          .ASCII  '0060XX          TMT NON-EXIST. MEM'<15><12>
003726' 020130  020040  004440  
003734' 046524  020124  047516  
003742' 026516  054105  051511  
003750' 027124  046440  046505  
003756' 005015                  
003760' 030060  030067  054130          .ASCII  '0070XX          TMT LATENCY'<15><12>
003766' 020040  020040  052011  
003774' 052115  046040  052101  
004002' 047105  054503  005015  
004010' 030460  030060  054130          .ASCII  '0100XX          REC PARITY ERROR'<15><12>
004016' 020040  020040  051011  
004024' 041505  050040  051101  
004032' 052111  020131  051105  
004040' 047522  006522     012  
004045'    060  030461  054060          .ASCII  '0110XX          REC NON-EXIST. MEM'<15><12>
004052' 020130  020040  004440  
004060' 042522  020103  047516  
004066' 026516  054105  051511  
004074' 027124  046440  046505  
004102' 005015                  
004104' 030460  030062  054130          .ASCII  '0120XX          REC CLOCK LOSS'<15><12>
004112' 020040  020040  051011  
004120' 041505  041440  047514  
004126' 045503  046040  051517  
004134' 006523     012          
004137'    060  030062  054060          .ASCII  '0200XX          REC FRAMING ERROR'<15><12>
004144' 020130  020040  004440  
004152' 042522  020103  051106  
004160' 046501  047111  020107  
004166' 051105  047522  006522  
004174'    012                  
004175'    060  030064  054060          .ASCII  '0400XX          REC OVERRUN OR LATENCY'<15><12>
004202' 020130  020040  004440  
004210' 042522  020103  053117  
004216' 051105  052522  020116  
004224' 051117  046040  052101  
004232' 047105  054503  005015  
004240'    000                  
004241'    116  020117  054523  NOCLKM: .ASCIZ  'NO SYSTEM CLOCK, PLEASE RECONFIGURE, PGM WILL HALT'
004246' 052123  046505  041440  
004254' 047514  045503  020054  
004262' 046120  040505  042523  
004270' 051040  041505  047117  
004276' 044506  052507  042522  
004304' 020054  043520  020115  
004312' 044527  046114  044040  
004320' 046101  000124          
                004434'             .=MSG+504.
004434'    000                  CRCMSG: .BYTE   0               ;MSG CRC
```

```
                        .BYTE   0
                        .EVEN
OUTPUT: .WORD   0               ;ONE WORD CHAR BUFFER FOR TTY OUT
MSGSIZ: 501.            ;GLOBAL, SIZE OF DATA MSG (NO HEADER)
WAIT:   .WORD   0               ;NETWORK MODULES WAIT FOR THIS TO BE 0
CKBEG:  .WORD   0               ;TMP STORAGE FOR POINTER TO CALLING MODULE
CKLINE: .WORD   0               ;TMP STORAGE FOR LOGICAL LINE #
ERCNT:  .WORD   0               ;COUNTS # OF ERRORS
;PWRCNT:: .WORD 0 DEFINED IN MONITOR QABL ON AND IS A GLOBAL - POWER FAIL INDICATOR FOR
;
;       OPTION DISPATCH TABLE.
;
ODISP:  FINISH                  ;ERROR RECOVERY.
        DONE                    ;STANDARD MESSAGE.
        USER                    ;USER DEFINED MESSAGE.
        ZEROS                   ;ALL ZERO BYTES MESSAGE.
        ONES                    ;ALL ONE BYTES MESSAGE.
        ALTER                   ;ALTERNATE ONE AND ZERO BITS.
        WORST                   ;WORST CASE MESSAGE.
;
IMFLAG: 0                       ;ONE TIME TYPE OUT FLAG.
;
;
CRLF:   .ASCIZ  <15><12>        ;CARRIAGE RETURN - LINE FEED.
GETMG:  .ASCII  <15><12>'SELECT NETWORK TEST MESSAGE OPTION.'<15><12>

        .ASCIZ  '(0-6 OR "H" FOR HELP): '

HELPM:  .ASCII  <15><12>'  TYPE   FOR THIS OPTION'<15><12>

        .ASCII  '    0      USE PREVIOUS MESSAGE OPTION'<15><12>

        .ASCII  '    1      STANDARD MESSAGE'<15><12>

        .ASCII  '    2      USER MESSAGE'<15><12>
```



```
        .ASCII  '    3      ALL ZEROS MESSAGE'<15><12>

        .ASCII  '    4      ALL ONES MESSAGE'<15><12>

        .ASCII  '    5      ALTERNATING 1 AND 0 BITS'<15><12>

        .ASCII  '    6      FOUR BIT COUNT'<15><12>

USERM:  .ASCIZ  'TYPE YOUR MESSAGE TERMINATED WITH "@"'<15><12>'$'

SETM1:  .ASCIZ  'SPECIFY IF THE FOLLOWING LINES ARE'

SETM2:  .ASCIZ  ' FULL DUPLEX '

SETM3:  .ASCIZ  ' MULTI-DROP '

SETM4:  .ASCIZ  '(Y) OR NOT(CR).'<15><12>

SETM5:  .ASCII  'LINE #'

SETM6:  .ASCIZ  'NN '
SETM7:  .ASCIZ  'ADDRESS='

SETM8:  .ASCIZ  'SLAVE ADDRESSES:'<15><12>

SMMSG:  .ASCII  <15><12>'TWO MULTIDROP MASTER LINES NOT ALLOWED.'<15><12>
```

```
  973  005426' 046440  046125  044524
  974  005434' 051104  050117  046440
  975  005442' 051501  042524  020122
  976  005450' 044514  042516  020123
  977  005456' 047516  020124  046101
  978  005464' 047514  042527  027104
  979  005472' 005015
  980  005474' 054524  042520  057040          .ASCIZ  'TYPE "C AND FIX SR1.'
  981  005502' 020103  047101  020104
  982  005510' 044506  020130  051123
  983  005516' 027061    000
  984          000001                  .END
```



```
ACSR     000052R           107#    464*
ADDR     000006R            86#
AEXIT    003356R           731     742#
ALTEP    000532R           207#    878
ASB      000056R           110#    469*
ASCOCT   003320R           645     658     729#
ASCO1    003322R           730#    735
ASTAT    000054R           109#
AWAS     000060R           111#    470*
BDCNV =  ****** G            1#
BEGIN    000000R            83#    291     478*    482
BIT0  =  000001            120#
BIT1  =  000002            120#
BIT10 =  002000            120#
BIT11 =  004000            120#
BIT12 =  010000            120#
BIT13 =  020000            120#
BIT14 =  040000            120#
BIT15 =  100000            120#
BIT2  =  000004            120#
BIT3  =  000010            120#
BIT4  =  000020            120#
BIT5  =  000040            120#
BIT6  =  000100            120#
BIT7  =  000200            120#
BIT8  =  000400            120#
BIT9  =  001000            120#
BREAK$=  104407            120#
BR1      000012R            88#
BR2      000013R            89#
CDATA$=  104414            120#
CKBEG    004444R           434*    464     478     866#
CKLINE   004446R           435*    457     465     867#
CKMSG    001554RG          122#    161*    431#    433*    454*
CLOCK =  ****** G          120#    130
COUNT    003435R           275*    276*    277*    772#
COUEUE   003306R           642     671     690     712     716#
CRC      001276RG          122#    200     285     341#
CRCHED   003442R           282*    777#
CRCMSG   004434RG          122#    210     220     287*    860#
CRCTAB   001454R           357     362     384#
CRLF     004472R           269     689     884#
CSRA     000050R           105#    465*
CTYP     002242R           547     550#
DATCK$=  104417            120#
DATEP$=  104405            120#    482
DEVSET   002116RG          122#    514#
DEXT     002556R           562     600     611#    626
DFLAG    003366R           524*    543     600     611     613*    748#
DONE     001010R           211     221     272#    874
DSET1    002210R           541#    614
DSET2    002270R           556#    563
DSET3    002276R           558#    606     610     620     651     653     661
DSET4    002314R           557     564#
DVID1    000014R            90#
ENDPS$=  104402            120#    291
```

```
END$   = 104403          120#
ERCNT  #04450R          439*   451     456*    461     868#
ERRCNT #00030R           97#   450*
ERRN$  = 104410          120#
ERROR$= 104404           120#
EXIT$  = 104400          120#
FCHAR  #03364R          580*   603     607     623     625     644*    647     657*    660     747#
FILCNT= ****** G         120#   308
FILL   #00536R          203     206     208#
FILLER= ****** G         120#   309
FINISH #01016R          268     273#    873
GETMG  #04475R          164     885#
GETMSG #00340R          167#
GETPA$= 104413           120#
GOPT   #00254R          134     138     142     153#    198
GWBUF$= 104412           120#
HELP   #00502R          188     190     195#
HELPM  #04574R          195     896#
IMFLAG #04470R          153     159*    881#
IMSG   #03445R          158     779#
INIT   #00022R           94#
KW11L  = ****** G        120#   133
KW11P  = ****** G        120#   137
KW11X  = ****** G        120#   141
LIM    #03372R          636*   637*    649     750#
MAP22$= 104415           120#
MDROP  #02620R          602     623#
MDSLAV #03004R          631     654#
MFLAG  #03370R          525*   632     634*    749#
MODNAM #00000R           84#
MODSP  #00164R           95     118#
MSG    #03444RG         121#   167     208     219     238     283     289*    290*    440     778#    859
MSGMED #03434RG         121#   278     771#
MSGN$  = 104411          120#
MSGSIZ #04440RG         121#   267*    272*    275     276     284     288     442     466     864#
MSGS$  = 104416          120#
MSG$   = 104406          120#
MTYP   #02234R          544     548#
NAMEM  #03410R          531*   532*    753#
NAMEMG #03374R          534     751#
NOCLKM #04241R          143     850#
OACNV  = ****** G         1#
ODISP  #04452R          194     873#
ONES   #00524R          204#    877
OPEN   = #00000           85      91      98      99     100     101     102     103     104     105     107     109     110
                         111     120#    145     166     178     186     197     237     260     271     281     286
OUTPUT #04436R          175*   176     183*    184     257*    258     577*    579     587*    589     595*    597     683*
                        697*   699     702*    704     707*    709     863#
PASCNT #00026R           96#
PIRQ$  = #00004          120#
POPSP  = #05726          120#
POPSP2= #22626           120#
PRTY   = #00000          88      89      120#
PRTY0  = #00000          120#
PRTY1  = #00040          120#
PRTY2  = #00100          120#
```



```
PRTY3  = #00140          120#
PRTY4  = #00200          120#
PRTY5  = #00240          120#
PRTY6  = #00300          120#
PRTY7  = #00340          120#
PS     = 177776          120#
PSW    = 177776          120#
PUSH   = #05746          120#
PUSH2  = #24646          120#
PWRCNT= ****** G         122#
QUE$   = 104401          120#
RESTRT #00166R          112     130#
RET    #03362R          517*   616     746#
RETT   #03362R          641     656     671#
RSTRT  #00062R          112#
R0SAV  #02100R          341*   380     471*    491     495#
R1SAV  #02102R          342*   379     472*    490     496#
R2SAV  #02104R          343*   378     473*    489     497#
R3SAV  #02106R          344*   377     474*    488     498#
R4SAV  #02110R          162*   163*    345*    376     475*    487     499#
R5SAV  #02112R          476*   486     500#
R6SAV  #02114R          477*   485     501#
SBADR  #00052R          106#
SECMAS #03042R          633     662#
SFTM1  #05233R          542     947#
SETM2  #05276R          546     953#
SETM3  #05314R          549     956#
SETM4  #05331R          551     959#
SETM5  #05353R          572     963#
SETM6  #05361R          555*   566*    570*    965#
SETM7  #05365R          655     966#
SETM8  #05376R          640     968#
SMMSG  #05421R          663     972#
SPOINT #00024R           95#
SPSIZ  = #00040           1#    113
SR     = ****** G        122#   155
SR1    #00016R           91#
START  #00166R           94     129#
STAT   #00020R           93#    480*    484*
SVR0   #00032R           98#
SVR1   #00034R           99#
SVR2   #00036R          100#
SVR3   #00040R          101#
SVR4   #00042R          102#
SVR5   #00044R          103#
SVR6   #00046R          104#
TERRM  #03310R          711     718#
TFLAG  #00164R          127#    514
TIME   #03420RG         120#   133*    137*    141*    160*    759#
TKB    = ****** G        121#   173     243     575     585     675
TKS    = ****** G        121#   168     169*    171     239*    241     274*    526     527*    573     583     615*    664*
                        673
TKSSAV #01272R          168*   274     323#    526*    615     664
TPB    = ****** G        121#   309*    314*
TPS    = ****** G        121#   302     303*    310     315     320*
TPSSAV #01274R          302*   320     324#
```

```
TPX    = 000000            120#
TRAPX  = 000020            120#
TYPE     001164RG          121#    144     157     165     177     185     196     236     259     270     299#    533     541
                           545     548     550     571     578     588     596     639     654     662     680     698     703
                           706     710
USER     000622R           235#    875
USERM    005162R           235     940#
VECTOR   000010R            87#
WAIT     004442RG          121#    273*    865#
WASADR   000054R           108#    466*    467*    468*
WORST    000556R           217#    879
XFLAG    000005R            85#
ZEROS    000520R           202#    876
.      = 005521R           172     311     316     665     715#    755#    859#

. ABS.   000000      000
         005521      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 N1AB,N1AB/SOL/CRF:SYM=DDXCOM,N1AB
 RUN-TIME:  1  2  .2 SECONDS
 RUN-TIME RATIO: 20/4=4.9
 CORE USED:  9K  (15 PAGES)
```

```
                        .REM %

                                IDENTIFICATION
                                --------------

                        PRODUCT CODE:   MAINDEC-11-DXNUA-A-D

                        PRODUCT NAME:   DU-11 DECNET DEC/X11 EXERCISER MODULE

                        DATE:   21-JUN-76

                        MAINTAINER:     DEC/X11 DIAGNOSTIC GROUP

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

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

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


COPYRIGHT  (C) 1976 DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
```

1. ABSTRACT

THIS TYPE OF DEC/X11 MODULE, REFERRED TO AS DECNET DEC/X11 MODULE, WILL DETECT SYSTEM INTERACTION ERRORS WHEN RUN IN CONJUNCTION WITH OTHER DEC/X11 MODULES EXERCISING OTHER DEVICES ON THE SYSTEM. ERRORS ARE REPORTED ON THE SYSTEM CONSOLE AS THEY OCCUR OR AS A SUMMARY REPORT, AT THE OPERATORS OPTION. IN ADDITION, THESE MODULES MAY BE USED TO VERIFY THE CONDITION OF THE ASSOCIATED MODEMS AND COMMUNICATION LINKS. IT WILL ONLY TEST COMMUNICATIONS PATHS CAPABLE OF RUNNING DDCMP. THIS MEANS THE DEVICES MUST USE NO PARITY, 8 LEVEL CODE, AND TERMINALS CANNOT BE TESTED.

NOTE

THIS DOCUMENT ASSUMES FAMILIARITY WITH THE PHILOSOPHY AND OPERATION OF DEC/X11. THE DETAILS OF LINKING, LOADING, AND RUNNING MODULES UNDER DEC/X11 CAN BE FOUND IN THE DEC/X11 USERS DOCUMENTATION AND REFERENCE GUIDE, MAINDEC-11-DXQBA. THE READER SHOULD ALSO HAVE READ THE DECNET DEC/X11 USER'S GUIDE, MAINDEC-11-DXQBC, WITHOUT THROUGHLY UNDERSTANDING THAT DOCUMENT, THERE IS LITTLE HOPE OF SUCCESSFUL TESTING.

2. HARDWARE REQUIREMENTS

THESE MODULES REQUIRE A COMPLETE PATH FROM A COMMUNICATIONS DEVICE TO A COMMUNICATIONS DEVICE. THIS MAY BE ACCOMPLISHED THROUGH A VARIETY OF METHODS:

A. 1 PROCESSOR, 1 COMM. DEVICE, AND A LOOP BACK DEVICE WITH MODEM SIMULATION CAPABILITIES. (MUST RUN FULL DUPLEX)

B. 1 PROCESSOR, 2 COMM. DEVICES, 2 MODEMS

C. 2 PROCESSORS, 2 COMM. DEVICES, 2 MODEMS

UP TO SIXTEEN LINES CAN BE HANDLED BY ONE MODULE BUT ONLY 2 LINES ARE KEPT RUNNING SIMULTANEOUSLY.

3. SOFTWARE REQUIREMENTS

A. THE DEC/X11 MONITOR MUST BE QABK OR A LATER REVISION.

B. THE N1A? MODULE MUST BE CONFIGURED DIRECTLY AFTER THE MONITOR*

C. ALL NETWORK MODULES MUST BE CONFIGURED NEXT.

D. THE N2A? MODULE MUST BE CONFIGURED NEXT.*

E. ONE OF THE 3 CLOCK MODULES MUST BE CONFIGURED NEXT. EITHER KWA?, KWB?, OR KWX?

F. ANY OTHER DESIRED DEC/X11 MODULES CAN THEN BE CONFIGURED IN ANY ORDER.

4. PRELIMINARY TESTING

THE APPROPRIATE STAND ALONE DIAGNOSTICS MUST BE RUN TO VERIFY THE CPU, AND THE COMMUNICATIONS DEVICE. THEN A DEC/X11 EXERCISER USING THE CONVENTIONAL (MAINTENANCE MODE) DEC/X11 MODULES FOR THE COMMUNICATIONS DEVICES MUST BE RUN. IF BOTH THESE STEPS PASS SUCCESSFULLY, DECNET DEC/X11 MAY BE ATTEMPTED.

5. PROGRAMMING CONSIDERATIONS

A. OTHER THAN THE REQUIREMENTS MENTIONED IN 3 REGARDING THE ORDER IN WHICH MODULES MUST BE REQUIRED, THESE DECNET DEC/X11 MODULES WILL BE COMPLETELY COMPATIBLE WITH THE DEC/X11 MONITORS AND ALL OTHER MODULES. THE ONLY OTHER EXECPTION IS THAT FOR A GIVEN COMMUNICATIONS DEVICE, THERE WILL NOW BE 2 DIFFERENT DEC/X11 MODULES CAPABLE OF BEING RUN, THE OLDER MAINTANCE MODE TYPE REQUIRING NO CONNECTIONS TO THE COMMUNICATION DEVICE, AND THE NEW DECNET TYPE REQUIRING A COMPLETED COMMUNICATIONS PATH. IT WILL BE POSSIBLE (AND SOMETIMES DESIREABLE) TO CONFIGURE 2 DIFFERENT DEC/X11 MODULES IN ONE RUN TIME EXERCISER WHICH TEST THE SAME PHYSICAL DEVICE. IF THIS IS DONE, ONLY ONE OF THE 2 MODULES MAY BE SELECTED FOR THE SAME RUN.

B. USING THESE DECNET DEC/X11 MODULES, IT IS POSSIBLE TO HAVE MANY NODES CONNECTED IN A NETWORK, ALL RUNNING DEC/X11 TOGETHER. ALL THESE EXERCISERS WOULD HAVE BEEN STARTED AT DIFFERENT TIMES. DUE TO DIFFERENT HARDWARE ON EACH SYSTEM, RUN TIMES AND END OF PASS TIMES MIGHT VARY CONSIDERABLY. THEREFORE IT IS NECCESSARY THAT THESE DECNET DEC/X11 MODULES BE EXTREMELY PATIENT AND HAVE A HIGH RETRY CAPABILITY IN ALL OPERATIONS. THIS HOWEVER, MEANS IT SOMETIMES MAY TAKE A LONG TIME (ON THE ORDER OF 10 MINUTES) BEFORE AN OPERATOR FINDS OUT A LINE IS DEFINITELY NOT RUNNING.

C. DUE TO THE EXCESSIVE PROGRAMMING COMPLICATIONS AND LARGE

---

* THESE ARE DESCRIBED IN THE DECNET DEC/X11 USER'S GUIDE.

BUFFERS REQUIRED TO RUN MANY LINES FROM ONE MODULE ALL AT THE SAME TIME, IT WAS DECIDED TO ONLY RUN 2 AT ANY ONE TIME. IF THE MODULE IS SET TO RUN 16 LINES, IT WILL RUN 1 PAIR FOR THE NUMBER OF TRANSFERS INDICATED BY LOCATION LPCNT, THEN IT WILL RUN A 2ND PAIR, THEN 3RD, ETC UP TO THE 8TH PAIR. IT WILL THEN REPORT AN END OF PASS AND GO BACK TO THE FIRST PAIR, THEN 2ND, ETC. THIS FORCES SOME CONSIDERATIONS TO BE MADE WHEN CONNECTING LINES TOGETHER SO THAT A LINE IN THE ACTIVE MODE IS NOT WAITING ON A LINE IN THE DORMANT MODE. PLEASE READ THE DECNET DEC/X11 USER'S GUIDE, ESPECIALLY THE SECTION ON 2 LINES AT A TIME CONSIDERATIONS.

D. THESE MODULES WILL, AS A GENERAL RULE, USE THE MINIMUM AMOUNT OF FEATURES OF A COMMUNICATION DEVICE REQUIRED TO SEND AND RECEIVE MESSAGES. THIS MEANS THAT CRC CALCULATING HARDWARE, CHARACTER RECOGNITION HARDWARE, ETC, WILL NOT BE USED. THESE FEATURES ARE FULLY TESTED BY STAND ALONE DIAGNOSTICS.

## 6. OPERATING PROCEDURES

A. THE MODULE IS CONFIGURED, LOADED, AND RUN AS SPECIFIED IN THE DEC/X11 REFERENCE GUIDE.

B. CONSOLE SWITCHES ARE STANDARD FOR DEC/X11

C. THE N1A? MODULE WILL SOLICIT FROM THE OPERATOR WHICH TYPE OF MESSAGE HE WANTS TO USE FOR DECNET ACTIVITY. THE OPTIONS INCLUDE:

- ALL ONES
- ALL ZEROS
- ALTERNATING ONES AND ZEROS
- A DIGIT (4 BIT) REPEATING COUNT PATTERN
- A PRECANNED ASCII MESSAGE
- USER SPECIFIED BY TYPING IN

D. THE N2A? MODULE WILL MAKE ANY AUTOMATIC PHONE CONNECTIONS USING ANY DN-11'S PRESENT AND WILL REQUEST THE OPERATOR TO DIAL ALL MANUAL CONNECTIONS. ALL OPERATORS AT EACH NODE UNDER TEST MUST REACH THIS POINT BEFORE ANY ONE NODE CONTINUES INTO THE EXERCISING PHASE.

E. THE N2A? MODULE WILL REQUEST A CARRIAGE RETURN WHEN THE OPERATOR WANTS TO BEGIN EXERCISING HIS NODE.

F. THERE ARE MANY OPTIONS WHICH MAY BE ALTERED BEFORE RUN TIME, THEY INCLUDE:

(1) LOCATION 164 IS A SWITCH WHICH INDICATES WHETHER OR NOT ALL ERRORS SHOULD BE REPORTED AS THEY HAPPEN. IF LOCATION 164 IS NON-ZERO (EQUAL TO N), THEN ERRORS ARE TOTALLED (ON A LINE BY LINE BASIS); EVERY N PASSES THIS

TOTAL NUMBER OF ERRORS FOR THE PREVIOUS N PASSES WILL BE REPORTED. THE VALUE OF N IS THE VALUE OF LOCATION 164. THE MAXIMUM NUMBER OF ERRORS THAT CAN BE TALLEYED IS 256 PER LINE. THEREFORE IF A LINE IS NOT KNOWN TO BE OF VERY GOOD QUALITY, THE VALUE OF LOCATION 164 TIMES THE VALUE OF LOCATION 166 SHOULD BE LESS THAN 128. (LOCATION 166 IS DEFINED NEXT.) THIS MEANS THE NUMBER OF MESSAGES PER PASS TIMES THE NUMBER OF PASSES PER SUMMARY REPORT SHOULD BE LESS THAN 128. IN THIS WAY, IF EVERY MESSAGE ATTEMPTED HAS AN ERROR ON BOTH TRANSMIT AND RECEIVE, THE ERROR TALLY WILL NOT OVERFLOW. IF THE LINE IS OF KNOWN GOOD QUALITY, THIS RESTRICTION MAY BE RELAXED. THE DEFAULT VALUE IS TO NOT REPORT TOTAL ERRORS.(LOC 164=0) OF COURSE DEC/X11 KEEPS TOTALS FOR THE ENTIRE RUN, REGARDLESS.

(2) LOCATION 166 INDICATES HOW MANY ITERATIONS SHOULD BE DONE BEFORE AN END OF PASS IS CALLED. AN ITERATION IS NORMALLY ONE TRANSMIT AND ONE RECEIVE, IN EITHER ORDER. THE DEFAULT VALUE IS 8.

(3) LOCATION 170 IS A TIME SCALING FACTOR. THERE ARE MANY TIME OUT LOOPS IN THE MODULES, SOME SHOULD LOGICALLY BE LONGER THAN OTHERS. THE VALUE IN LOCATION 170 IS THE NUMBER OF SECONDS TO WAIT FOR THE SMALLEST TIME OUTS. SOME TIME OUTS WILL BE A MULTIPLE OF THIS FACTOR. DEFAULT IS 30 SECONDS.

(4) LOCATION 172 IS AN ERROR TOLERATION LEVEL, ON A SINGLE MESSAGE BASIS. SO IF IN AN ATTEMPT TO DO ONE ITERATION (USUALLY A TRANSMIT THEN RECIEVE OR VICE-VERSA) THE NUMBER OF ERRORS REACHES THE VALUE IN LOCATION 172, THAT LINE IS DROPPED FROM TESTING. IF BEFORE THIS LEVEL IS REACHED, AN ITERATION IS CORRECTLY COMPLETED, THEN THE ERROR COUNT WHICH IS COMPARED TO LOCATION 172 IS RESET TO ZERO. THE DEFAULT IS 10.

(5) LOCATION 174 CONTAINS 2 IDENTICAL BYTES FOR USE AS SYNC CHARACTERS. THE DEFAULT IS 113226 WHICH IS 2 BYTES OF 226. IF THIS IS CHANGED, BOTH BYTES MUST BE IDENTICAL TO EACH OTHER. THIS LOCATION IS A DON'T CARE FOR ASYNCHRONOUS DEVICES, BECAUSE THEY "SYNC" ON THE FIRST BYTE OF THE ACTUAL MESSAGE, NOT ON PRECEDING CHARACTERS AS SYNCHRONOUS DEVICES DO.

(6) LOCATION 176 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN RECEIVE-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, SLAVE. (BIT 7 = 1, BITS 4,5, AND 6 = 0'S IN THE PHYSICAL LINE TABLE. SEE ITEM 6-F-9 BELOW). THEN PATCH THE LOCATION POINTED TO BY LOCATION 176 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, SLAVE BY THIS MODULE TO RUN IN RECEIVE ONLY MODE. THIS LOCATION MAY BE PATCHED BACK TO A 401 TO RESUME NORMAL REC-TMT TESTING.

(7) LOCATION 200 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN TRANSMIT-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, MASTER (BITS 7 AND 5 = 1'S, BITS 4 AND 6 = 0'S IN THE PHYSICAL-LINE TABLE. SEE ITEM 6-F-9 BELOW.) THEN PATCH THE LOCATION POINTED TO BY LOCATION 200 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, MASTER BY THIS MODULE TO RUN IN TRANSMIT ONLY MODE. THE LOCATION MAY BE PATCHED BACK TO 401 TO RESUME NORMAL TMT-REC TESTING.

G. LOCATIONS DVA, VCT, BR1 AND BR2 MUST BE SET TO PROPERLY MATCH THE FIRST DEVICE TO BE TESTED BY THIS MODULE.

## 7. ERROR REPORTING

A. ERRORS ARE EITHER TOTALLED AND PRINTED AS SUMMARIES ON A LINE BY LINE BASIS, OR THEY ARE PRINTED AS THEY OCCUR. SEE SECTION 6F1. THEY ARE 2 CLASSES OF ERRORS DATA ERRORS AND OTHER ERRORS.

B. DATA ERRORS ARE ACTUALLY REPORTED BY THE N1A? MODULE BECAUSE THIS IS WHERE THE SHARED DATA COMPARE ERROR ROUTINE IS LOCATED. SO ALL DECNET DEC/X11 MODULE DATA COMPARE ERRORS LOOK LIKE THE STANDARD DEC/X11 DATA COMPARE ERROR MESSAGE EXCEPT:

(1) THE MODULE NAME TYPED IS FFXX, WHERE FF IS FILLED IN TO BE THE NAME OF THE DECNET DEC/X11 MODULE WHICH CALLED THE N1A? MODULE TO DO THE COMPARE. IN OTHER WORDS FF REPRESENTS THE MODULE THAT "HAD" THE ERROR.

(2) THE ERROR NUMBER IS A COUNT OF THE DATA COMPARE ERRORS IN THE CURRENT MESSAGE BEING CHECKED, IT IS THEREFORE RESET TO ZERO EVERY TIME A NEW MESSAGE IS TO BE CHECKED.

(3) ACSR CONTAINS THE ADDRESS OF THE DECNET DEC/X11 MODULE WHICH CALLED THIS ROUTINE TO HAVE ITS MESSAGE CHECKED.

(4) SBADR CONTAINS THE LINE # OF THE DEVICE WHICH HAD THE DATA ERROR.

(5) WASADR CONTAINS THE COUNT (IN OCTAL) OF WHICH CHARACTER IN THE MESSAGE THIS BAD CHARACTER WAS.

(6) THE "SHOULD BE AND "WAS" ITEMS ARE NORMAL.

C. OTHER ERRORS ARE REPORTED USING THE NORMAL ERROR CALL IN

DEC/X11 CSRA AND ACSR ARE STANDARD. STATC CONTAINS A CODED

ERROR NUMBER, WITH THE CODE BEING CONSISTENT IN ALL DECNET DEC/X11 MODULES. THE LOW ORDER 2 DIGITS OF STATC CONTAIN WHICH LINE HAD THE ERROR THE OTHER DIGITS CONTAIN THIS CODE:

| | |
|---|---|
| 0010XX | REC TIME OUT |
| 0020XX | TMT CLOCK LOSS |
| 0030XX | HEADER CRC ERROR |
| 0040XX | MSG CRC ERROR |
| 0050XX | TMT TIME OUT |
| 0060XX | TMT NON-EXISTENT MEMORY |
| 0070XX | TMT LATENCY |
| 0100XX | REC DATA PARITY ERROR |
| 0110XX | REC NON-EXISTENT MEMORY |
| 0120XX | REC CLOCK LOSS |
| 0200XX | REC FRAMING ERROR |
| 0400XX | REC OVERRUN OR LATENCY ERROR |

THIS METHOD SHOULD ELIMINATE THE CONSTANT NEED FOR HAVING A LISTING ON HAND TO DECIDE WHAT THE ERRORS WHERE, THE FIRST TIME THE RUN TIME EXERCISER IS RUN AFTER LOADING, THE N1A? MODULE WILL TYPE THIS ERROR CODE LIST OUT TO THE SYSTEM CONSOLE FOR THE OPERATORS SUBSEQUENT USE.

D. ALL ERRORS HAVE A RETRY LIMIT (SEE SECTION 6F4) WHICH CAN BE MODIFIED.

E. ON POWER-FAIL, POWER-UP CONDITIONS, DECNET DEC/X11 MODULES WILL DROP THEMSELVES. THIS IS DONE BECAUSE THE PHONE CONNECTIONS WILL HAVE BEEN LOST AND OPERATOR INTERVENTION WILL BE REQUIRED.

8. TEST SEQUENCE

LINES ARE TESTED BY PAIRS IN SEQUENTIAL ORDER. THIS ORDER MAY BE MODIFIED BY MODIFYING THE PHYSICAL LINE TABLE. (SEE DECNET DEC/X11 USER'R GUIDE). THERE ARE SEVERAL TESTING OPTIONS WITH SLIGHTLY DIFFERENT FUNCTIONS MAKING UP AN ITERATION:

A. POINT-TO-POINT SLAVE. IN THIS CASE THE LINE WAITS TO RECEIVE A MESSAGE, WHEN IT DOES, IT CHECKS IT, AND THEN TRANSMITS A MESSAGE BACK. IT WILL REPORT ERRORS ON EXCESSIVE WAITS.

B. POINT-TO-POINT MASTER. IN THIS CASE THE LINE SENDS A MESSAGE AND WAITS TO RECEIVE ONE IN RETURN. IT THEN CHECKS THE RECEIVED MESSAGE. IF IT TIMES-OUT, IT TRANSMITS AGAIN, IT WILL CONTINUE TO TIME OUT AND RE-TRANSMIT UNTIL EITHER A MESSAGE IS RECEIVED, OR IT TIMES OUT COUNT EQUALS LOCATION

172, THE ERROR TOLERATION LEVEL. (SEE SECTION 6F4). IT WILL THEN REPORT AN ERROR AND BEGIN AGAIN. AFTER THE NUMBER OF

CONSECUTIVE TIME-OUT ERRORS REPORTED EQUALS LOCATION 172, THE LINE WILL BE DROPPED.

C. POINT-TO-POINT SLAVE RECEIVE ONLY. IN THIS MODE THE LINE WAITS ONLY TO RECEIVE, CHECKS THE MESSAGE, AND WAITS TO RECEIVE AGAIN.

D. POINT-TO-POINT MASTER-TRANSMIT ONLY. IN THIS MODE THE LINE JUST TRANSMITS AND THEN TRANSMITS AGAIN ,ETC.

E. MULTI-DROP MASTER. IN THIS MODE, THE LINE WILL SEND A DDCMP MESSAGE OUT FOR THE ADDRESS CONTAINED IN THE MULTI-DROP MENU TABLE, WAIT TO RECEIVE A MESSAGE BACK, AND THEN CHECK THE DATA.
* NOT YET IMPLEMENTED

F. MULTI-DROP SLAVE. IN THIS MODE, THE LINE WILL WAIT FOR A MESSAGE WITH THE MATCHING ADDRESS. THEN IT WILL TRANSMIT A MESSAGE.
* NOT YET IMPLEMENTED

G. THESE BASIC ITERATIONS ARE REPEATED UP TO THE NUMBER OF TIMES IN LOCATION 166. (SEE SECTION 6F2) FOR EACH LINE SELECTED FOR TEST, AND THE END-OF-PASS IS DECLARED.

9. PASS TIMES

PASS TIMES ARE EXTREMELY CONFIGURATION DEPENDENT. DIFFERENT BAUD RATES, MESSAGE SIZES, AND TESTING MODES ALL GREATLY AFFECT THE PASS TIME. BELOW ARE SOME BALL PARK TIMES, BUT ANY ONE SYSTEM COULD VARY TREMENDOUSLY FROM THESE:
-RUNNING ALONE, 1 LINE ON AN 11/05 AT 9600 BAUD,HALF DUPLEX, POINT-TO-POINT, WITH A 512 CHARACTER MESSAGE, TAKES 15 SECONDS.

NOTE 1 AT VARIOUS MONITOR CALLS THROUGHOUT THIS MODULE, IT IS LIKELY THAT WHEN 2 LINES ARE RUNNING, A MONITOR CALL CAN BE MADE AND BEFORE CONTORL IS RETURNED TO THE MODULE, THE OTHER LINE MAY CAUSE THE EXACT SAME CALL TO BE MADE. IN FACT, WHEN RUNNING 2 LINES FULL DUPLEX, IT MAY BE POSSIBLE TO HAVE 4 CALLS TO THE MONITOR FROM ONE SPOT IN THE MODULE, BEFORE THE FIRST CALL IS RETURNED FROM. THIS IS WHY AT SUCH CALLS, THE CODE SETS COUNTERS BEFORE DOING CALLS, SO THAT WHEN IT IS RETURNED TO, IT CAN KEEP WHICH

```
                        LINE DID WHAT STRAIGHT.
                        
                        
                        NOTE 2  WHEN RUNNING 2 LINES LOOPED TO EACH OTHER OUT OF THE SAME
                        MODULE, THE LOW ORDER ONE (PER ORDER OF DVC, LOC, 14) SHOULD BE THE SLAVE,
                        AND THE HIGHER ORDER ONE SHOULD BE MASTER.
                        %
```

```
                                ;MODEM RATES MUST BE THE SAME!!!!!
000000'                                 IOMOD <NUAA >, 1,4,5,5
000000'                                 MODULE  140000,NUAA ,1,4,5,5,
                                        .TITLE  NUAA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  4    9/6/75
                                        .LIST   BIN
                                ;*****************************************************************
000000'                         BEGIN:
000000' 052516  040501      040 MODNAM: .ASCII  /NUAA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000001                  ADDR:   1+0             ;1ST DEVICE ADDR.
000010' 000004                  VECTOR: 4+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' 000332'                 INIT:   START                   ;MODULE START ADDR.
000024' 000164'                 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.
000062' 000532'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000164'                         MODSP:
                                ;*****************************************************************
                                        .GLOBL  WAIT, TIME, MSGSIZ, CRMSG, CRCMSG,MSGHED, MSG, PWRCNT
                                        .GLOBL  DEVSET
                                ;
                                ;
000164' 000000                  REPORT: 0               ;FLAG, IF =0, THEN ERRORS ON RECEIVED MSG'S
                                                        ;ARE TOTALLED AND PRINTED EVERY # PASSES
                                                        ;IF =0, ERRORS ARE ALL REPORTED AS THEY HAPPEN.
000166' 000010                  LPCNT:  10              ;ITERATIONS PER END PASS CALL
000170' 000036                  TIMELN: 30.             ;TIME OUT FACTOR, IN SECONDS
000172' 000012                  ERRLVL: 10.             ;NUMBER OF ERRORS TOLERATED
000174' 113226                  SYNC:   113226          ;2 BYTES OF SYNC CHARACTER
000176' 001150'                         ONEREC          ;ADDRESS OF WHERE 240 GOES
```



```
000200' 001472'                         ONETMT          ;ADDRESS OF WHERE 240 GOES

000202' 000000                  SPARE1: 0
000204' 000000                  SPARE2: 0
000206' 000000                  SPARE3: 0
000210' 000000                  SPARE4: 0
                                ;
                                ;
                                ;PHYSICAL LINE TABLE. BYTE 0 CONTAINS THE PHYSICAL
                                ;LINE # FOR THE FIRST LINE TO BE RUN, BYTE 1 WILL BE
                                ;THE PHYSICAL LINE RUN SECOND, ETC.
                                ;EACH BYTE IN THE TABLE HAS THE FOLLOWING BIT MEANINGS:
                                ;BIT 0-3        PHYSICAL LINE #
                                ;BIT 4          0 FOR POINT-TO POINT, 1 FOR MULTI-DROP
                                ;BIT 5          0 FOR SLAVE, 1 FOR MASTER (DERIVED FROM SR1)
                                ;BIT 6          0 FOR HALF DUPLEX, 1 FOR FULL
                                ;BIT 7          0 DO NOT TEST, 1 TEST (DERIVED FROM DVID1)
000212'     000                 PHYTAB: .BYTE   0
000213'     001                         .BYTE   1
000214'     002                         .BYTE   2
000215'     003                         .BYTE   3
000216'     004                         .BYTE   4
000217'     005                         .BYTE   5
000220'     006                         .BYTE   6
000221'     007                         .BYTE   7
000222'     010                         .BYTE   10
000223'     011                         .BYTE   11
000224'     012                         .BYTE   12
000225'     013                         .BYTE   13
000226'     014                         .BYTE   14
000227'     015                         .BYTE   15
000230'     016                         .BYTE   16
000231'     017                         .BYTE   17

                                ;
                                ;
                                ;ERRTAB ,KEEPS TRACK OF RETRY ERROR COUNTS FOR
                                ;EACH LINE
                                ;
000232'     000                 ERRTAB: .BYTE   0
000233'     000                         .BYTE   0
000234'     000                         .BYTE   0
000235'     000                         .BYTE   0
000236'     000                         .BYTE   0
000237'     000                         .BYTE   0
000240'     000                         .BYTE   0
000241'     000                         .BYTE   0
000242'     000                         .BYTE   0
000243'     000                         .BYTE   0
000244'     000                         .BYTE   0
000245'     000                         .BYTE   0
000246'     000                         .BYTE   0
000247'     000                         .BYTE   0
000250'     000                         .BYTE   0
000251'     000                         .BYTE   0
```

```
                                ;
                                ;ECOUNT TABLE KEEPS TRACK OF # OF BAD MSG'S
000252'  000                    ECOUNT: .BYTE   0
000253'  000                            .BYTE   0
000254'  000                            .BYTE   0
000255'  000                            .BYTE   0
000256'  000                            .BYTE   0
000257'  000                            .BYTE   0
000260'  000                            .BYTE   0
000261'  000                            .BYTE   0
000262'  000                            .BYTE   0
000263'  000                            .BYTE   0
000264'  000                            .BYTE   0
000265'  000                            .BYTE   0
000266'  000                            .BYTE   0
000267'  000                            .BYTE   0
000270'  000                            .BYTE   0
000271'  000                            .BYTE   0
                                ;
                                ;
                                ;MULTI-DROP STATION ADDRESS TABLE
                                ;BYTE 0 IS THE MULTI-DROP STATION ADDRESS
                                ;FOR LOGICAL LINE 0, BYTE 1 IS FOR LOGICAL LINE 1, ETC.
                                ;THIS ENTRY ONLY HAS MEANING IF CORRESPONDING
                                ;BYTE IN PHYSICAL LINE TABLE HAS BITS 4 AND 7 SET
                                ;AND BIT 5 OFF, IE MULTI-DROP SLAVE TO BE TESTED
000272'  000                    MDATAB: .BYTE   0
000273'  001                            .BYTE   1
000274'  002                            .BYTE   2
000275'  003                            .BYTE   3
000276'  004                            .BYTE   4
000277'  005                            .BYTE   5
000300'  006                            .BYTE   6
000301'  007                            .BYTE   7
000302'  010                            .BYTE   10
000303'  011                            .BYTE   11
000304'  012                            .BYTE   12
000305'  013                            .BYTE   13
000306'  014                            .BYTE   14
000307'  015                            .BYTE   15
000310'  016                           .BYTE   16
000311'  017                           .BYTE   17
                                ;
                                ;MULTI-DROP STATION ADDRESS MENU TABLE
                                ;THIS TABLE INDICATES TO ANY LINE RUNNING AS
                                ;MULTI-DROP MASTER, WHICH STATION ADDRESS TO
                                ;ATTEMPT TO EXERCISE ON THE MULTI-DROP LINE
                                ;IT ALLOWS UP TO 15 DIFFERENT STATIONS TO BE
                                ;TESTED ON A LINE.  EACH BYTE WHICH IS NOT=-1
                                ;INDICATES A STATION ADDRESS TO BE TESTED. EACH
                                ;STATION IS TESTED, BY THE ORDER OF THE TABLE,
                                ;UNTIL THE FIRST OCCURANCE OF A -1 (BINARY 11111111)
                                ;
000312'  377                    MDMTAB: .BYTE   -1
000313'  377                            .BYTE   -1
000314'  377                            .BYTE   -1
```



```
000315'  377                            .BYTE   -1
000316'  377                            .BYTE   -1
000317'  377                            .BYTE   -1
000320'  377                            .BYTE   -1
000321'  377                            .BYTE   -1
000322'  377                            .BYTE   -1
000323'  377                            .BYTE   -1
000324'  377                            .BYTE   -1
000325'  377                            .BYTE   -1
000326'  377                            .BYTE   -1
000327'  377                            .BYTE   -1
000330'  377                            .BYTE   -1
000331'  377                            .BYTE   -1
                                ;
                                ;
                                ;
000332' 016700  177456          START:  MOV     DVID1,R0        ;GET DEVICES TO BE TESTED
000336' 001002                          BNE     1$              ;ANY ACTIVE?
000340' 104403  000000'                 END$,BEGIN              ;
000344' 012705  000000'         1$:     MOV     #BEGIN,R5       ;PREPARE FOR NXT JSR
000350' 004567  000000G                 JSR     R5,DEVSET       ;GO ASK FOR PARAMETERS
000354' 012767  000200  004246          MOV     #200,MASK       ;SET UP MASK
000362' 012701  000232'         2$:     MOV     #PHYTAB+16.,R1  ;GET END OF DEVICE TABLE
000366' 006300                  3$:     ASL     R0              ;CHECK EACH BIT
000370' 103003                          BCC     4$              ;BR IF NOT ON
000372' 156741  004232                  BISB    MASK,-(R1)      ;SET THIS LINE ACTIVE
000376' 000402                          BR      5$
000400' 146741  004224          4$:     BICB    MASK,-(R1)      ;CLEAR THIS LINE
000404' 105061  000040          5$:     CLRB    32.(R1)         ;CLEAR ECOUNT TABLE
000410' 020127  000211'                 CMP     R1,#PHYTAB-1    ;DONE?
000414' 001364                          BNE     3$              ;BR IF NOT
000416' 026727  004206  000040          CMP     MASK,#40        ;BEEN HERE BEFORE?
000424' 001406                          BEQ     6$              ;BR IF YES
000426' 012767  000040  004174          MOV     #40,MASK        ;SET UP MASK
000434' 016700  177356                  MOV     SR1,R0          ;GET SLAVE/MASTER INFO
000440' 000750                          BR      2$              ;GO SET UP
000442' 005201                  6$:     INC     R1              ;GET PHYTAB ADDR
000444' 016700  177336                  MOV     ADDR,R0         ;GET DEVICE ADDR
000450' 105721                  7$:     TSTB    (R1)+           ;DEVICE ACTIVE?
000452' 100010                          BPL     8$              ;BR IF NOT
000454' 012710  000002                  MOV     #2,(R0) ;SET DATA TERM READY
000460' 112767  000074  177507          MOVB    #74,SYNC+1
000466' 016760  177502  000002          MOV     SYNC,2(R0)      ;WRITE PARAMETER CONTROL REGISTER
                                                        ;PARITY OFF
                                                        ;8 BITS PER CHAR
                                                        ;SYNCHRONOUS INTERNAL FILE
000474' 062700  000010          8$:     ADD     #10,R0          ;BUMP R0
000500' 020127  000232'                 CMP     R1,#PHYTAB+16.  ;DONE?
000504' 001361                          BNE     7$              ;BR IF NOT
000506' 005067  004114          9$:     CLR     MSGCNT          ;COUNTS WHEN TO DO SUMMARY
000512' 005767  000000G                 TST     WAIT            ;LINES SET UP?
000516' 001405                          BEQ     RESTRT          ;BR IF YES
000520' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000524' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000530' 000766                          BR      9$              ;WAIT TILL WAIT IS CLEARED BY NET2
                                ;
```

```
                                        ;
                                        ;THROUGH-OUT THE PROGRAM MANY LOCATIONS ARE REFERENCED
                                        ;OFF OF (R4).  THIS IS A MEANS OF KEEPING TWO SETS OF
                                        ;VARIABLES, ONE FOR EACH OF THE ACTIVE LINES.
                                        ;R4 ALWAYS CONTAINS A 0 IF THE PROGRAM IS CURRENTLY
                                        ;REFERENCING LINE 0 OR A 2 IF IT IS REFERENCING
                                        ;LINE 1.
                                        ;
000532' 005767  000000G         RESTRT: TST     PWRCNT          ;HAS THERE BEEN A POWERFAIL
000536' 001402                          BEQ     1$              ;CONTINUE IF NOT,
000540' 104403  000000'                 END$,BEGIN              ;
000544' 005004                  1$:     CLR     R4              ;FIRST ACTIVE LINE
000546' 016706  177252                  MOV     SPOINT,R6       ;INIT STACK
000552' 005067  004224                  CLR     DROPR4          ;CLEAR ALL EXIT FLAGS
000556' 005067  004222                  CLR     DROPR4+2
000562' 005067  004220                  CLR     DTERR4
000566' 005067  004216                  CLR     DTERR4+2
000572' 005067  004214                  CLR     CRCR4
000576' 005067  004212                  CLR     CRCR4+2
000602' 005067  004210                  CLR     RCTOR4
000606' 005067  004206                  CLR     RCTOR4+2
000612' 005067  004204                  CLR     TXTOR4
000616' 005067  004202                  CLR     TXTOR4+2
000622' 005067  004200                  CLR     FDBKR4
000626' 005067  004176                  CLR     FDBKR4+2
000632' 005067  004174                  CLR     CHKR4
000636' 005067  004172                  CLR     CHKR4+2

000642' 005067  004132                  CLR     PASTIM          ;CLR PASS TIME FLAG
000646' 005067  004124                  CLR     BRKFLG          ;CLR FLAG THAT SAYS WE'RE IN BREAK
000652' 012767  000212' 003752          MOV     #PHYTAB,NXTLIN  ;POINTS TO NEXT LINE TO TEST
000660' 005067  004004                  CLR     LINACT          ;SHOW NO LINES ACTIVE
000664' 026727  003742  000232' GETNXT: CMP     NXTLIN,#PHYTAB+16.;DONE ALL LINES?
000672' 001440                          BEQ     DONE            ;BR IF SO
000674' 016700  003732                  MOV     NXTLIN,R0       ;GET ADDR OF NEXT LINE
000700' 010064  004676'                 MOV     R0,CURLIN(R4)   ;SAVE IT
000704' 105060  000020                  CLRB    20(R0)          ;CLEAR RETRY COUNT FOR THIS LINE
000710' 005267  003716                  INC     NXTLIN          ;BUMP FOR NEXT TIME
000714' 111000                          MOVB    (R0),R0         ;GET PHYSICAL LINE BYTE
000716' 002362                          BGE     GETNXT          ;BR BACK IF BIT 7 NOT ON
000720' 016701  177062                  MOV     ADDR,R1         ;SET UP TO CALC ADDR
000724' 004767  004106                  JSR     PC,GETADR       ;GO CALC ADDR
000730' 010164  004634'                 MOV     R1,DVASAV(R4)   ;SAVE DVA FOR THIS LINE
000734' 016701  177050                  MOV     VECTOR,R1       ;SET UP TO CALC VECTOR ADDR
000740' 004767  004072                  JSR     PC,GETADR       ;GO CALC
000744' 010164  004640'                 MOV     R1,VCTSAV(R4)   ;SAVE IT
000750' 005267  003714                  INC     LINACT          ;SHOW A LINE'S RUNNING
000754' 005064  004712'                 CLR     LOOP(R4)        ;CLEAR ITERATION COUNT
000760' 005064  004772'                 CLR     RCVTO(R4)       ;CLR REC TIME OUT COUNT
000764' 032700  000020                  BIT     #20,R0          ;MULTI DROP OR NOT?
A 000770' 001377                        BNE     MD              ;BR IF IT IS
000772' 000427                          BR      PP              ;BR IF NOT TO P TO P
                                        ;
                                        ;
000774' 005767  003670          DONE:   TST     LINACT          ;ANY LINES RUNNING?
001000' 001402                          BEQ     1$              ;BR IF NO
```



```
001002' 000167  003324                  JMP     TIMCHK          ;IF YES GO CHECK FOR TIME OUT
001006' 012701  000212'         1$:     MOV     #PHYTAB,R1      ;MUST CHECK IF MAYBE ALL LINES DROPPED
001012' 105721                  2$:     TSTB    (R1)+           ;IS THIS LINE STILL SELECTED?
001014' 100405                          BMI     3$              ;BR OUT IF YES
001016' 022701  000232'                 CMP     #PHYTAB+16.,R1  ;CHECKED ALL 16?
001022' 001373                          BNE     2$              ;BR BACK IF NO
001024' 104403  000000'                 END$,BEGIN              ;
                                        ;
001030' 012767  000001  003742  3$:     MOV     #1,PASTIM       ;SHOW IT IS TIME FOR ENDPAS
                                                                ;FOR BREAK LOOP
001036' 005767  003734                  TST     BRKFLG          ;ARE WE IN A BREAK LOOP?
001042' 001002                          BNE     4$              ;BR IF WE ARE
001044' 000167  003376                  JMP     EPASS           ;GO DO END OF PASS
                                        ;
001050'                         4$:
001050' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERUPT.
                                ;POINT TO POINT
001052' 012764  000001  004766' PP:     MOV     #1,FDFLAG(R4)   ;SAY WERE FULL DUPLEX
001060' 032700  000100                  BIT     #100,R0         ;HALF OR FULL DUPLEX?
001064' 001402                          BEQ     1$              ;BR IF HALF
001066' 000167  000512                  JMP     PPFD            ;JMP TO FULL DUPLEX ROUTINE
001072' 005064  004766'         1$:     CLR     FDFLAG(R4)      ;CORRECT FLAG CAUSE WERE NOT FD
001076' 032700  000040                  BIT     #40,R0          ;SLAVE OR MASTER?
001102' 001141                          BNE     PPHDM           ;BR IF MASTER
                                        ;
                                        ;POINT TO POINT, HALF DUPLEX, SLAVE
001104'                         PPHDS:
001104' 016464  004644' 004650'         MOV     AMSGBF(R4),RECADR(R4)  ;SET UP INPUT AREA
001112' 016764  000000G 004654'         MOV     MSGSIZ,RECCNT(R4)  ;SET UP COUNT
001120' 062764  000012  004654'         ADD     #12,RECCNT(R4)  ;CORRECT FOR HEADER AND CRC
001126' 016764  177036  004660'         MOV     TIMELN,RECTIM(R4)  ;HOW LONG TO WAIT
001134' 066764  177030  004660'         ADD     TIMELN,RECTIM(R4)       ;DOUBLE IT
001142' 004767  000766                  JSR     PC,REC          ;GO RECEIVE DATA
001146' 000462                          BR      FERR            ;BR TO ERROR ROUTINE
001150' 000401                  ONEREC: BR      1$              ;MAKE THIS A NOP=240 TO
001152' 000432                          BR      PPHDS2          ;DO ONE-WAY-IN (REC. ONLY)
001154' 012764  000000G 004722' 1$:     MOV     #MSGHED,TXADR1(R4)  ;GET MESS ADR
001162' 016764  000000G 004732'         MOV     MSGSIZ,TXCNT1(R4)  ;GET MSG SIZE
001170' 062764  000012  004732'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC'S
001176' 012764  000004  004736'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
001204' 012764  011152' 004726'         MOV     #FILL,TXADR2(R4)
001212' 004767  002722                  JSR     PC,CTS          ;GO GET CLEAR TO SEND
001216' 016764  176746  004742'         MOV     TIMELN,TMTTIM(R4)       ;GIVE 'EM 1/2 MINUTE
001224' 004767  002176                  JSR     PC,TMT          ;GO TRANSMIT
001230' 000423                          BR      DERR            ;BR TO ERROR ROUTINE
001232' 042774  000004  004634' PPHDS2: BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
001240' 005264  004712'                 INC     LOOP(R4)        ;COUNT AN ITERATION
001244' 026764  176716  004712'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH?
001252' 003314                          BGT     PPHDS           ;NO, GO DO ANOTHER TIME
001254' 005367  003410          LNDONE: DEC     LINACT
001260' 016403  004634'                 MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
001264' 042713  177775                  BIC     #177775,(R3)    ;KILL ALL BUT DTR
001270' 005063  000004                  CLR     4(R3)           ;KILL TRANSMITTER TOO
001274' 000167  177364                  JMP     GETNXT          ;YES DO NEXT LINE
                                        ;
                                        ;
```

```
001300' 042774  000004  004634' DERR:   BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
001306' 012703  177777                  MOV     #-1,R3          ;SET FLAG FOR WHICH RETURN
001312' 000401                          BR      FERR1
001314' 005003                  FERR:   CLR     R3              ;SET FLAG FOR WHICH RETURN
001316' 016401  004676'         FERR1:  MOV     CURLIN(R4),R1   ;GET LINE # ADDR
001322' 105261  000020                  INCB    20(R1)          ;COUNT AN ERROR
001326' 126167  000020  176636          CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
001334' 002003                          BGE     DROP            ;BR IF YES
001336' 005703                          TST     R3              ;DO A TMT OR DO A REC?
001340' 001703                          BEQ     ONEREC          ;BR IF TMT
001342' 000736                          BR      PPHDS2          ;GO TRY AGAIN
001344' 142711  000200          DROP:   BICB    #200,(R1)       ;DROP THE LINE FROM TEST
001350' 005264  005002'                 INC     DROPR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
001354' 104411  011156' 000000'         MSGN$,ADROP,BEGIN       ;ASCII MESSAGE CALL WITH COMMON HEADER
001362' 005767  003414                  TST     DROPR4
001366' 001402                          BEQ     1$
001370' 005004                          CLR     R4
001372' 000402                          BR      2$
001374' 012704  000002          1$:     MOV     #2,R4
001400' 005364  005002'         2$:     DEC     DROPR4(R4)
001404' 000723                          BR      LNDONE

                                ;
                                ;POINT TO POINT HALF DUPLEX MASTER
                                ;
001406'                         PPHDM:
001406' 012764  000000G 004722'         MOV     #MSGHED,TXADR1(R4)      ;GET MSG ADDRESS
001414' 016764  000000G 004732'         MOV     MSGSIZ,TXCNT1(R4)       ;GET MSG SIZE
001422' 062764  000012  004732'         ADD     #12,TXCNT1(R4)          ;CORRECT FOR CRC AND HEADER
001430' 012764  000004  004736'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
001436' 012764  011152' 004726'         MOV     #FILL,TXADR2(R4)
001444' 004767  002470                  JSR     PC,CTS                  ;GO GET CLEAR TO SEND
001450' 016764  176514  004742'         MOV     TIMELN,TMTTIM(R4)               ;GIVE 'EM 1/2 MINUTE
001456' 004767  001744                  JSR     PC,TMT                  ;GO TRANSMIT
001462' 000434                          BR      GERR                    ;ERROR RETURN
001464' 042774  000004  004634'         BIC     #4,@DVASAV(R4)          ;KILL REQ TO SEND
001472' 000401                  ONETMT: BR      1$                      ;MAKE THIS A NOP=240 FOR ONE
                                                                        ;WAY OUT, TMT ONLY, OTHER SIDE
                                                                        ;OF LINE MUST DO ONE WAY IN.
001474' 000417                          BR      PPHDM2                  ;USED FOR ONE WAY OUT ONLY
001476' 016464  004644' 004650' 1$:     MOV     AMSGBF(R4),RECADR(R4)   ;SET UP INPUT AREA
001504' 016764  000000G 004654'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
001512' 062764  000012  004654'         ADD     #12,RECCNT(R4)          ;CORRECT FOR HEADER AND CRC
001520' 016764  176444  004660'         MOV     TIMELN,RECTIM(R4)       ;GIVE EM 1/2 MINUTES
001526' 004767  000402                  JSR     PC,REC                  ;GO RECEIVE
001532' 000413                          BR      HERR                    ;ERROR RETURN
001534' 005264  004712'         PPHDM2: INC     LOOP(R4)                ;COUNT AN ITERATION
001540' 026764  176422  004712'         CMP     LPCNT,LOOP(R4)          ;DONE ENOUGH?
001546' 003317                          BGT     PPHDM                   ;BR IF NOT
001550' 000167  177500                  JMP     LNDONE
                                ;
                                ;
001554' 042774  000004  004634' GERR:   BIC     #4,@DVASAV(R4)          ;KILL REQ TO SEND
001562' 016401  004676'         HERR:   MOV     CURLIN(R4),R1           ;GET LINE # ADDRESS
001566' 105261  000020                  INCB    20(R1)                  ;COUNT AN ERROR
001572' 126167  000020  176372          CMPB    20(R1),ERRLVL           ;TOO MANY ERRORS?
```



```
001600' 002261                          BGE     DROP                    ;BR IF YES
001602' 000754                          BR      PPHDM2                  ;GO TRY AGAIN
                                ;
                                ;
                                ;POINT -TO-POINT FULL DUPLEX
                                ;
001604' 032700  000040          PPFD:   BIT     #40,R0          ;MASTER OR SLAVE?
001610' 001003                          BNE     1$              ;BR IF MASTER
001612' 012764  177777  004766'         MOV     #177777,FDFLAG(R4)      ;SHOW ITS SLAVE
001620' 004767  002314          1$:     JSR     PC,CTS          ;GO GET CLEAR TO SEND
001624' 016464  004644' 004650' PPFD1:  MOV     AMSGBF(R4),RECADR(R4)   ;GET INPUT AREA
001632' 016764  000000G 004654'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
001640' 062764  000012  004654'         ADD     #12,RECCNT(R4)  ;CORRECT FOR CRC AND HEADER
001646' 004767  000262                  JSR     PC,REC          ;GO GET RECEIVING STARTED
001652' 000500                          BR      JERR            ;REC ERROR RETURN
001654' 000441                          BR      FDRCDN          ;REC DONE
001656' 012764  000000G 004722'         MOV     #MSGHED,TXADR1(R4)      ;FULL DUPLEX RETURN FROM REC
                                                                ;GET TMT ADDR
001664' 016764  000000G 004732'         MOV     MSGSIZ,TXCNT1(R4)       ;GET TMT COUNT
001672' 062764  000012  004732'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC
001700' 012764  000004  004736'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
001706' 012764  011152' 004726'         MOV     #FILL,TXADR2(R4)
001714' 004767  001506                  JSR     PC,TMT          ;GO TMT
001720' 000455                          BR      JERR            ;TMT ERROR RETURN
001722' 000412                          BR      1$              ;GOOD TMT DONE RETURN
001724' 012700  002122'                 MOV     #FDTO,R0        ;FULL DUP RETURN, SET UP FOR TIME OUT
001730' 016701  176234                  MOV     TIMELN,R1               ;GIVE EM 1 MINUTE
001734' 066701  176230                  ADD     TIMELN,R1
001740' 012702  000000                  MOV     #0,R2           ;SHOW ITS A REC WAIT
001744' 000167  002306                  JMP     TIMKEP          ;GO WAIT
                                ;
001750' 012764  000001  004752' 1$:     MOV     #1,TMTDON(R4)   ;SHOW TMT IS DONE
001756' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                ;
                                ;
001760' 005764  004752'         FDRCDN: TST     TMTDON(R4)      ;IS TMT DONE YET?
001764' 001020                          BNE     4$              ;BR IF YES
001766' 005264  005026'                 INC     FDBKR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
001772' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
001776' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002002' 005767  003020                  TST     FDBKR4
002006' 001402                          BEQ     2$
002010' 005004                          CLR     R4
002012' 000402                          BR      3$
002014' 012704  000002          2$:     MOV     #2,R4
002020' 005364  005026'         3$:     DEC     FDBKR4(R4)
002024' 000755                          BR      FDRCDN          ;BR AND WAIT SOME MORE
002026' 005264  004712'         4$:     INC     LOOP(R4)        ;COUNT AN ITERATION
002032' 026764  176130  004712'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH
002040' 003271                          BGT     PPFD1           ;NO, GO DO MORE
002042' 042774  000004  004634'         BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
002050' 000167  177200                  JMP     LNDONE
                                ;
                                ;
                                ;FD TMT ERROR
002054' 012764  000001  004752' JERR:   MOV     #1,TMTDON(R4)   ;DON'T BOTHER WAITING FOR TMT
```

```
 886  002062' 012764  177777  004672'          MOV     #177777,ABORT(R4)       ;TELL INT ROUTINES TO GIVE UP
 887  002070' 016401  004676'                  MOV     CURLIN(R4),R1   ;GET CURRENT LINE INFO
 888  002074' 105261  000020                   INCB    20(R1)          ;COUNT ANOTHER ERROR
 889  002100' 126167  000020  176064           CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
 890  002106' 002724                           BLT     FDRCD%          ;NO GO TRY AGAIN
 891  002110' 042774  000004  004634'          BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
 892  002116' 000167  177222                   JMP     DROP            ;GIVE UP ON THIS LINE
 893                                   ;
 894                                   ;
 895                                   ;FULL DUPLEX RECEIVER TIMEOUTS COME HERE
 896  002122' 012764  002054' 004716' FDTO:    MOV     #JERR,RECRTN(R4)        ;SET UP CORRECT RETURN
 897  002130' 000167  001124                   JMP     RDTO            ;GO REPORT REC TIME OUT
 898                                   ;
```



```
 899                                   ;
 900                                   ;SUBROUTINE TO RECEIVE DATA
 901                                   ;CALLED WITH A JSR PC,REC
 902                                   ;AFTER SETTING LOCATIONS RECADR(R4),RECCNT(R4),RECTIM(R4),VCTSAV(R4),
 903                                   ;AND DVASAV(R4), RETURNS ON ERROR OR RETURNS
 904                                   ;2 BYTES PAST CALL IF NO ERROR
 905                                   ;
 906  002134' 012664  004716'          REC:    MOV     (R6)+,RECRTN(R4);SAVE RTN ADDR
 907  002140' 016401  004640'                  MOV     VCTSAV(R4),R1   ;GET VECTOR ADR
 908  002144' 012711  002244'                  MOV     #RECINT,(R1)    ;MOVE IN INT. ROUTINE ADR
 909  002150' 060421                           ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
 910  002152' 116711  175634                   MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
 911  002156' 005064  004672'                  CLR     ABORT(R4)       ;CLEAR ABORT FLAG
 912  002162' 012764  000002  004664'          MOV     #2,SYNCNT(R4)   ;MUST GET 2 SYN CHARACTERS
 913  002170' 005764  004766'                  TST     FDFLAG(R4)              ;FULL DUPLEX?
 914  002174' 001410                           BEQ     1$              ;BR IF NOT
 915  002176' 052774  000120  004634'          BIS     #120,@DVASAV(R4)        ;KICK OFF REC
 916  002204' 016402  004716'                  MOV     RECRTN(R4),R2   ;GET RETURN ADDRESS
 917  002210' 062702  000004                   ADD     #4,R2           ;FD RETURN
 918  002214' 010207                           MOV     R2,R7           ;RETURN
 919  002216' 012700  003260'          1$:     MOV     #RDTO,R0        ;RETURN ADDR
 920  002222' 016401  004660'                  MOV     RECTIM(R4),R1   ;HOW LONG TO WAIT
 921  002226' 012702  000000                   MOV     #0,R2           ;SHOWS ITS A REC WAIT
 922  002232' 052774  000120  004634'          BIS     #120,@DVASAV(R4);ENABLE INT AND READ
 923  002240' 000167  002012                   JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
 924                                   ;
 925                                   ;RECEIVER INTERRUPT ROUTINE
 926                                   ;
 927  002244' 000405                   RECINT: BR      1$
 928  002246' 010446                           MOV     R4,-(R6)        ;SAVE R4
 929  002250' 010346                           MOV     R3,-(R6)        ;SAVE R3
 930  002252' 012704  000002                   MOV     #2,R4           ;SET UP FOR SECOND LINE
 931  002256' 000403                           BR      2$
 932  002260' 010446                   1$:     MOV     R4,-(R6)        ;SAVE R4
 933  002262' 010346                           MOV     R3,-(R6)        ;SAVE R3
 934  002264' 005004                           CLR     R4              ;SET UP FOR FIRST LINE
 935  002266' 016403  004634'          2$:     MOV     DVASAV(R4),R3   ;GET THE DEVICE ADDRESS
 936  002272' 005764  004672'                  TST     ABORT(R4)       ;GIVE UP?
 937  002276' 001403                           BEQ     3$              ;BR IF NO
 938  002300' 042713  000160                   BIC     #160,(R3)       ;DISABLE REC INT,S
 939  002304' 000460                           BR      7$              ;AND LEAVE
 940  002306' 011364  004706'          3$:     MOV     (R3),ERRWD(R4)  ;SAVE STATUS
 941  002312' 016346  000002                   MOV     2(R3),-(R6)     ;SAVE CHARACTER AND ERR BITS
 942  002316' 005764  004664'                  TST     SYNCNT(R4)      ;ARE WE IN SYNC
 943  002322' 001421                           BEQ     5$              ;BR IF YES
 944  002324' 122716  000220                   CMPB    #220,(R6)       ;IS IT A DLE?
 945  002330' 001406                           BEQ     4$              ;BR IF YES
 946  002332' 126716  175636                   CMPB    SYNC,(R6)       ;IS IT A SYNC CHAR?
 947  002336' 001440                           BEQ     6$              ;YES, IGNORE IT
 948  002340' 042713  000020                   BIC     #20,(R3)        ;RESYNC IF NOT
 949  002344' 000435                           BR      6$              ;LEAVE
 950  002346' 005064  004664'          4$:     CLR     SYNCNT(R4)      ;SHOW WE'RE IN SYN
 951  002352' 005764  004766'                  TST     FDFLAG(R4)      ;FD SLAVE?
 952  002356' 100003                           BPL     5$              ;BR IF NOT
 953  002360' 012764  000010  004756'          MOV     #10,TSYNCT(R4)  ;LET TMT GO IF FULL DUP
 954  002366' 005767  175572           5$:     TST     REPORT          ;REPORT ERR NOW OR LATER?
```

```
 002372' 001013                 BNE     51$             ;BR IF LATER
 002374' 105764  004706'        TSTB    ERRWD(R4)       ;IS REC DONE BIT SET?
 002400' 100040                 BPL     RERR            ;BR IF NOT
 002402' 005716                 TST     (R6)            ;DATA ERROR?
 002404' 100006                 BPL     51$             ;BR IF NO DATA ERR
 002406' 012664  004706'        MOV     (R6)+,ERRWD(R4) ;SAVE INFORMATION
 002412' 042764  100000  004706' BIC     #100000,ERRWD(R4)       ;USE BIT 15 FOR FLAG TO RERR
 002420' 000431                 BR      RERR+2
 002422' 111674  004650'  51$:  MOVB    (R6),@RECADR(R4)        ;STORE THE CHAR IN BUFFER
 002426' 005264  004650'        INC     RECADR(R4)              ;FOR NEXT TIME
 002432' 005364  004654'        DEC     RECCNT(R4)      ;COUNT IT
 002436' 003406                 BLE     8$              ;BR IF DONE
 002440' 005726                 TST     (R6)+           ;FIX STACK
 002442' 052713  000120         BIS     #120,(R3)       ; READ ANOTHEP
 002446' 012603          7$:    MOV     (R6)+,R3        ;RESTORE R3
 002450' 012604                 MOV     (R6)+,R4        ;RESTORE R4
 002452' 000002                 RTI                     ;RETURN
 002454' 016467  004702' 000014  8$:    MOV     RECD(R4),RPIRQ+2
 002462' 005726                 TST     (R6)+           ;FIX STACK
 002464' 042713  000160         BIC     #160,(R3)       ;DISABLE REC INT.S
 002470' 012603                 MOV     (R6)+,R3        ;RESTORE REGS
 002472' 012604                 MOV     (R6)+,R4
 002474'                RPIRQ:
                                ;---------------------------------------------------------------
 002474' 000004  002704' 000000'        PIRQ$,RECDN,BEGIN               ; RTI AND QUEUE UP TO CONTINUE AT RECDN
                                ;---------------------------------------------------------------
                                ;
 002502' 005726                 RERR:   TST     (R6)+           ;FIX STACK
 002504' 012603                 MOV     (R6)+,R3        ;RESTORE REG
 002506' 042774  000160  004634' BIC     #160,@DVASAV(R4)        ;RESET THE RECEIVER
 002514' 012767  002542' 000014 MOV     #2$,1$+2                ;SET UP TO PIRQ TO
 002522' 060467  000010         ADD     R4,1$+2         ;THE RIGHT PLACE
 002526' 060467  000004         ADD     R4,1$+2
 002532' 012604                 MOV     (R6)+,R4        ;RESTORE REG
 002534'                        1$:
                                ;---------------------------------------------------------------
 002534' 000004  002542' 000000'        PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
                                ;---------------------------------------------------------------
 002542' 005004                 2$:     CLR     R4              ;SHOW ITS LINE 0
 002544' 000402                 BR      3$              ;
 002546' 012704  000002         MOV     #2,R4           ;SHOW ITS LINE 1
 002552' 016467  004634' 175270 3$:     MOV     DVASAV(R4),CSRA ;SET UP FOR ERR CALL
 002560' 016467  004706' 175264 MOV     ERRWD(R4),ACSR  ;SET UP FOR ERR CALL
 002566' 016401  004676'        MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE BYTE
 002572' 111102                 MOVB    (R1),R2         ;GET VALUE OF LINE #
 002574' 042702  177760         BIC     #177760,R2      ;THROW OTHER BITS AWAY
 002600' 005764  004706'        TST     ERRWD(R4)               ;IS IT A DATA OR LINE CHANGE ERR
 002604' 100010                 BPL     DATAER          ;BR IF DATA ERR
 002606' 010267  175242         MOV     R2,ASTAT        ;SET UP ERR INDICATION
 002612' 062767  002000  175234 ADD     #002000,ASTAT   ;SHOW WHICH TYPE
                                ;***************************************************************
 002620' 104404  000000'        ERROR$,BEGIN            ;DATA SET CHANGE
                                ;***************************************************************
 002624' 104400                 EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                ;
                                ;
```



```
 002626' 010267  175222         DATAER: MOV     R2,ASTAT        ;SET UP ERR LINE #
 002632' 116467  004707' 175215         MOVB    ERRWD+1(R4),ASTAT+1;GET ERR TYPE
 002640' 042767  107400  175206         BIC     #107400,ASTAT   ;LEAVE ONLY ERR BITS
 002646' 005264  005006'                INC     DTERR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
                                        ;***************************************************************
 002652' 104404  000000'                ERROR$,BEGIN            ;DATA ERR
                                        ;***************************************************************
 002656' 005767  002124                 TST     DTERR4
 002662' 001402                 BEQ     1$
 002664' 005004                 CLR     R4
 002666' 000402                 BR      2$
 002670' 012704  000002         1$:     MOV     #2,R4
 002674' 005364  005006'        2$:     DEC     DTERR4(R4)
                                ;
                                ;CHECK  ASTAT IN ERROR MESSAGE
                                ;0100XX = REC DATA PARITY ERROR
                                ;0200XX = FRAMING ERROR
                                ;0400XX = OVERRUN
                                ;XX = PHYSICAL LINE #
                                ;
 002700' 000167  000230                 JMP     RRET            ;GO DO AN ERROR EXIT
                                ;
                                ;
                                ;
 002704' 005004                 RECDN:  CLR     R4              ;SHOW WE'RE LINE 0
 002706' 000402                 BR      1$
 002710' 012704  000002         MOV     #2,R4           ;SHOW WE'RE LINE 1
 002714' 042774  177771  004634' 1$:     BIC     #177771,@DVASAV(R4)     ;SHUT UP DEVICE
 002722' 005064  004772'        CLR     RCVTO(R4)       ;RESET TIME OUT RETRY COUNT
 002726' 005064  005072'        CLR     RETTAB(R4)      ;KILL IMPENDING TIMEOUT
 002732' 005767  000000G        3$:     TST     CKMSG           ;IS CHECK MSG ROUTINE BUSY?
 002736' 001420                 BEQ     6$              ;BR IF NOT
 002740' 005264  005032'        INC     CHKR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
 002744' 104407  000000'        BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
 002750' 104407  000000'        BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
 002754' 005767  002052         TST     CHKR4
 002760' 001402                 BEQ     4$
 002762' 005004                 CLR     R4
 002764' 000402                 BR      5$
 002766' 012704  000002         4$:     MOV     #2,R4
 002772' 005364  005032'        5$:     DEC     CHKR4(R4)
 002776' 000755                 BR      3$
 003000' 016402  004644'        6$:     MOV     AMSGBF(R4),R2   ;GET REC DATA
 003004' 026762  177776G 000006 CMP     MSG-2,6(R2)     ;IS HEADER CRC OK?
 003012' 001407                 BEQ     7$              ;BR IF OK
 003014' 005767  175144         TST     REPORT          ;REPORT ERR NOW?
 003020' 001055                 BNE     X5              ;BR IF LATER
 003022' 012767  003000  175024 MOV     #3000,ASTAT     ;SHOW ERR TYPE
 003030' 000466                 BR      RECERR
 003032'                        7$:
 003032' 117467  004676' 000040 MOVB    @CURLIN(R4),9$  ;GET LINE #
 003040' 042767  177760  000032 BIC     #177760,9$      ;LEAVE ONLY LINE BITS
 003046' 005767  175112         TST     REPORT          ;REPORT BAD DATA?
 003052' 001403                 BEQ     8$              ;BR IF YES
 003054' 052767  100000  000016 BIS     #100000,9$      ;SET FLAG TO CKMSG
 003062' 062702  000010         8$:     ADD     #10,R2          ;GET ADDR OF MSG
```

```
1067  003066' 010267  000010                  MOV     R2,9$+2
1068  003072' 004567  000002G                 JSR     R5,CKMSG+2      ;GO CHECK MSG
1069  003076' 000000'                          BEGIN
1070  003100' 000000                  9$:     0
1071  003102' 000000                          0
1072  003104' 000417                          BR      X4              ;BAD RETURN
1073  003106' 066702  000000G                 ADD     MSGSIZ,R2       ;POINT IT TO MSG CRC, DATA WAS OK
1074  003112' 126722  000000G                 CMPB    CRCMSG,(R2)+    ;CHECK 1ST CRC BYTE
1075  003116' 001025                          BNE     CER
1076  003120' 126722  000001G                 CMPB    CRCMSG+1,(R2)+  ;CHECK SECOND
1077  003124' 001022                          BNE     CER
1078  003126' 062764  000002  004716'         ADD     #2,RECRTN(R4)   ;GOOD RETURN
1079  003134' 005864  005072'         RRET:   CLR     RETTAB(R4)      ;KILL IMPENDING TIMEOUTS
1080  003140' 000174  004716'                 JMP     @RECRTN(R4)     ;AND GO BACK
1081                                  ;
1082                                  ;
1083  003144' 005767  175014          X4:     TST     REPORT          ;REPORT ERR NOW OR LATER?
1084  003150' 001001                          BNE     X5              ;BR IF LATER
1085  003152' 000770                          BR      RRET            ;OTHERWISE RETURN
1086                                  ;
1087  003154' 117402  004676'         X5:     MOVB    @CURLIN(R4),R2  ;GET CURRENT LINE #
1088  003160' 042702  177760                  BIC     #177760,R2      ;
1089  003164' 105262  000252'                 INCB    ECOUNT(R2)      ;COUNT AN ERROR FOR THAT LINE
1090  003170' 000761                          BR      RRET            ;AND RETURN
1091                                  ;
1092                                  ;
1093  003172' 005767  174766          CER:    TST     REPORT          ;REPORT NOW OR LATER?
1094  003176' 001366                          BNE     X5              ;BR IF LATER
1095  003200' 012767  004000  174646          MOV     #004000,ASTAT   ;INDICATE ERROR TYPE
1096  003206' 117402  004676'         RECERR: MOVB    @CURLIN(R4),R2  ;GENERATE LINE #
1097  003212' 042702  177760                  BIC     #177760,R2      ;
1098  003216' 050267  174632                  BIS     R2,ASTAT        ;ADD IN LINE #
1099  003222' 005264  005012'                 INC     CRCR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
1100                                          ;*****************************************************************
1101  003226' 104404  000000'                 ERROR$,BEGIN            ;CRC ERROR
1102                                          ;*****************************************************************
1103  003232' 005767  001554                  TST     CRCR4
1104  003236' 001402                          BEQ     1$
1105  003240' 005004                          CLR     R4
1106  003242' 000402                          BR      2$
1107  003244' 012704  000002          1$:     MOV     #2,R4
1108  003250' 005364  005012'         2$:     DEC     CRCR4(R4)
1109  003254' 000167  177654                  JMP     RRET            ;BAD RETURN
1110                                  ;
1111                                  ;READ TIME OUT ROUTINE
1112                                  ;
1113  003260' 012764  177777  004672' RDTO:   MOV     #177777,ABORT(R4);TELL INT ROUTINES TO GIVE UP
1114  003266' 005264  004772'                 INC     RCVTO(R4)       ;COUNT ANOTHER RETRY
1115  003272' 026467  004772' 174672          CMP     RCVTO(R4),ERRLVL        ;ENOUGH TRYS?
1116  003300' 001043                          BNE     4$              ;BR IF NO
1117  003302' 017467  004634' 174542          MOV     @DVASAV(R4),ACSR;GET REC STATUS
1118  003310' 042774  177771  004634'         BIC     #177771,@DVASAV(R4)     ;KILL ALL BUT DTR
1119  003316' 016467  004634' 174524          MOV     DVASAV(R4),CSRA ;PUT ADDR IN FOR ERR CALL
1120  003324' 016401  004676'                 MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE BYTE
1121  003330' 111102                          MOVB    (R1),R2         ;GET VALUE OF PHY LINE #
1122  003332' 042702  177760                  BIC     #177760,R2      ;THROW OTHER BITS AWAY
```



```
1123  003336' 062702  001000                  ADD     #1000,R2        ;INDICATE REC TIMEOUT ERR
1124  003342' 010267  174506                  MOV     R2,ASTAT        ;SET UP FOR ER CALL
1125  003346' 005264  005016'                 INC     RCTOR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
1126                                          ;*****************************************************************
1127  003352' 104404  000000'                 ERROR$,BEGIN            ;REC TIMED OUT
1128                                          ;*****************************************************************
1129  003356' 005767  001434                  TST     RCTOR4
1130  003362' 001402                          BEQ     1$
1131  003364' 005004                          CLR     R4
1132  003366' 000402                          BR      2$
1133  003370' 012704  000002          1$:     MOV     #2,R4
1134  003374' 005364  005016'         2$:     DEC     RCTOR4(R4)
1135  003400' 005064  004772'                 CLR     RCVTO(R4)       ;RESET RETRY COUNT
1136  003404' 000174  004716'         3$:     JMP     @RECRTN(R4)     ;ERROR RTN
1137                                  ;
1138  003410' 016401  004676'         4$:     MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
1139  003414' 105361  000020                  DECB    20(R1)          ;DEC ERROR COUNT TO CORRECT FOR
1140                                                                  ;INSUING INC
1141  003420' 005364  004712'                 DEC     LOOP(R4)        ;CORRECT FOR LATER CALLING THIS A GOOD RUN
1142  003424' 000767                          BR      3$              ;RETURN
1143                                  ;
1144                                  ;
1145                                  ;SUBROUTINE TO TRANSMIT DATA
1146                                  ;CALLED WITH A JSR PC,TMT
1147                                  ;AFTER SETTING LOCATIONS TXADR1(R1), TXADR2(R4), TXCNT1(R4),TXCNT2(R4), TMTIM(R4),
1148                                  ;VCTSAV(R4), DVASAV(R4), SUBROUTINE CTS MUST BE CALLED FIRST.
1149                                  ;RETURNS ON ERROR, RETURNS 2 BYTES PAST NORMAL RETURN
1150                                  ;IF NO ERROR
1151                                  ;
1152  003426' 012664  004746'         TMT:    MOV     (SP)+,TMTRET(R4)  ;SAVE RTN ADDRESS
1153  003432' 005064  004752'                 CLR     TMTDON(R4)      ;INITIALIZE FLAG
1154  003436' 016401  004640'                 MOV     VCTSAV(R4),R1   ;GET REC VECTOR ADDRESS
1155  003442' 062701  000004                  ADD     #4,R1           ;MAKE IT TMT VECTOR
1156  003446' 012711  003576'                 MOV     #TMTINT,(R1)    ;MOVE IN INT ROUTINE ADDRESS
1157  003452' 060421                          ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
1158  003454' 116711  174332                  MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
1159  003460' 005064  004672'                 CLR     ABORT(R4)       ;CLEAR TIME OUT ABORT FLAG
1160  003464' 012764  000010  004756'         MOV     #10,TSYNCT(R4)  ;SEND TEN SYN CHARACTERS
1161  003472' 016403  004634'                 MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
1162  003476' 005764  004766'                 TST     FDFLAG(R4)      ;ARE WE FULL DUPLEX?
1163  003502' 001417                          BEQ     2$              ;BR IF NOT, ELSE
1164  003504' 100003                          BPL     1$              ;SKIP IF MASTER
1165  003506' 012764  177777  004756'         MOV     #177777,TSYNCT(R4)      ;MAKE TMT WAIT FOR A REC'D SYNC
1166  003514' 012763  000120  000004  1$:     MOV     #120,4(R3)      ;SET TMT INT ENABLE
1167  003522' 116763  174446  000006          MOVB    SYNC,6(R3)      ;SEND A FILL CHAR
1168  003530' 016402  004746'                 MOV     TMTRET(R4),R2   ;GET BAD RET ADDR
1169  003534' 062702  000004                  ADD     #4,R2           ;MAKE IT THE FULL DUPLEX RETURN
1170  003540' 010207                          MOV     R2,R7           ;AND RETURN
1171                                  ;
1172                                  ;
1173  003542' 012700  004022'         2$:     MOV     #TXTO,R0        ;TIME OUT RETURN ADDR
1174  003546' 016401  004742'                 MOV     TMTTIM(R4),R1   ;HOW LONG TO WAIT
1175  003552' 012702  000004                  MOV     #4,R2           ;SHOW ITS TRANSMITTER
1176  003556' 012763  000120  000004          MOV     #120,4(P3)      ;SET TMT INT ENABLE AND SEND BIT
1177  003564' 116763  174404  000006          MOVB    SYNC,6(R3)      ;SEND A FILL CHAR
1178  003572' 000167  000460                  JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
```

```
                                        ;TMT INTERRUPT ROUTINE
                                        ;
003576' 000405                  TMTINT: BR      1$
003600' 010446                          MOV     R4,-(SP)        ;SAVE R4
003602' 010346                          MOV     R3,-(SP)        ;SAVE R3
003604' 012704  000002                  MOV     #2,R4           ;SET TO LINE 1
003610' 000403                          BR      2$
003612' 010446                  1$:     MOV     R4,-(SP)        ;SAVE R4
003614' 010346                          MOV     R3,-(SP)        ;SAVE R3
003616' 005004                          CLR     R4              ;SET TO LINE 0
003620' 005764  004672'         2$:     TST     ABORT(R4)       ;GIVE UP?
003624' 100417                          BMI     TIEXIT          ;BR IF YES
003626' 016403  004634'                 MOV     DVASAV(R4),R3   ;GET DVA
003632' 005764  004732'                 TST     TXCNT1(R4)      ;FIRST BUFFER TMT'ED?
003636' 001015                          BNE     FTBUF           ;BR IF NOT
003640' 005764  004736'                 TST     TXCNT2(R4)      ;SECOND BUFFER TMT'ED?
003644' 001434                          BEQ     TIDONE          ;BR IF YES
003646' 117463  004726' 000006          MOVB    @TXADR2(R4),6(R3)  ;SEND NEXT BYTE
003654' 005264  004726'                 INC     TXADR2(R4)      ;MOV ADDRESS POINTER
003660' 005364  004736'                 DEC     TXCNT2(R4)      ;COUNT IT
003664' 012603                  TIEXIT: MOV     (R6)+,R3        ;RESTORE R3
003666' 012604                          MOV     (R6)+,R4        ;RESTORE R4
003670' 000002                          RTI                     ;RETURN
003672' 005764  004756'         FTBUF:  TST     TSYNCT(R4)      ;SENT ENOUGH FILL CHARS?
003676' 001407                          BEQ     2$              ;BR IF YES
                                                                ;IF NEGITIVE
003700' 100402                          BMI     1$              ;SEND TILL RECINT CLEARS TSYNCT
003702' 005364  004756'                 DEC     TSYNCT(R4)      ;ELSE COUNT IT
003706' 116763  174262  000006  1$:     MOVB    SYNC,6(R3)      ;SEND IF YES
003714' 000763                          BR      TIEXIT          ;AND LEAVE
003716' 117463  004722' 000006  2$:     MOVB    @TXADR1(R4),6(R3)  ;SEND NEXT BYTE
003724' 005264  004722'                 INC     TXADR1(R4)      ;MOVE ADDRESS POINTER
003730' 005364  004732'                 DEC     TXCNT1(R4)      ;COUNT IT
003734' 000753                          BR      TIEXIT          ;AND LEAVE
                                        ;
                                        ;ALL CHAR'S HAVE BEEN TMT'ED
003736' 042763  000120  000004  TIDONE: BIC     #120,4(R3)      ;KILL TMT'ER
003744' 012603                          MOV     (R6)+,R3        ;RESTORE R3
003746' 005064  005076'                 CLR     RETTAB+4(R4)    ;KILL IMPENDING TIMEOUT
003752' 062764  000002  004746'         ADD     #2,TMTRET(R4)   ;MAKE IT A GOOD RETURN
003760' 005704                          TST     R4              ;LINE 0 OR 1
003762' 001004                          BNE     1$              ;BR IF LINE 1
003764' 012604                          MOV     (R6)+,R4        ;RESTORE R4
                                        ;---------------------------------------------------------------
003766' 000004  004004' 000000'         PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
                                        ;---------------------------------------------------------------
003774' 012604                  1$:     MOV     (R6)+,R4        ;RESTORE R4
                                        ;---------------------------------------------------------------
003776' 000004  004012' 000000'         PIRQ$,3$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 3$
                                        ;---------------------------------------------------------------
004004' 005004                  2$:     CLR     R4              ;SET TO LINE 0
004006' 000174  004746'                 JMP     @TMTRET(R4)     ;RETURN
004012' 012704  000002          3$:     MOV     #2,R4           ;SET TO LINE 1
004016' 000174  004746'                 JMP     @TMTRET(R4)     ;RETURN
                                        ;
                                        ;
```



```
                                        ;ROUTINE IF TRANSMIT TIMES OUT
                                        ;
004022' 012764  177777  004672' TXTO:   MOV     #177777,ABORT(R4)       ;TELL INT ROUTINES TO GIVE UP
004030' 016467  004634' 174012          MOV     DVASAV(R4),CSRA ;GET ADDR OF STATUS
004036' 062767  000004  174004          ADD     #4,CSRA         ;MAKE IT TMT CSR
004044' 017767  174000  174000          MOV     @CSRA,ACSR      ;SAVE STATUS
004052' 042777  177767  173770          BIC     #177767,@CSRA           ;KILL TRANSMISSION AND INT
004060' 016401  004676'                 MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
004064' 111102                          MOVB    (R1),R2         ;GET VALUE
004066' 042702  177760                  BIC     #177760,R2      ;LEAVE ONLY LINE #
004072' 062702  005000                  ADD     #5000,R2        ;INDICATE TMT TIMEOUT
004076' 010267  173752                  MOV     R2,ASTAT        ;SET UP FOR ERROR CALL
004102' 005264  005022'                 INC     TXTOR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
                                        ;***************************************************************
004106' 104404  000000'                 ERROR$,BEGIN            ;TMT TIMEOUT
                                        ;***************************************************************
004112' 005767  000704                  TST     TXTOR4
004116' 001402                          BEQ     1$
004120' 005004                          CLR     R4
004122' 000402                          BR      2$
004124' 012704  000002          1$:     MOV     #2,R4
004130' 005364  005022'         2$:     DEC     TXTOR4(R4)
004134' 000174  004746'                 JMP     @TMTRET(R4)     ;BAD RETURN
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;
                                        ;SUB GETS CLEAR TO SEND FOR LINE IN DVASAV(R4) AND VCTSAV(R4)
                                        ;
004140' 012664  004762'         CTS:    MOV     (R6)+,CTSRET(R4)        ;SAVE RET ADDRESS
004144' 016401  004640'                 MOV     VCTSAV(R4),R1   ;GET VECTOR ADDR
004150' 012711  004206'                 MOV     #CTSINT,(R1)    ;MOV IN INT ROUTINE ADDR
004154' 060421                          ADD     R4,(R1)+        ;SHOWS WHICH LINE
004156' 116711  173630                  MOVB    BR1,(R1)        ;GET BR1 LEVEL
004162' 032774  020000  004634'         BIT     #20000,@DVASAV(R4)  ;IS CLEAR TO SEND UP?
004170' 001402                          BEQ     RTSSET          ;BR IF NOT
004172' 016407  004762'                 MOV     CTSRET(P4),R7   ;RETURN
004176' 012774  000046  004634' RTSSET: MOV     #46,@DVASAV(R4)  ;SET REQ TO SEND
004204' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                        ;INT ROUTINE
004206' 000403                  CTSINT: BR      1$              ;DIFFERENT PIRQ'S
                                        ;---------------------------------------------------------------
004210' 000004  004224' 000000'         PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
                                        ;---------------------------------------------------------------
004216'                         1$:
                                        ;---------------------------------------------------------------
004216' 000004  004232' 000000'         PIRQ$,3$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 3$
                                        ;---------------------------------------------------------------
004224' 012704  000002          2$:     MOV     #2,R4           ;SET UP FOR LINE 1
004230' 000401                          BR      4$
004232' 005004                  3$:     CLR     R4              ;SET UP FOR LINE 0
004234' 032774  020000  004634' 4$:     BIT     #20000,@DVASAV(R4)  ;IS CLEAR TO SEND SET?
004242' 001755                          BEQ     RTSSET          ;GO TRY AGAIN IF NOT
004244' 042774  000040  004634'         BIC     #40,@DVASAV(R4) ;DISABLE DATA SET CHG INT'S
004252' 016407  004762'                 MOV     CTSRET(R4),R7   ;RETURN
```

```
                                ;
                                ;
                                ;
                                ;ALL ROUTINES COME HERE WHILE WAITING FOR I/O
                                ;ROUTINE EITHER KICKS OFF SECOND LINE, NOTIFYS
                                ;ROUTINES OF TIMEOUTS, OR JUST DOES BREAKS.
                                ;I/O ROUTINES THAT FINISH UP BEFORE TIMEOUT VALUE
                                ;MUST CLEAR LOCATION RETTAB(R4) +0 IF RECEIVER OP +4
                                ;IF TMT.
                                ;CALLED WITH A JMP TO TIMKEP AFTER LOADING
                                ; R0 = WHERE TO RETURN IF TIME OUT
                                ; R1 = HOW LONG TO WAIT (IN SECONDS)
                                ; R2 = 0 IF ITS A REC TIMEING OR 4 IF ITS A TMT WAIT
                                ; R4 = 0 FOR ACTIVE LINE 0, 2 FOR ACTIVE LINE 1
                                ;
                                ;
004256' 060402                  TIMKEP: ADD     R4,R2           ;FIGURE OUT WHICH TABLE ENTRY
004260' 010162  005062'                 MOV     R1,TIMTAB(R2)   ;STORE HOW LONG TO WAIT
004264' 067762  000000G 005062'         ADD     @TIME,TIMTAB(R2)        ;ADD PRESENT TIME
004272' 010462  005102'                 MOV     R4,R4TAB(R2)    ;SAVE WHICH ACTIVE LINE
004276' 010062  005072'                 MOV     R0,RETTAB(R2)   ;WHERE TO RET TO, PLUS
                                                                ;IF THIS ENTRY IN TABLE IS
                                                                ;0 THEN NO TIMEOUT WILL
                                                                ;OCCUR FOR THIS ENTRY SET
004302' 022767  000002  000360          CMP     #2,LINACT       ;ARE 2 LINES GOING?
004310' 001410                          BEQ     TIMCHK          ;BR IF YES
004312' 005704                          TST     R4              ;ARE WE LINE 0?
004314' 001402                          BEQ     1$              ; BR IF YES
004316' 005004                          CLR     R4              ; IF NO MAKE IT 0
004320' 000402                          BR      2$
004322' 012704  000002          1$:     MOV     #2,R4           ;WE WERE 0 SO NOW DO 1
004326' 000167  174332          2$:     JMP     GETNXT          ;GO DO OTHER LINE
                                ;
                                ;
004332' 012702  005072'         TIMCHK: MOV     #RETTAB,R2      ;GET ADDR OF TIME STATUSES
004336' 005722                  1$:     TST     (R2)+           ;ANY WAIT GOING ON?
004340' 001026                          BNE     4$              ;BR IF YES
004342' 020227  005102'         11$:    CMP     R2,#RETTAB+10   ;ARE WE DONE LOOKING
004346' 001373                          BNE     1$              ;BR BACK IF NO
004350' 005767  000424                  TST     PASTIM          ;TIME FOR END OF PASS
004354' 001402                          BEQ     2$              ;BR IF NO
004356' 000167  000064                  JMP     EPASS           ;GO REPORT END OF PASS
                                ;
004362' 005767  000410          2$:     TST     BRKFLG          ;ARE WE ALREADY IN A BREAK?
004366' 001401                          BEQ     3$              ;BR IF NO
004370' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERUPT.
                                ;
004372' 012767  000001  000376  3$:     MOV     #1,BRKFLG               ;SHOW WERE DOING BREAK
004400' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
004404' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
004410' 005067  000362                  CLR     BRKFLG          ;SHOW WERE BACK
004414' 000746                          BR      TIMCHK
                                ;
004416' 005742                  4$:     TST     -(R2)           ;MOVE R2 BACK
004420' 027762  000000G 177770          CMP     @TIME,-10(R2)   ;IS TIME UP?
004426' 003002                          BGT     5$              ;BR IF YES
```



```
004430' 005722                          TST     (R2)+           ;BUMP R2 BACK AGAIN
004432' 000743                          BR      11$             ;GO CHECK OTHERS
                                ;
004434' 016204  000010          5$:     MOV     10(R2),R4       ;SET R4 CORRECTLY FOR WHICH LINE
004440' 011203                          MOV     (R2),R3         ;SAVE RETURN
004442' 005012                          CLR     (R2)            ;CLEAR THIS ENTRY
004444' 010307                          MOV     R3,R7           ;JMP TO TIME OUT ROUTINE
                                ;
                                ;
                                ;
                                ;EPASS DOES END OF PASS CALL AND PRINTS ERR SUMMARY
                                ;IF LOCATION REPORT IS NOT 0
                                ;
004446' 005767  173512          EPASS:  TST     REPORT          ;DO ERR SUM OR NOT
004452' 001461                          BEQ     5$              ;BR IF NO
004454' 005267  000146                  INC     MSGCNT          ;COUNT A PASS
004460' 026767  000142  173476          CMP     MSGCNT,REPORT   ;TIME FOR A MSG?
004466' 001053                          BNE     5$              ;IF NOT EVEN 8, SKIP MSG
004470' 005067  000132                  CLR     MSGCNT
004474' 012700  000212'                 MOV     #PHYTAB,R0      ;FIND WHICH LINES ARE RUNNING
004500' 012701  011252'                 MOV     #SUM1+4,R1      ;POINTS TO MSG
004504' 105710                  1$:     TSTB    (R0)            ;IS LINE IN USE?
004506' 100403                          BMI     2$              ;BR IF YES
004510' 012711  054130                  MOV     #54130,(R1)     ;PUT XX IN # OF ERRS
004514' 000427                          BR      4$
004516' 116002  000040          2$:     MOVB    32.(R0),R2      ;GET # OF ERRS
004522' 105060  000040                  CLRB    32.(R0)         ;RESET ERR COUNT
004526' 022702  000077                  CMP     #77,R2          ;ANY GOOD MSGS
004532' 002416                          BLT     3$              ;BR IF NO
004534' 010203                          MOV     R2,R3           ;PUT IN MSG
004536' 042703  177770                  BIC     #177770,R3      ;LEAVE ONLY LOW DIGIT
004542' 062703  000060                  ADD     #60,R3  ;MAKE IT ASCII
004546' 110361  000001                  MOVB    R3,1(R1)        ;STORE IT AWAY
004552' 006202                          ASR     R2              ;GET NEXT DIGIT
004554' 006202                          ASR     R2
004556' 006202                          ASR     R2
004560' 062702  000060                  ADD     #60,R2          ;MAKE IT ASCII
004564' 110211                          MOVB    R2,(R1) ;PUT IT IN MSG
004566' 000402                          BR      4$
004570' 012711  042102          3$:     MOV     #42102,(R1)     ;PUT IN AN 'BD' FOR BAD
004574' 005200                  4$:     INC     R0              ;BUMP LINE POINTER
004576' 062701  000010                  ADD     #10,R1          ;POINT TO NXT MSG
004602' 020027  000232'                 CMP     R0,#PHYTAB+20   ;DONE
004606' 001336                          BNE     1$              ;GO DO MORE
004610' 104411  004622' 000000'         MSGN$,MSUM,BEGIN        ;ASCII MESSAGE CALL WITH COMMON HEADER
                                ;
004616'                         5$:
004616' 104402  000000'                 ENDPS$,BEGIN            ;SIGNAL END OF PASS. CONTINUE AT RESTRT
                                ;
004622' 011162'                 MSUM:   SUM
004624' 177777                          -1
                                ;
004626' 000000                  MSGCNT: 0                       ;COUNTS END PASSES FOR MESSAGE CALL
004630' 000000                  MASK:   0                       ;USED FOR BIC'ING IN LOOPS
004632' 000000                  NXTLIN: 0                       ;ADR OF NEXT LIN
```

```
1403  004634' 000000                  DVASAV: 0                       ;HOLDS DVA FOR ACTIVE L#0
1404  004636' 000000                          0                       ;HOLDS DVA FOR ACTIVE L#1
1405  004640' 000000                  VCTSAV: 0                       ;HOLDS VCT FOR ACTIVE L#0
1406  004642' 000000                          0                       ;HOLDS VECTOR FOR ACTIVE L#1
1407  004644' 005112'                 AMSGBF: MSGBF0                  ;ADDRESS OF MSG BUFFER 0
1408  004646' 007132'                         MSGBF1                  ;ADDRESS OF MSG BUFFER 1
1409  004650' 000000                  RECADR: 0                       ;CURRENT REC WORD POINTER 0
1410  004652' 000000                          0                       ;CURRENT REC WORD POINTER 1
1411  004654' 000000                  RECCNT: 0                       ;REC BYTE COUNT FOR 0
1412  004656' 000000                          0                       ;REC BYTE COUNT FOR 1
1413  004660' 000000                  RECTIM: 0                       ;TEMP HOLDS TIME OUT VALUE FOR A REC
1414  004662' 000000                          0                       ;SAME BUT FOR LINE 1
1415  004664' 000000                  SYNCNT: 0                       ;COUNTS HOW MANY SYN CHARACTERS REC
1416  004666' 000000                          0                       ;SHOULD WAIT FOR.
1417  004670' 000000                  LINACT: 0                       ;HOLD THE NUMBER OF LINES RUNNING,0-2
1418  004672' 000000                  ABORT:  0                       ;ABORT FLAG FOR L#0
1419  004674' 000000                          0                       ;ABORT FLAG FOR L#1
1420  004676' 000000                  CURLIN: 0                       ;POINTS TO STATUS BYTE FOR CURRENT L#0
1421  004700' 000000                          0                       ;POINTS TO STATUS BYTE FOR CURRENT L#1
1422  004702' 002704'                 RECD:   RECDN                   ;WHERE TO PIRQ AFTER REC DONE
1423  004704' 002710'                         RECDN+4                 ;
1424  004706' 000000                  ERRWD:  0                       ;HOLDS ERR WORDS TEMPORARILY
1425  004710' 000000                          0                       ;HOLDS ERR WORDS TEMPORARILY
1426  004712' 000000                  LOOP:   0                       ;HOLDS LOOP COUNT FOR LINE # 0
1427  004714' 000000                          0                       ;HOLDS LOOP COUNT FOR LINE # 1
1428  004716' 000000                  RECRTN: 0                       ;HOLDS RETURN ADDR FOR LINE #0
1429  004720' 000000                          0                       ;HOLDS RETURN ADDR FOR LINE #1
1430  004722' 000000                  TXADR1: 0                       ;HOLDS 1ST ADDR TO TMT FROM, LINE 0
1431  004724' 000000                          0                       ;SAME FOR LINE 1
1432  004726' 000000                  TXADR2: 0                       ;HOLDS 2ND ADDR TO TMT FROM, LINE 0
1433  004730' 000000                          0                       ;SAME FOR LINE 1
1434  004732' 000000                  TXCNT1: 0                       ;HOLDS 1ST TMT COUNT FOR LINE 0
1435  004734' 000000                          0                       ;SAME FOR LINE 1
1436  004736' 000000                  TXCNT2: 0                       ;HOLDS 2ND TMT COUNT FOR LINE 0
1437  004740' 000000                          0                       ;SAME FOR LINE 1
1438  004742' 000000                  TMITIM: 0                       ;HOLDS TIME OUT TIME FOR LINE 0
1439  004744' 000000                          0                       ;SAME FOR LINE 1
1440  004746' 000000                  TMTRET: 0                       ;HOLDS RTN ADDR FOR TMT, LINE 0
1441  004750' 000000                          0                       ;SAME FOR LINE 1
1442  004752' 000000                  TMTDON: 0                       ;FLAG FOR WHEN A TMT IS DONE
1443  004754' 000000                          0                       ;SAME FOR OTHER LINE
1444  004756' 000000                  TSYNCT: 0                       ;HOLDS # OF SYN CHAR TO SEND, LINE 0
1445  004760' 000000                          0                       ;SAME FOR LINE 1
1446  004762' 000000                  CTSRET: 0                       ;HOLDS RETURN ADDR FOR CTS
1447  004764' 000000                          0                       ;SAME FOR LINE 1
1448  004766' 000000                  FDFLAG: 0                       ;0 IF HALF DUPLEX, 1 IF FULL
1449  004770' 000000                          0
1450  004772' 000000                  RCVTO:  0                       ;COUNTS REC TIME OUT RETRYS
1451  004774' 000000                          0
1452  004776' 000000                  BRKFLG: 0                       ;IS A ONE DURING BREAKS
1453  005000' 000000                  PASTIM: 0                       ;HAS A 1 WHEN ITS TIME FOR END OF PASS
1454  005002' 000000                  DROPR4: 0       ;EXIT FLAGS
1455  005004' 000000                          0
1456  005006' 000000                  DTERR4: 0
1457  005010' 000000                          0
1458  005012' 000000                  CRCR4:  0
```



```
1459  005014' 000000                          0
1460  005016' 000000                  RCTOR4: 0
1461
1462  005020' 000000                          0
1463  005022' 000000                  TXTOR4: 0
1464  005024' 000000                          0
1465  005026' 000000                  FDBKR4: 0
1466
1467  005030' 000000                          0
1468  005032' 000000                  CHKR4:  0
1469  005034' 000000                          0
1470
1471                                  ;
1472                                  ;
1473                                  ;R1 HAS ADDR OR VECTOR
1474                                  ;R0 HAS LINE # IN LOW BITS, RETURNS WITH ADDR OR VECTOR IN R1
1475  005036'                         GETADR:
1476  005036' 010046                          MOV     R0,-(R6)        ;SAVE R0
1477  005040' 042700  177760                  BIC     #177760,R0      ;SAVE ONLY LINE #
1478  005044' 001404                  1$:     BEQ     2$              ;DONE
1479  005046' 062701  000010                  ADD     #10,R1          ;BUMP TO NEXT LINE
1480  005052' 005300                          DEC     R0              ;ADDED ENOUGH TIMES?
1481  005054' 000773                          BR      1$              ;GO BACK
1482  005056' 012600                  2$:     MOV     (R6)+,R0        ;RESTORE R0
1483  005060' 000207                          RTS     PC              ;RETURN WITH DVA IN R1
1484                                  ;
1485                                  ;
1486  005062' 000000                  TIMTAB: 0                       ;WHEN TO TIME OUT FOR REC,LINE 0
1487  005064' 000000                          0                       ;WHEN TO TIME OUT FOR REC,LINE 1
1488  005066' 000000                          0                       ;WHEN TO TIME OUT FOR TXT,LINE 0
1489  005070' 000000                          0                       ;WHEN TO TIME OUT FO TXT,LINE 1
1490  005072' 000000                  RETTAB: 0                       ;WHERE TO RETURN FOR REC LINE 0
1491  005074' 000000                          0                       ;WHERE TO RETURN FOR REC LINE 1
1492  005076' 000000                          0                       ;WHERE TO RETURN FOR TXT LINE 0
1493  005100' 000000                          0                       ;WHERE TO RETURN FOR TXT LINE 1
1494  005102' 000000                  R4TAB:  0                       ;HOLD R4 FOR RETURNS
1495  005104' 000000                          0
1496  005106' 000000                          0
1497  005110' 000000                          0
1498                                  ;
1499                                  ;
1500  005112' 031010                  MSGBF0: .BLKW   520.            ;REC BUFFER FOR LINE 0
1501  007132' 001010                  MSGBF1: .BLKW   520.            ;REC BUFFER FOR LINE 1
1502  011152' 177777                  FILL:   177777                  ;4 CHARACTERS OF FILLER BYTES
1503  011154' 177777                          177777
1504  011156' 011450'                 ADROP:  MDROP
1505  011160' 177777                          -1
1506                                  .EVEN
1507  011162' 005015  044523  041516  SUM:    .ASCII  <15><12>'SINCE LAST MSG PER LINE, IN OCTAL '<15><12>
1508  011170' 020105  040514  052123
1509  011176' 046440  043523  050040
1510  011204' 051105  046040  047111
1511  011212' 026105  044440  020116
1512  011220' 041517  040524  020114
1513  011226' 005015
1514  011230' 044514  042516  021440          .ASCII  'LINE #  ERRS'<15><12>
```

```
 1515  011236' 020040  051105  051522
 1516  011244' 005015
 1517  011246' 030060  026455  030060  SUM1:   .ASCII  '00--00'<15><12>
 1518  011254' 005015
 1519  011256' 030460  026455  030060          .ASCII  '01--00'<15><12>
 1520  011264' 005015
 1521  011266' 031060  026455  030060          .ASCII  '02--00'<15><12>
 1522  011274' 005015
 1523
 1524
 1525  011276' 031460  026455  030060          .ASCII  '03--00'<15><12>
 1526  011304' 005015
 1527  011306' 032060  026455  030060          .ASCII  '04--00'<15><12>
 1528  011314' 005015
 1529
 1530
 1531
 1532  011316' 032460  026455  030060          .ASCII  '05--00'<15><12>
 1533  011324' 005015
 1534  011326' 033060  026455  030060          .ASCII  '06--00'<15><12>
 1535  011334' 005015
 1536  011336' 033460  026455  030060          .ASCII  '07--00'<15><12>
 1537  011344' 005015
 1538  011346' 030061  026455  030060          .ASCII  '10--00'<15><12>
 1539  011354' 005015
 1540  011356' 030461  026455  030060          .ASCII  '11--00'<15><12>
 1541  011364' 005015
 1542  011366' 031061  026455  030060          .ASCII  '12--00'<15><12>
 1543  011374' 005015
 1544  011376' 031461  026455  030060          .ASCII  '13--00'<15><12>
 1545  011404' 005015
 1546  011406' 032061  026455  030060          .ASCII  '14--00'<15><12>
 1547  011414' 005015
 1548  011416' 032461  026455  030060          .ASCII  '15--00'<15><12>
 1549  011424' 005015
 1550  011426' 033061  026455  030060          .ASCII  '16--00'<15><12>
 1551  011434' 005015
 1552  011436' 033461  026455  030060          .ASCII  '17--00'<15><12>
 1553  011444' 005015
 1554
 1555  011446'    000                          .BYTE   0
 1556          011450'                         .EVEN
 1557                                  ;
 1558  011450' 044514  042516  053440  MDROP:  .ASCIZ  'LINE WAS DROPPED'
 1559  011456' 051501  042040  047522
 1560  011464' 050120  042105     000
 1561          000001                  .END
```



```
ABORT   000672R          896*   911*   936   1113*  1159*  1189   1237*  1418#
ACSP    000052R          469#   997*  1117*  1240*
ADDP    000006R          448#   643    702
ADROP   011156R          787   1504#
AMSGBF  000644R          745    814    840   1053   1407#
ASB     000056R          472#
ASTAT   000054R          471#  1003*  1004*  1011*  1012*  1013*  1058*  1095*  1098*  1124*  1246*
AWAS    000060R          473#
BDCNV = ******  G          1#
BEGIN   000000R          445#   624    625    658    659    672    724    787    868    869    979    991   1006
                        1016   1044   1045   1069   1101   1127   1223   1227   1249   1278   1282   1339   1340
                        1391   1395
BIT0  = 000001           482#
BIT1  = 000002           482#
BIT10 = 002000           482#
BIT11 = 004000           482#
BIT12 = 010000           482#
BIT13 = 020000           482#
BIT14 = 040000           482#
BIT15 = 100000           482#
BIT2  = 000004           482#
BIT3  = 000010           482#
BIT4  = 000020           482#
BIT5  = 000040           482#
BIT6  = 000100           482#
BIT7  = 000200           482#
BIT8  = 000400           482#
BIT9  = 001000           482#
BREAK$= 104407           482#   658    659    868    869   1044   1045   1339   1340
BRKFLG  004776R          691*   728   1334   1338*  1341*  1452#
BR1     000012R          450#   910   1158   1269
BR2     000013R          451#
CDATA$= 104414           482#
CFR     003172R         1075   1077   1093#
CHKR4   005032R          687*   688*  1043*  1046   1051*  1468#
CKMSG = ******  G        482#  1041   1068
CRCMSG= ******  G        482#  1074   1076
CRCR4   005012R          679*   680*  1099*  1103   1108*  1458#
CSRA    000050R          467#   996*  1119*  1238*  1239*  1240   1241*
CTS     004110R          759    805    839   1265#
CTSINT  004206R         1267   1276#
CTSRET  004762R         1265*  1272   1290   1446#
CURLIN  004676R          697*   778    827    887    998   1061   1087   1096   1120   1138   1242   1420#
DATAER  002626R         1002   1011#
DATCK$= 104417           482#
DATER$= 104405           482#
DERR    001300R          762    774#
DEVSET= ******  G        483#   626
DONE    000774R          695    716#
DROP    001344R          781    785#   830    892
DROPR4  005002R          675*   676*   786*   788    793*  1454#
DTERR4  005006R          677*   678*  1014*  1018   1023*  1456#
DVASAV  004634R          704*   763*   768    774*   809*   826*   880*   891*   915*   922*   935   984*   996
                        1038*  1117   1118*  1119   1161   1191   1238   1270   1273*  1287   1289*  1403#
DVID1   000014R          452#   622
ECOUNT  000252R          552#  1089*
```

```
ENDPS$= 104402           482#  1395
END$  = 104403           482#   624     672     724
EPASS   000446R          730   1332    1360#
EPRCNT  000030R          459#
ERRLVL  000172R          491#   780     829     889    1115
ERRN$ = 104410           482#
ERROR$= 104404           482#  1006    1016    1101    1127    1249
ERRTAB  000232R          533#
ERRWD   004706R          940*   956     960*    961*    997    1001    1012    1424#
EXIT$ = 104400           482#   733     862    1008    1274    1336
FDBKR4  005026R          685*   686*    867*    870     875*   1465#
FDFLAG  004766R          735*   739*    838*    913     951    1162    1448#
FDRCDN  001760R          845    865#    876     890
FDTO    002122R          855    896#
FERR    001314R          751    777#
FERR1   001316R          776    778#
FILL    011152R          758    804     851    1502#
FTBUF   003672R         1193   1202#
GERR    001554R          808    826#
GETADR  005036R          703    706    1475#
GETNXT  000664R          694#   701     771    1322
GETPA$= 104413           482#
GWBUF$= 104412           482#
HERR    001562R          819    827#
INIT    000022R          456#
JERR    002054R          844    853     885#    896
LINACT  004670R          693*   708*    716     767*   1315    1417#
LNDONE  001254R          767#   794     823     881
LOOP    004712R          709*   764*    765     820*    821     877*    878    1141*   1426#
LPCNT   000166R          489#   765     821     878
MAP22$= 104415           482#
MASK    004630R          627*   631     633     637     639*   1401#
MD    = ******  GX         712
MDATAB  000272R          576#
MDMTAB  000312R          603#
MDROP   011450R         1504   1558#
MODNAM  000000R          446#
MODSP   000164R          457    480#
MSG   = ******  G        482#  1054
MSGBF0  005112R         1407   1500#
MSGBF1  007132R         1408   1501#
MSGCNT  004626R          655*  1362*   1363    1365*   1400#
MSGHED= ******  G        482#   754     800     846
MSGN$ = 104411           482#   787    1391
MSGSIZ= ******  G        482#   746     755     801     815     841     848    1073
MSGS$ = 104416           482#
MSG$  = 104406           482#
MSUM    004622R         1391   1397#
NXTLIN  004632R          692*   694     696     699*   1402#
OACNV = ******  G          1#
ONEREC  001150R          493    752#    783
ONETMT  001472R          494    810#
OPEN  = 000000           447    453     460     461     462     463     464     465     466     467     469     471     472
                         473    482#
PASCNT  000026R          458#
PASTIM  005000R          690*   726*   1330    1453#
```



```
PHYTAB  000212R          511#   628     635     653     692     694     719     722    1366    1389
PIRQ$ = 000004           482#   979     991    1223    1227    1278    1282
POPSP = 005726           482#
POPSP2= 022626           482#
PP      001052R          713    735#
PPFD    001604R          738    836#
PPFD1   001624R          840#   879
PPHDM   001406R          741    799#    822
PPHDM2  001534R          813    820#    831
PPHDS   001104R          744#   766
PPHDS2  001240R          753    764#    784
PRTY  = 000000           482#
PRTY0 = 000000           482#
PRTY1 = 000040           482#
PRTY2 = 000100           482#
PRTY3 = 000140           482#
PRTY4 = 000200           482#
PRTY5 = 000240           450    451     482#
PRTY6 = 000300           482#
PRTY7 = 000340           482#
PS    = 177776           482#
PSW   = 177776           482#
PUSH  = 005746           482#
PUSH2 = 024646           482#
PWRCNT= ******  G        482#   670
QUE$  = 104401           482#
RCTOR4  005016R          681#   682*   1125*   1129    1134*   1460#
RCVTO   004772R          710*  1039*   1114*   1115    1135*   1450#
RDTO    003260R          897    919    1113#
REC     002134R          750    818     843     906#
RECADR  004650R          745*   814*    840*    963*    964*   1409#
RECCNT  004654R          746*   747*    815*    816*    841*    842*    965*   1411#
RECD    004702R          972   1422#
RECDN   002704R          979   1035#   1422    1423
RECERR  003206R         1059   1096#
RECINT  002244R          908    927#
RECRTN  004716R          896*   906*    916    1078*   1080    1136    1428#
RECTIM  004660R          748*   749*    817*    920    1413#
REPORT  000164R          486#   954    1056    1063    1083    1093    1360    1363
RERR    002502R          957    962     982#
RESTRT  000532R          474    657     670#
RETTAB  005072R         1040*  1079*   1217*   1311*   1325    1328    1490#
RPIRQ   002474R          972*   977#
RRET    003134R         1031   1079#   1085    1090    1109
RSTRT   000062R          474#
RTSSET  004176R         1271   1273#   1288
R4TAB   005102R         1310*  1494#
SRADR   000052R          468#
SPARE1  000202R          496#
SPARE2  000204R          497#
SPARE3  000206R          498#
SPARE4  000210R          499#
SPOINT  000024R          457#   674
SPSIZ = 000040             1#   475
SP1     000016R          453#   640
START   000332R          456    622#
```

```
STAT    000020R          455#
SUM     011162R         1397    1507#
SUM1    011246R         1367    1517#
SVR0    000032R          460#
SVR1    000034R          461#
SVR2    000036R          462#
SVR3    000040R          463#
SVR4    000042R          464#
SVR5    000044R          465#
SVR6    000046R          466#
SYNC    000174R          492#    647*    648     946    1167    1177    1207
SYNCNT  004664R          912*    942     950*   1415#
TIDONE  003736R         1195    1215#
TIEXIT  003664R         1190    1199#   1208    1212
TIMCHK  004332R          718    1316    1325#   1342
TIME  = ******  G        482#   1309    1345
TIMELN  000170R          490#    748     749     760     806     817     856     857
TIMKEP  004256R          859     923    1178    1307#
TIMTAB  005062R         1308*   1309*   1486#
TMT     003426R          761     807     852    1152#
TMTIDON 004752R          861*    865     885*   1153*   1442#
TMTINT  003576R         1156    1181#
TMTRET  004746R         1152*   1168    1218*   1230    1232    1257    1440#
TMTTIM  004742R          760*    806*   1174    1438#
TPX   = 000000           482#
TRAPX = 000020           482#
TSYNCT  004756R          953*   1160*   1165*   1202    1206*   1444#
TXADR1  004722R          754*    800*    846*   1209    1210*   1430#
TXADR2  004726R          758*    804*    851*   1196    1197*   1432#
TXCNT1  004732R          755*    756*    801*    802*    848*    849*   1192    1211*   1434#
TXCNT2  004736R          757*    803*    850*   1194    1198*   1436#
TXTO    004022R         1173    1237#
TXTOR4  005022R          683*    684*   1247*   1251    1256*   1463#
VCTSAV  004640R          707*    907    1154    1266    1405#
VECTOR  000010R          449#    705
WAIT  = ******  G        482#    656
WASADR  000054R          470#
XFLAG   000005R          447#
X4      003144R         1072    1083#
X5      003154R         1057    1084    1087#   1094
.     = 011471R         1500#   1501#   1556#


. ABS.  000000     000
        011471     001


% ERRORS DETECTED:  1
% DEFAULT GLOBALS GENERATED:  1

 *NUAA,NUAA/SOL/CRF:SYM=DDXCOM,NUAA
 RUN-TIME:  7 12 1 SECONDS
 RUN-TIME RATIO: 53/20=2.5
 CORE USED:  8K  (15 PAGES)
```

```
                .REM %

                                        IDENTIFICATION
                                        --------------

                                PRODUCT CODE:   MAINDEC-11-DXNQA-A-D

                                PRODUCT NAME:   DQ-11 DECNET DEC/X11 EXERCISER MODULE

                                DATE:   21-JUN-76

                                MAINTAINER:     DEC/X11 DIAGNOSTIC GROUP
```

## 1. ABSTRACT

THIS TYPE OF DEC/X11 MODULE, REFERRED TO AS DECNET DEC/X11 MODULE, WILL DETECT SYSTEM INTERACTION ERRORS WHEN RUN IN CONJUNCTION WITH OTHER DEC/X11 MODULES EXERCISING OTHER DEVICES ON THE SYSTEM. ERRORS ARE REPORTED ON THE SYSTEM CONSOLE AS THEY OCCUR OR AS A SUMMARY REPORT, AT THE OPERATORS OPTION. IN ADDITION, THESE MODULES MAY BE USED TO VERIFY THE CONDITION OF THE ASSOCIATED MODEMS AND COMMUNICATION LINKS. IT WILL ONLY TEST COMMUNICATIONS PATHS CAPABLE OF RUNNING DDCMP. THIS MEANS THE DEVICES MUST USE NO PARITY, 8 LEVEL CODE, AND TERMINALS CANNOT BE TESTED.

NOTE

THIS DOCUMENT ASSUMES FAMILIARITY WITH THE PHILOSOPHY AND OPERATION OF DEC/X11. THE DETAILS OF LINKING, LOADING, AND RUNNING MODULES UNDER DEC/X11 CAN BE FOUND IN THE DEC/X11 USERS DOCUMENTATION AND REFERENCE GUIDE, MAINDEC-11-DXQBA. THE READER SHOULD ALSO HAVE READ THE DECNET DEC/X11 USER'S GUIDE, MAINDEC-11-DXQBC, WITHOUT THROUGHLY UNDERSTANDING THAT DOCUMENT, THERE IS LITTLE HOPE OF SUCCESSFUL TESTING.

## 2. HARDWARE REQUIREMENTS

THESE MODULES REQUIRE A COMPLETE PATH FROM A COMMUNICATIONS DEVICE TO A COMMUNICATIONS DEVICE. THIS MAY BE ACCOMPLISHED THROUGH A VARIETY OF METHODS:

A. 1 PROCESSOR, 1 COMM. DEVICE, AND A LOOP BACK DEVICE WITH MODEM SIMULATION CAPABILITIES. (MUST RUN FULL DUPLEX)

B. 1 PROCESSOR, 2 COMM. DEVICES, 2 MODEMS

C. 2 PROCESSORS, 2 COMM. DEVICES, 2 MODEMS

UP TO SIXTEEN LINES CAN BE HANDLED BY ONE MODULE BUT ONLY 2 LINES ARE KEPT RUNNING SIMULTANEOUSLY.

## 3. SOFTWARE REQUIREMENTS

A. THE DEC/X11 MONITOR MUST BE QABK OR A LATER REVISION.

B. THE N1A? MODULE MUST BE CONFIGURED DIRECTLY AFTER THE MONITOR*

C. ALL NETWORK MODULES MUST BE CONFIGURED NEXT.

D. THE N2A? MODULE MUST BE CONFIGURED NEXT.*

E. ONE OF THE 3 CLOCK MODULES MUST BE CONFIGURED NEXT. EITHER KWA?, KWB?, OR KWX?

F. ANY OTHER DESIRED DEC/X11 MODULES CAN THEN BE CONFIGURED IN ANY ORDER.

4. PRELIMINARY TESTING

THE APPROPRIATE STAND ALONE DIAGNOSTICS MUST BE RUN TO VERIFY THE CPU, AND THE COMMUNICATIONS DEVICE. THEN A DEC/X11 EXERCISER USING THE CONVENTIONAL (MAINTENANCE MODE) DEC/X11 MODULES FOR THE COMMUNICATIONS DEVICES MUST BE RUN. IF BOTH THESE STEPS PASS SUCCESSFULLY, DECNET DEC/X11 MAY BE ATTEMPTED.

5. PROGRAMMING CONSIDERATIONS

A. OTHER THAN THE REQUIREMENTS MENTIONED IN 3 REGARDING THE ORDER IN WHICH MODULES MUST BE REQUIRED, THESE DECNET DEC/X11 MODULES WILL BE COMPLETELY COMPATIBLE WITH THE DEC/X11 MONITORS AND ALL OTHER MODULES. THE ONLY OTHER EXECPTION IS THAT FOR A GIVEN COMMUNICATIONS DEVICE, THERE WILL NOW BE 2 DIFFERENT DEC/X11 MODULES CAPABLE OF BEING RUN, THE OLDER MAINTANCE MODE TYPE REQUIRING NO CONNECTIONS TO THE COMMUNICATION DEVICE, AND THE NEW DECNET TYPE REQUIRING A COMPLETED COMMUNICATIONS PATH. IT WILL BE POSSIBLE (AND SOMETIMES DESIREABLE) TO CONFIGURE 2 DIFFERENT DEC/X11 MODULES IN ONE RUN TIME EXERCISER WHICH TEST THE SAME PHYSICAL DEVICE. IF THIS IS DONE, ONLY ONE OF THE 2 MODULES MAY BE SELECTED FOR THE SAME RUN.

B. USING THESE DECNET DEC/X11 MODULES, IT IS POSSIBLE TO HAVE MANY NODES CONNECTED IN A NETWORK, ALL RUNNING DEC/X11 TOGETHER. ALL THESE EXERCISERS WOULD HAVE BEEN STARTED AT DIFFFERENT TIMES. DUE TO DIFFERENT HARDWARE ON EACH SYSTEM, RUN TIMES AND END OF PASS TIMES MIGHT VARY CONSIDERABLY. THEREFORE IT IS NECCESSARY THAT THESE DECNET DEC/X11 MODULES BE EXTREMELY PATIENT AND HAVE A HIGH RETRY CAPABILITY IN ALL OPERATIONS. THIS HOWEVER, MEANS IT SOMETIMES MAY TAKE A LONG TIME (ON THE ORDER OF 10 MINUTES) BEFORE AN OPERATOR FINDS OUT A LINE IS DEFINITELY NOT RUNNING.

C. DUE TO THE EXCESSIVE PROGRAMMING COMPLICATIONS AND LARGE

---

* THESE ARE DESCRIBED IN THE DECNET DEC/X11 USER'S GUIDE.

BUFFERS REQUIRED TO RUN MANY LINES FROM ONE MODULE ALL AT THE SAME TIME, IT WAS DECIDED TO ONLY RUN 2 AT ANY ONE TIME. IF THE MODULE IS SET TO RUN 16 LINES, IT WILL RUN 1 PAIR FOR THE NUMBER OF TRANSFERS INDICATED BY LOCATION LPCNT, THEN IT WILL RUN A 2ND PAIR, THEN 3RD, ETC UP TO THE 8TH PAIR. IT WILL THEN REPORT AN END OF PASS AND GO BACK TO THE FIRST PAIR, THEN 2ND, ETC. THIS FORCES SOME CONSIDERATIONS TO BE MADE WHEN CONNECTING LINES TOGETHER SO THAT A LINE IN THE ACTIVE MODE IS NOT WAITING ON A LINE IN THE DORMANT MODE. PLEASE READ THE DECNET DEC/X11 USER'S GUIDE, ESPECIALLY THE SECTION ON 2 LINES AT A TIME CONSIDERATIONS.

D. THESE MODULES WILL, AS A GENERAL RULE, USE THE MINIMUM AMOUNT OF FEATURES OF A COMMUNICATION DEVICE REQUIRED TO SEND AND RECEIVE MESSAGES. THIS MEANS THAT CRC CALCULATING HARDWARE, CHARACTER RECOGNITION HARDWARE, ETC, WILL NOT BE USED. THESE FEATURES ARE FULLY TESTED BY STAND ALONE DIAGNOSTICS.

## 6. OPERATING PROCEDURES

A. THE MODULE IS CONFIGURED, LOADED, AND RUN AS SPECIFIED IN THE DEC/X11 REFERENCE GUIDE.

B. CONSOLE SWITCHES ARE STANDARD FOR DEC/X11

C. THE N1A? MODULE WILL SOLICIT FROM THE OPERATOR WHICH TYPE OF MESSAGE HE WANTS TO USE FOR DECNET ACTIVITY. THE OPTIONS INCLUDE:

- ALL ONES
- ALL ZEROS
- ALTERNATING ONES AND ZEROS
- A DIGIT (4 BIT) REPEATING COUNT PATTERN
- A PRECANNED ASCII MESSAGE
- USER SPECIFIED BY TYPING IN

D. THE N2A? MODULE WILL MAKE ANY AUTOMATIC PHONE CONNECTIONS USING ANY DN-11'S PRESENT AND WILL REQUEST THE OPERATOR TO DIAL ALL MANUAL CONNECTIONS. ALL OPERATORS AT EACH NODE UNDER TEST MUST REACH THIS POINT BEFORE ANY ONE NODE CONTINUES INTO THE EXERCISING PHASE.

E. THE N2A? MODULE WILL REQUEST A CARRIAGE RETURN WHEN THE OPERATOR WANTS TO BEGIN EXERCISING HIS NODE.

F. THERE ARE MANY OPTIONS WHICH MAY BE ALTERED BEFORE RUN TIME, THEY INCLUDE:

(1) LOCATION 164 IS A SWITCH WHICH INDICATES WHETHER OR NOT ALL ERRORS SHOULD BE REPORTED AS THEY HAPPEN. IF LOCATION 164 IS NON-ZERO (EQUAL TO N), THEN ERRORS ARE TOTALLED (ON A LINE BY LINE BASIS); EVERY N PASSES THIS

TOTAL NUMBER OF ERRORS FOR THE PREVIOUS N PASSES WILL BE REPORTED. THE VALUE OF N IS THE VALUE OF LOCATION 164. THE MAXIMUM NUMBER OF ERRORS THAT CAN BE TALLEYED IS 256 PER LINE. THEREFORE IF A LINE IS NOT KNOWN TO BE OF VERY GOOD QUALITY, THE VALUE OF LOCATION 164 TIMES THE VALUE OF LOCATION 166 SHOULD BE LESS THAN 128. (LOCATION 166 IS DEFINED NEXT.) THIS MEANS THE NUMBER OF MESSAGES PER PASS TIMES THE NUMBER OF PASSES PER SUMMARY REPORT SHOULD BE LESS THAN 128. IN THIS WAY, IF EVERY MESSAGE ATTEMPTED HAS AN ERROR ON BOTH TRANSMIT AND RECEIVE, THE ERROR TALLY WILL NOT OVERFLOW. IF THE LINE IS OF KNOWN GOOD QUALITY, THIS RESTRICTION MAY BE RELAXED. THE DEFAULT VALUE IS TO NOT REPORT TOTAL ERRORS.(LOC 164=0) OF COURSE DEC/X11 KEEPS TOTALS FOR THE ENTIRE RUN, REGARDLESS.

(2) LOCATION 166 INDICATES HOW MANY ITERATIONS SHOULD BE DONE BEFORE AN END OF PASS IS CALLED. AN ITERATION IS NORMALLY ONE TRANSMIT AND ONE RECEIVE, IN EITHER ORDER. THE DEFAULT VALUE IS 8.

(3) LOCATION 170 IS A TIME SCALING FACTOR. THERE ARE MANY TIME OUT LOOPS IN THE MODULES, SOME SHOULD LOGICALLY BE LONGER THAN OTHERS. THE VALUE IN LOCATION 170 IS THE NUMBER OF SECONDS TO WAIT FOR THE SMALLEST TIME OUTS. SOME TIME OUTS WILL BE A MULTIPLE OF THIS FACTOR. DEFAULT IS 30 SECONDS.

(4) LOCATION 172 IS AN ERROR TOLERATION LEVEL, ON A SINGLE MESSAGE BASIS. SO IF IN AN ATTEMPT TO DO ONE ITERATION (USUALLY A TRANSMIT THEN RECIEVE OR VICE-VERSA) THE NUMBER OF ERRORS REACHES THE VALUE IN LOCATION 172, THAT LINE IS DROPPED FROM TESTING. IF BEFORE THIS LEVEL IS REACHED, AN ITERATION IS CORRECTLY COMPLETED, THEN THE ERROR COUNT WHICH IS COMPARED TO LOCATION 172 IS RESET TO ZERO. THE DEFAULT IS 10.

(5) LOCATION 174 CONTAINS 2 IDENTICAL BYTES FOR USE AS SYNC CHARACTERS. THE DEFAULT IS 113226 WHICH IS 2 BYTES OF 226. IF THIS IS CHANGED, BOTH BYTES MUST BE IDENTICAL TO EACH OTHER. THIS LOCATION IS A DON'T CARE FOR ASYNCHRONOUS DEVICES, BECAUSE THEY "SYNC" ON THE FIRST BYTE OF THE ACTUAL MESSAGE, NOT ON PRECEDING CHARACTERS AS SYNCHRONOUS DEVICES DO.

(6) LOCATION 176 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN RECEIVE-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, SLAVE. (BIT 7 = 1, BITS 4,5, AND 6 = 0'S IN THE PHYSICAL LINE TABLE. SEE ITEM 6-F-9 BELOW). THEN PATCH THE LOCATION POINTED TO BY LOCATION 176 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, SLAVE BY THIS MODULE TO RUN IN RECEIVE ONLY MODE. THIS LOCATION MAY BE PATCHED BACK TO A 401 TO RESUME NORMAL REC-TMT TESTING.

(7) LOCATION 200 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN TRANSMIT-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, MASTER (BITS 7 AND 5 = 1'S, BITS 4 AND 6 = 0'S IN THE PHYSICAL-LINE TABLE. SEE ITEM 6-F-9 BELOW.) THEN PATCH THE LOCATION POINTED TO BY LOCATION 200 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, MASTER BY THIS MODULE TO RUN IN TRANSMIT ONLY MODE. THE LOCATION MAY BE PATCHED BACK TO 401 TO RESUME NORMAL TMT-REC TESTING.

G. LOCATIONS DVA, VCT, BR1 AND BR2 MUST BE SET TO PROPERLY MATCH THE FIRST DEVICE TO BE TESTED BY THIS MODULE.

7. ERROR REPORTING

A. ERRORS ARE EITHER TOTALLED AND PRINTED AS SUMMARIES ON A LINE BY LINE BASIS, OR THEY ARE PRINTED AS THEY OCCUR. SEE SECTION 6F1. THEY ARE 2 CLASSES OF ERRORS DATA ERRORS AND OTHER ERRORS.

B. DATA ERRORS ARE ACTUALLY REPORTED BY THE N1A? MODULE BECAUSE THIS IS WHERE THE SHARED DATA COMPARE ERROR ROUTINE IS LOCATED. SO ALL DECNET DEC/X11 MODULE DATA COMPARE ERRORS LOOK LIKE THE STANDARD DEC/X11 DATA COMPARE ERROR MESSAGE EXCEPT:

(1) THE MODULE NAME TYPED IS FFXX, WHERE FF IS FILLED IN TO BE THE NAME OF THE DECNET DEC/X11 MODULE WHICH CALLED THE N1A? MODULE TO DO THE COMPARE. IN OTHER WORDS FF REPRESENTS THE MODULE THAT "HAD" THE ERROR.

(2) THE ERROR NUMBER IS A COUNT OF THE DATA COMPARE ERRORS IN THE CURRENT MESSAGE BEING CHECKED, IT IS THEREFORE RESET TO ZERO EVERY TIME A NEW MESSAGE IS TO BE CHECKED.

(3) ACSR CONTAINS THE ADDRESS OF THE DECNET DEC/X11 MODULE WHICH CALLED THIS ROUTINE TO HAVE ITS MESSAGE CHECKED.

(4) SBADR CONTAINS THE LINE # OF THE DEVICE WHICH HAD THE DATA ERROR.

(5) WASADR CONTAINS THE COUNT (IN OCTAL) OF WHICH CHARACTER IN THE MESSAGE THIS BAD CHARACTER WAS.

(6) THE "SHOULD BE AND "WAS" ITEMS ARE NORMAL.

C. OTHER ERRORS ARE REPORTED USING THE NORMAL ERROR CALL IN

DEC/X11 CSRA AND ACSR ARE STANDARD. STATC CONTAINS A CODED

ERROR NUMBER, WITH THE CODE BEING CONSISTENT IN ALL DECNET DEC/X11 MODULES. THE LOW ORDER 2 DIGITS OF STATC CONTAIN WHICH LINE HAD THE ERROR THE OTHER DIGITS CONTAIN THIS CODE:

| | |
|---|---|
| 0010XX | REC TIME OUT |
| 0020XX | TMT CLOCK LOSS |
| 0030XX | HEADER CRC ERROR |
| 0040XX | MSG CRC ERROR |
| 0050XX | TMT TIME OUT |
| 0060XX | TMT NON-EXISTENT MEMORY |
| 0070XX | TMT LATENCY |
| 0100XX | REC DATA PARITY ERROR |
| 0110XX | REC NON-EXISTENT MEMORY |
| 0120XX | REC CLOCK LOSS |
| 0200XX | REC FRAMING ERROR |
| 0400XX | REC OVERRUN OR LATENCY ERROR |

THIS METHOD SHOULD ELIMINATE THE CONSTANT NEED FOR HAVING A LISTING ON HAND TO DECIDE WHAT THE ERRORS WHERE. THE FIRST TIME THE RUN TIME EXERCISER IS RUN AFTER LOADING, THE N1A? MODULE WILL TYPE THIS ERROR CODE LIST OUT TO THE SYSTEM CONSOLE FOR THE OPERATORS SUBSEQUENT USE.

D. ALL ERRORS HAVE A RETRY LIMIT (SEE SECTION 6F4) WHICH CAN BE MODIFIED.

E. ON POWER-FAIL, POWER-UP CONDITIONS, DECNET DEC/X11 MODULES WILL DROP THEMSELVES. THIS IS DONE BECAUSE THE PHONE CONNECTIONS WILL HAVE BEEN LOST AND OPERATOR INTERVENTION WILL BE REQUIRED.

8. TEST SEQUENCE

LINES ARE TESTED BY PAIRS IN SEQUENTIAL ORDER. THIS ORDER MAY BE MODIFIED BY MODIFYING THE PHYSICAL LINE TABLE. (SEE DECNET DEC/X11 USER'R GUIDE). THERE ARE SEVERAL TESTING OPTIONS WITH SLIGHTLY DIFFERENT FUNCTIONS MAKING UP AN ITERATION:

A. POINT-TO-POINT SLAVE. IN THIS CASE THE LINE WAITS TO RECEIVE A MESSAGE, WHEN IT DOES, IT CHECKS IT, AND THEN TRANSMITS A MESSAGE BACK. IT WILL REPORT ERRORS ON EXCESSIVE WAITS.

B. POINT-TO-POINT MASTER. IN THIS CASE THE LINE SENDS A MESSAGE AND WAITS TO RECEIVE ONE IN RETURN. IT THEN CHECKS THE RECEIVED MESSAGE. IF IT TIMES-OUT, IT TRANSMITS AGAIN, IT WILL CONTINUE TO TIME OUT AND RE-TRANSMIT UNTIL EITHER A MESSAGE IS RECEIVED, OR IT TIMES OUT COUNT EQUALS LOCATION

172, THE ERROR TOLERATION LEVEL. (SEE SECTION 6F4). IT WILL THEN REPORT AN ERROR AND BEGIN AGAIN. AFTER THE NUMBER OF

CONSECUTIVE TIME-OUT ERRORS REPORTED EQUALS LOCATION 172, THE LINE WILL BE DROPPED.

C. POINT-TO-POINT SLAVE RECEIVE ONLY. IN THIS MODE THE LINE WAITS ONLY TO RECEIVE, CHECKS THE MESSAGE, AND WAITS TO RECEIVE AGAIN.

D. POINT-TO-POINT MASTER-TRANSMIT ONLY. IN THIS MODE THE LINE JUST TRANSMITS AND THEN TRANSMITS AGAIN ,ETC.

E. MULTI-DROP MASTER. IN THIS MODE, THE LINE WILL SEND A DDCMP MESSAGE OUT FOR THE ADDRESS CONTAINED IN THE MULTI-DROP MENU TABLE, WAIT TO RECEIVE A MESSAGE BACK, AND THEN CHECK THE DATA.
* NOT YET IMPLEMENTED

F. MULTI-DROP SLAVE. IN THIS MODE, THE LINE WILL WAIT FOR A MESSAGE WITH THE MATCHING ADDRESS. THEN IT WILL TRANSMIT A MESSAGE.
* NOT YET IMPLEMENTED

G. THESE BASIC ITERATIONS ARE REPEATED UP TO THE NUMBER OF TIMES IN LOCATION 166. (SEE SECTION 6F2) FOR EACH LINE SELECTED FOR TEST, AND THE END-OF-PASS IS DECLARED.

9. PASS TIMES

PASS TIMES ARE EXTREMELY CONFIGURATION DEPENDENT. DIFFERENT BAUD RATES, MESSAGE SIZES, AND TESTING MODES ALL GREATLY AFFECT THE PASS TIME. BELOW ARE SOME BALL PARK TIMES, BUT ANY ONE SYSTEM COULD VARY TREMENDOUSLY FROM THESE:
-RUNNING ALONE, 1 LINE ON AN 11/05 AT 9600 BAUD,HALF DUPLEX, POINT-TO-POINT, WITH A 512 CHARACTER MESSAGE, TAKES 15 SECONDS.

NOTE 1 AT VARIOUS MONITOR CALLS THROUGHOUT THIS MODULE, IT IS LIKELY THAT WHEN 2 LINES ARE RUNNING, A MONITOR CALL CAN BE MADE AND BEFORE CONTORL IS RETURNED TO THE MODULE, THE OTHER LINE MAY CAUSE THE EXACT SAME CALL TO BE MADE. IN FACT, WHEN RUNNING 2 LINES FULL DUPLEX, IT MAY BE POSSIBLE TO HAVE 4 CALLS TO THE MONITOR FROM ONE SPOT IN THE MODULE, BEFORE THE FIRST CALL IS RETURNED FROM. THIS IS WHY AT SUCH CALLS, THE CODE SETS COUNTERS BEFORE DOING CALLS, SO THAT WHEN IT IS RETURNED TO, IT CAN KEEP WHICH LINE DID WHAT STRAIGHT.

```
                        NOTE 2  WHEN RUNNING 2 DQ'S OUT OF THE SAME MODULE WHICH ARE CONNECTED
                        TO EACH OTHER, THE LOW ORDER (PER ORDER OF DVC, LOCATION 14) ONE
                        SHOULD BE SLAVE, THE HIGHER ORDER ONE SHOULD BE MASTER.  FURTHERMORE,
                        IF DOING THIS IN FULL DUPLEX, THE TIMEOUT CONSTANT (LOCATION 170)
                        SHOULD BE MOD'ED FROM 36 TO 5 SECONDS.(IN OCTAL)
                        %
```

```
                                IOMOD <NQAA >, 1,4,5,5
                                MODULE  140000,NQAA ,1,4,5,5,
                                .TITLE  NQAA DEC/X11 SYSTEM EXERCISER MODULE
                        ;       DDXCOM  VERSION  4    9/6/75
                                .LIST   BIN
                        ;*****************************************************************
000000'                 BEGIN:
000000' 050516 040501 040       MODNAM: .ASCII  /NQAA / ;MODULE NAME.
000005'    000                  XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000001                  ADDR:   1+0             ;1ST DEVICE ADDR.
000010' 000004                  VECTOR: 4+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' 000332'                 INIT:   START                   ;MODULE START ADDR.
000024' 000164'                 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' 000003                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000056' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000060' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000062' 000662'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
                                .REPT   SPSIZ                   ;MODULE STACK STARTS HERE.
                                .NLIST
                                .WORD   0
                                .LIST
                                .ENDR
000164'                         MODSP:
                        ;*****************************************************************
                                .GLOBL  WAIT, TIME, MSGSIZ, CKMSG, CRCMSG,MSGHED, MSG,PWRCNT
                                .GLOBL  DEVSET
                        ;
                        ;
000164' 000000          REPORT: 0               ;FLAG, IF =#, THEN ERRORS ON RECEIVED MSG'S
                                                ;ARE TOTALLED AND PRINTED EVERY # PASSES
                                                ;IF =0, ERRORS ARE ALL REPORTED AS THEY HAPPEN.
000166' 000010          LPCNT:  10              ;ITERATIONS PER END PASS CALL
000170' 000036          TIMELN: 30.             ;TIME OUT FACTOR, IN SECONDS
000172' 000012          ERRLVL: 10.             ;NUMBER OF ERRORS TOLERATED
000174' 113226          SYNC:   113226          ;2 BYTES OF SYNC CHARACTER
000176' 001310'                 ONEREC          ;ADDRESS OF WHERE 240 GOES
000200' 001606'                 ONETNT          ;ADDRESS OF WHERE 240 GOES
```



```
000202' 000000          SPARE1: 0
000204' 000000          SPARE2: 0
000206' 000000          SPARE3: 0
000210' 000000          SPARE4: 0
                        ;
                        ;
                        ;PHYSICAL LINE TABLE. BYTE 0 CONTAINS THE PHYSICAL
                        ;LINE # FOR THE FIRST LINE TO BE RUN. BYTE 1 WILL BE
                        ;THE PHYSICAL LINE RUN SECOND, ETC.
                        ;EACH BYTE IN THE TABLE HAS THE FOLLOWING BIT MEANINGS:
                        ;BIT 0-3        PHYSICAL LINE #
                        ;BIT 4          0 FOR POINT-TO POINT, 1 FOR MULTI-DROP
                        ;BIT 5          0 FOR SLAVE, 1 FOR MASTER (DERIVED FROM SR1)
                        ;BIT 6          0 FOR HALF DUPLEX, 1 FOR FULL
                        ;BIT 7          0 DO NOT TEST, 1 TEST (DERIVED FROM DVID1)
000212'    000          PHYTAB: .BYTE   0
000213'    001                  .BYTE   1
000214'    002                  .BYTE   2
000215'    003                  .BYTE   3
000216'    004                  .BYTE   4
000217'    005                  .BYTE   5
000220'    006                  .BYTE   6
000221'    007                  .BYTE   7
000222'    010                  .BYTE   10
000223'    011                  .BYTE   11
000224'    012                  .BYTE   12
000225'    013                  .BYTE   13
000226'    014                  .BYTE   14
000227'    015                  .BYTE   15
000230'    016                  .BYTE   16
000231'    017                  .BYTE   17

                        ;
                        ;
                        ;ERRTAB ,KEEPS TRACK OF RETRY ERROR COUNTS FOR
                        ;EACH LINE
                        ;
000232'    000          ERRTAB: .BYTE   0
000233'    000                  .BYTE   0
000234'    000                  .BYTE   0
000235'    000                  .BYTE   0
000236'    000                  .BYTE   0
000237'    000                  .BYTE   0
000240'    000                  .BYTE   0
000241'    000                  .BYTE   0
000242'    000                  .BYTE   0
000243'    000                  .BYTE   0
000244'    000                  .BYTE   0
000245'    000                  .BYTE   0
000246'    000                  .BYTE   0
000247'    000                  .BYTE   0
000250'    000                  .BYTE   0
000251'    000                  .BYTE   0
                        ;
                        ;
```

```
                                ;ECOUNT TABLE KEEPS TRACK OF # OF BAD MSG'S
000252' 000                     ECOUNT: .BYTE   0
000253' 000                             .BYTE   0
000254' 000                             .BYTE   0
000255' 000                             .BYTE   0
000256' 000                             .BYTE   0
000257' 000                             .BYTE   0
000260' 000                             .BYTE   0
000261' 000                             .BYTE   0
000262' 000                             .BYTE   0
000263' 000                             .BYTE   0
000264' 000                             .BYTE   0
000265' 000                             .BYTE   0
000266' 000                             .BYTE   0
000267' 000                             .BYTE   0
000270' 000                             .BYTE   0
000271' 000                             .BYTE   0
                                ;
                                ;
                                ;MULTI-DROP STATION ADDRESS TABLE
                                ;BYTE 0 IS THE MULTI-DROP STATION ADDRESS
                                ;FOR LOGICAL LINE 0, BYTE 1 IS FOR LOGICAL LINE 1, ETC.
                                ;THIS ENTRY ONLY HAS MEANING IF CORRESPONDING
                                ;BYTE IN PHYSICAL LINE TABLE HAS BITS 4 AND 7 SET
                                ;AND BIT 5 OFF, IE MULTI-DROP SLAVE TO BE TESTED
000272' 000                     MDATAB: .BYTE   0
000273' 001                             .BYTE   1
000274' 002                             .BYTE   2
000275' 003                             .BYTE   3
000276' 004                             .BYTE   4
000277' 005                             .BYTE   5
000300' 006                             .BYTE   6
000301' 007                             .BYTE   7
000302' 010                             .BYTE   10
000303' 011                             .BYTE   11
000304' 012                             .BYTE   12
000305' 013                             .BYTE   13
000306' 014                             .BYTE   14
000307' 015                             .BYTE   15
000310' 016                             .BYTE   16
000311' 017                             .BYTE   17
                                ;
                                ;MULTI-DROP STATION ADDRESS MENU TABLE
                                ;THIS TABLE INDICATES TO ANY LINE RUNNING AS
                                ;MULTI-DROP MASTER, WHICH STATION ADDRESS TO
                                ;ATTEMPT TO EXERCISE ON THE MULTI-DROP LINE
                                ;IT ALLOWS UP TO 15 DIFFERENT STATIONS TO BE
                                ;TESTED ON A LINE.  EACH BYTE WHICH IS NOT=-1
                                ;INDICATES A STATION ADDRESS TO BE TESTED, EACH
                                ;STATION IS TESTED, BY THE ORDER OF THE TABLE,
                                ;UNTIL THE FIRST OCCURANCE OF A -1 (BINARY 11111111)
                                ;
000312' 377                     MDMTAB: .BYTE   -1
000313' 377                             .BYTE   -1
000314' 377                             .BYTE   -1
000315' 377                             .BYTE   -1
```



```
000316' 377                             .BYTE   -1
000317' 377                             .BYTE   -1
000320' 377                             .BYTE   -1
000321' 377                             .BYTE   -1
000322' 377                             .BYTE   -1
000323' 377                             .BYTE   -1
000324' 377                             .BYTE   -1
000325' 377                             .BYTE   -1
000326' 377                             .BYTE   -1
000327' 377                             .BYTE   -1
000330' 377                             .BYTE   -1
000331' 377                             .BYTE   -1
                                ;
                                ;
                                ;
000332' 016700 177456           START:  MOV     DVID1,R0        ;GET DEVICES TO BE TESTED
000336' 001002                          BNE     1$              ;ANY ACTIVE?
000340' 104403 000000'                  END$,BEGIN              ;
000344' 012705 000000'          1$:     MOV     #BEGIN,R5       ;PREPARE FOR NXT JSR
000350' 004567 000000G                  JSR     R5,DEVSET       ;GO ASK FOR PARAMETERS
000354' 012767 000200  004364           MOV     #200,MASK       ;SET UP MASK
000362' 012701 000232'          2$:     MOV     #PHYTAB+16.,R1  ;GET END OF DEVICE TABLE
000366' 006300                  3$:     ASL     R0              ;CHECK EACH BIT
000370' 103003                          BCC     4$              ;BR IF NOT ON
000372' 156741 004350                   BISB    MASK,-(R1)      ;SET THIS LINE ACTIVE
000376' 000402                          BR      5$
000400' 146741 004342           4$:     BICB    MASK,-(R1)      ;CLEAR THIS LINE
000404' 105061 000040           5$:     CLRB    32.(R1)         ;CLEAR ECOUNT TABLE
000410' 020127 000211'                  CMP     R1,#PHYTAB-1    ;DONE?
000414' 001364                          BNE     3$              ;BR IF NOT
000416' 026727 004324  000040           CMP     MASK,#40        ;BEEN HERE BEFORE?
000424' 001406                          BEQ     6$              ;BR IF YES
000426' 012767 000040  004312           MOV     #40,MASK        ;SET UP MASK
000434' 016700 177356                   MOV     SR1,R0          ;GET SLAVE/MASTER INFO
000440' 000750                          BR      2$              ;GO SET UP
000442' 005201                  6$:     INC     R1              ;GET PHYTAB ADDR
000444' 016700 177336                   MOV     ADDR,R0         ;GET DEVICE ADDR
000450' 105721                  7$:     TSTB    (R1)+           ;DEVICE ACTIVE?
000452' 100054                          BPL     10$             ;BR IF NOT
000454' 012760 001000  000002           MOV     #1000,2(R0)     ;SET DATA TERM READY
000462' 005060 000004                   CLR     4(R0)           ;CLR ERR REGISTER
000466' 012702 000017                   MOV     #17,R2  ;SET UP TO CLEAR 16 SEC. REGS
000472' 110260 000005           8$:     MOVB    R2,5(R0)        ;SET SEC. REG. PTR
000476' 052760 010000  000004           BIS     #10000,4(R0)    ;ENABLE EA BIT WRITING
000504' 042760 060000  000004           BIC     #60000,4(R0)    ;CLEAR EA BITS
000512' 012760 000000  000006           MOV     #0,6(R0)        ;CLR  ENTRY
000520' 005302                          DEC     R2              ;POINT TO NXT REG.
000522' 002363                          BGE     8$              ;BR BACK TILL DONE
000524' 112760 000017  000001           MOVB    #17,1(R0)       ;POINT TO CHAR DET REGS
000532' 132760 000017  000001           BITB    #17,1(R0)       ;IS THERE A DQ-BB OPTION?
000540' 001421                          BEQ     10$             ;BR IF NOT (#17 BITS WON'T SET)
000542' 012703 000017                   MOV     #17,R3          ;SET UP A COUNT REG
000546' 112760 000030  000005   9$:     MOVB    #30,5(R0)               ;SELECT SEC REG 10
000554' 005060 000006                   CLR     6(R0)           ;CLEAR AN ENTRY
000560' 112760 000034  000005           MOVB    #34,5(R0)       ;SELECT SEC REG 14
000566' 005060 000006                   CLR     6(R0)           ;CLEAR AN ENTRY
```

```
                                        DEC     R3              ;DONE ENOUGH?
                                        BLT     10$             ;BR OUT IF YES
                                        DECB    1(R0)           ;ELSE POINT TO NEXT PAIR
                                        BR      9$              ;AN GO CLEAR
                        000010          10$:    ADD     #10,R0  ;BUMP R0
                        000232'                 CMP     R1,#PHYTAB+16.  ;DONE?
                                                BNE     7$      ;BR IF NOT
                        004122          11$:    CLR     MSGCNT  ;COUNTS WHEN TO DO SUMMARY
                        000005                  MOV     #5,R1   ;SET SYNC FIELD IN FRONT OF MSG
                        00000@G                 MOV     #MSGHED,R2      ;TO 10 BYTES OF SYNC CHAR'S
                        177336          12$:    MOV     SYNC,-(R2)      ;WRITE 2 SYNC CHAR'S
                                                DEC     R1      ;COUNT IT
                                                BNE     12$     ;BR BACK IF NOT DONE
                        000000G         13$:    TST     WAIT    ;LINES SET UP?
                                                BEQ     RESTRT  ;BR IF YES
                        000000'                 BREAK$,BEGIN    ;TEMPORARY RETURN TO MONITOR,...
                        000000'                 BREAK$,BEGIN    ;THEN CONTINUE AT NEXT INSTRUCTION.
                                                BR      13$     ;WAIT TILL WAIT IS CLEARED BY NET2
                                        ;
                                        ;
                                        ;THROUGH-OUT THE PROGRAM MANY LOCATIONS ARE REFERENCED
                                        ;OFF OF (R4).  THIS IS A MEANS OF KEEPING TWO SETS OF
                                        ;VARIABLES, ONE FOR EACH OF THE ACTIVE LINES.
                                        ;R4 ALWAYS CONTAINS A 0 IF THE PROGRAM IS CURRENTLY
                                        ;REFERENCING LINE 0 OR A 2 IF IT IS REFERENCING
                                        ;LINE 1.
                                        ;
                        000000G         RESTRT: TST     PWRCNT  ;HAS THERE BEEN A POWER FAIL?
                                                BEQ     1$      ;CONTINUE IF NOT,
                        000000'                 END$,BEGIN      ;
                                        1$:     CLR     R4      ;FIRST ACTIVE LINE
                        177122                  MOV     SPOINT,R6       ;INIT STACK
                        004216                  CLR     DROPR4  ;CLEAR ALL EXIT FLAGS
                        004214                  CLR     DROPR4+2
                        004212                  CLR     DTERR4
                        004210                  CLR     DTERR4+2
                        004206                  CLR     CRCR4
                        004204                  CLR     CRCR4+2
                        004202                  CLR     RCTOR4
                        004200                  CLR     RCTOR4+2
                        004176                  CLR     TXTOR4
                        004174                  CLR     TXTOR4+2
                        004172                  CLR     FDBKR4
                        004170                  CLR     FDBKR4+2
                        004166                  CLR     CHKR4
                        004164                  CLR     CHKR4+2

                        004124                  CLR     PASTIM  ;CLR PASS TIME FLAG
                        004116                  CLR     BRKFLG  ;CLR FLAG THAT SAYS WE'RE IN BREAK
                        000212' 003740          MOV     #PHYTAB,NXTLIN  ;POINTS TO NEXT LINE TO TEST
                        003776                  CLR     LINACT  ;SHOW NO LINES ACTIVE
                        000232' 003730  GETNXT: CMP     NXTLIN,#PHYTAB+16.;DONE ALL LINES?
                                                BEQ     DONE    ;BR IF SO
                        003720                  MOV     NXTLIN,R0       ;GET ADDR OF NEXT LINE
                        005020'                 MOV     R0,CURLIN(R4)   ;SAVE IT
                        000020                  CLRB    20(R0)  ;CLEAR RETRY COUNT FOR THIS LINE
```



```
                        003704                  INC     NXTLIN  ;BUMP FOR NEXT TIME
                                                MOVB    (R0),R0 ;GET PHYSICAL LINE BYTE
                                                BGE     GETNXT  ;BR BACK IF BIT 7 NOT ON
                        176732                  MOV     ADDR,R1 ;SET UP TO CALC ADDR
                        004106                  JSR     PC,GETADR       ;GO CALC ADDR
                        004752'                 MOV     R1,DVASAV(R4)   ;SAVE DVA FOR THIS LINE
                        000002                  ADD     #2,R1   ;POINT TO TMT CSR
                        004762'                 MOV     R1,TMTSAV(R4)   ;SAVE VALUE
                        176710                  MOV     VECTOR,R1       ;SET UP TO CALC VECTOR ADDR
                        004062                  JSR     PC,GETADR       ;GO CALC
                        004756'                 MOV     R1,VCTSAV(R4)   ;SAVE IT
                        003676                  INC     LINACT  ;SHOW A LINE'S RUNNING
                        005034'                 CLR     LOOP(R4)        ;CLEAR ITERATION COUNT
                        005114'                 CLR     RCVTO(R4)       ;CLR REC TIME OUT COUNT
                        000020                  BIT     #20,R0  ;MULTI DROP OR NOT?
A                                               BNE     MD      ;BR IF IT IS
                                                BR      PP      ;BR IF NOT TO P TO P
                                        ;
                                        ;
                        003652          DONE:   TST     LINACT  ;ANY LINES RUNNING?
                                                BEQ     1$      ;BR IF NO
                        003302                  JMP     TIMCHK  ;IF YES GO CHECK FOR TIME OUT
                        000212'         1$:     MOV     #PHYTAB,R1      ;MUST CHECK IF MAYBE ALL LINES DROPPED
                                        2$:     TSTB    (R1)+   ;IS THIS LINE STILL SELECTED?
                                                BMI     3$      ;BR OUT IF YES
                                                CMP     #PHYTAB+16.,R1  ;CHECKED ALL 16?
                                                BNE     2$      ;BR BACK IF NO
                                                END$,BEGIN      ;
                                        ;
                        000001  003724  3$:     MOV     #1,PASTIM       ;SHOW IT IS TIME FOR ENDPAS
                                                                ;FOR BREAK LOOP
                                                TST     BRKFLG  ;ARE WE IN A BREAK LOOP?
                                                BNE     4$      ;BR IF WE ARE
                        003354                  JMP     EPASS   ;GO DO END OF PASS
                                        ;
                                        4$:
                                                EXIT$           ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                        ;POINT TO POINT
                        000001  005110' PP:     MOV     #1,FDFLAG(R4)   ;SAY WERE FULL DUPLEX
                        000100                  BIT     #100,R0 ;HALF OR FULL DUPLEX?
                                                BEQ     1$      ;BR IF HALF
                        000466                  JMP     PPFD    ;JMP TO FULL DUPLEX ROUTINE
                        005110'         1$:     CLR     FDFLAG(R4)      ;CORRECT FLAG CAUSE WERE NOT FD
                        000040                  BIT     #40,R0  ;SLAVE OR MASTER?
                                                BNE     PPHDM   ;BR IF MASTER
                                        ;
                                        ;POINT TO POINT, HALF DUPLEX, SLAVE
                                        PPHDS:
                        004766' 004772'         MOV     AMSGBF(R4),RECADR(R4)  ;SET UP INPUT AREA
                        00000@G 004776'         MOV     #MSGSIZ,RECCNT(R4)  ;SET UP COUNT
                        000012  004776'         ADD     #12,RECCNT(R4)  ;CORRECT FOR HEADER AND CRC
                        176676  005002'         MOV     TIMELN,RECTIM(R4)  ;HOW LONG TO WAIT
                        176670  005002'         ADD     TIMELN,RECTIM(R4)       ;DOUBLE IT
                        000726                  JSR     PC,REC  ;GO RECEIVE DATA
                                                BR      FERR    ;BR TO ERROR ROUTINE
                                        ONEREC: BR      1$      ;MAKE THIS A NOP=240 TO
```

```
 775  001312' 000424                          BR      PPHDS2          ;DO ONE-WAY-IN (REC. ONLY)
 776  001314' 012764  000000G 005044' 1$:     MOV     #MSGHED,TXADR1(R4)  ;GET MESS ADR
 777  001322' 016764  000000G 005054'         MOV     MSGSIZ,TXCNT1(R4)  ;GET MSG SIZE
 778  001330' 062764  000012  005054'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC'S
 779  001336' 004767  002700                  JSR     PC,CTS          ;GO GET CLEAR TO SEND
 780  001342' 016764  176622  005064'         MOV     TIMELN,TMTTIM(R4)       ;GIVE 'EM 1/2 MINUTE
 781  001350' 004767  002054                  JSR     PC,TMT          ;GO TRANSMIT
 782  001354' 000425                          BR      DERR            ;BR TO ERROR ROUTINE
 783  001356' 042774  000400  004762'         BIC     #400,@TMTSAV(R4)        ;KILL REQ TO SEND
 784  001364' 005264  005034' PPHDS2: INC     LOOP(R4)        ;COUNT AN ITERATION
 785  001370' 026764  176572  005034'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH!
 786  001376' 003322                          BGT     PPHDS           ;NO, GO DO ANOTHER TIME
 787  001400' 005367  003406          LNDONE: DEC     LINACT
 788  001404' 016403  004752'                 MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
 789  001410' 005013                          CLR     (R3)                    ;KILL RECEIVER SIDE
 790  001412' 042763  000471  000002          BIC     #471,2(R3)              ;KILL TMT SIDE
 791  001420' 005063  000004                  CLR     4(R3)           ;KILL TRANSMITTER TOO
 792  001424' 000167  177364                  JMP     GETNXT          ;YES DO NEXT LINE
 793                                  ;
 794                                  ;
 795  001430' 042774  000400  004762' DERR:   BIC     #400,@TMTSAV(R4)        ;KILL REQ TO SEND
 796  001436' 012703  177777                  MOV     #-1,R3          ;SET FLAG FOR WHICH RETURN
 797  001442' 000401                          BR      FERR1
 798  001444' 005003                  FERR:   CLR     R3              ;SET FLAG FOR WHICH RETURN
 799  001446' 016401  005020'         FERR1:  MOV     CURLIN(R4),R1   ;GET LINE # ADDR
 800  001452' 105261  000020                  INCB    20(R1)          ;COUNT AN ERROR
 801  001456' 126167  000020  176506          CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
 802  001464' 002003                          BGE     DROP            ;BR IF YES
 803  001466' 005703                          TST     R3              ;DO A TMT OR DO A REC?
 804  001470' 001707                          BEQ     ONEREC          ;BR IF TMT
 805  001472' 000734                          BR      PPHDS2          ;GO TRY AGAIN
 806  001474' 142711  000200          DROP:   BICB    #200,(R1)       ;DROP THE LINE FROM TEST
 807  001500' 005264  005124'                 INC     DROPR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
 808  001504' 104411  011306' 000000'         MSGN$,ADROP,BEGIN       ;ASCII MESSAGE CALL WITH COMMON HEADER
 809  001512' 005767  003406                  TST     DROPR4
 810  001516' 001402                          BEQ     1$
 811  001520' 005004                          CLR     R4
 812  001522' 000402                          BR      2$
 813  001524' 012704  000002          1$:     MOV     #2,R4
 814  001530' 005364  005124'         2$:     DEC     DROPR4(R4)
 815  001534' 000721                          BR      LNDONE
 816
 817                                  ;
 818                                  ;POINT TO POINT HALF DUPLEX MASTER
 819                                  ;
 820  001536'                         PPHDM:
 821  001536' 012764  000000G 005044'         MOV     #MSGHED,TXADR1(R4)      ;GET MSG ADDRESS
 822  001544' 016764  000000G 005054'         MOV     MSGSIZ,TXCNT1(R4)       ;GET MSG SIZE
 823  001552' 062764  000012  005054'         ADD     #12,TXCNT1(R4)          ;CORRECT FOR CRC AND HEADER
 824  001560' 004767  002456                  JSR     PC,CTS          ;GO GET CLEAR TO SEND
 825  001564' 016764  176400  005064'         MOV     TIMELN,TMTTIM(R4)       ;GIVE 'EM 1/2 MINUTE
 826  001572' 004767  001632                  JSR     PC,TMT          ;GO TRANSMIT
 827  001576' 000434                          BR      GERR            ;ERROR RETURN
 828  001600' 042774  000400  004762'         BIC     #400,@TMTSAV(R4)        ;KILL REQ TO SEND
 829  001606' 000401          ONETMT: BR      1$                      ;MAKE THIS A NOP=240 FOR ONE
 830                                                                  ;WAY OUT, TMT ONLY, OTHER SIDE
```



```
 831                                                                  ;OF LINE MUST DO ONE WAY IN.
 832  001610' 000417                          BR      PPHDM2                  ;USED FOR ONE WAY OUT ONLY
 833  001612' 016464  004766' 004772' 1$:     MOV     AMSGBF(R4),RECADR(R4)   ;SET UP INPUT AREA
 834  001620' 016764  000000G 004776'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
 835  001626' 062764  000012  004776'         ADD     #12,RECCNT(R4)          ;CORRECT FOR HEADER AND CRC
 836  001634' 016764  176330  005002'         MOV     TIMELN,RECTIM(R4)       ;GIVE EM 1/2 MINUTES
 837  001642' 004767  000366                  JSR     PC,REC          ;GO RECEIVE
 838  001646' 000413                          BR      HERR            ;ERROR RETURN
 839  001650' 005264  005034'         PPHDM2: INC     LOOP(R4)        ;COUNT AN ITERATION
 840  001654' 026764  176306  005034'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH?
 841  001662' 003325                          BGT     PPHDM           ;BR IF NOT
 842  001664' 000167  177510                  JMP     LNDONE
 843                                  ;
 844                                  ;
 845  001670' 042774  000400  004762' GERR:   BIC     #400,@TMTSAV(R4)        ;KILL REQ TO SEND
 846  001676' 016401  005020'         HERR:   MOV     CURLIN(R4),R1   ;GET LINE # ADDRESS
 847  001702' 105261  000020                  INCB    20(R1)          ;COUNT AN ERROR
 848  001706' 126167  000020  176256          CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
 849  001714' 002267                          BGE     DROP            ;BR IF YES
 850  001716' 000754                          BR      PPHDM2          ;GO TRY AGAIN
 851                                  ;
 852                                  ;
 853                                  ;POINT -TO-POINT FULL DUPLEX
 854                                  ;
 855  001720' 032703  000040          PPFD:   BIT     #40,R0          ;MASTER OR SLAVE?
 856  001724' 001003                          BNE     1$              ;BR IF MASTER
 857  001726' 012764  177777  005110'         MOV     #177777,FDFLAG(R4)      ;SHOW ITS SLAVE
 858  001734' 004767  002302                  JSR     PC,CTS          ;GO GET CLEAR TO SEND
 859  001740' 016464  004766' 004772' PPFD1:  MOV     AMSGBF(R4),RECADR(R4)   ;GET INPUT AREA
 860  001746' 016764  000000G 004776'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
 861  001754' 062764  000012  004776'         ADD     #12,RECCNT(R4)  ;CORRECT FOR CRC AND HEADER
 862  001762' 004767  000246                  JSR     PC,REC          ;GO GET RECEIVING STARTED
 863  001766' 000472                          BR      JERR            ;REC ERROR RETURN
 864  001770' 000433                          BR      FDRCDN          ;REC DONE
 865  001772' 012764  000000G 005044'         MOV     #MSGHED,TXADR1(R4)      ;FULL DUPLEX RETURN FROM REC
 866                                                                  ;GET TMT ADDR
 867  002000' 016764  000000G 005054'         MOV     MSGSIZ,TXCNT1(R4)       ;GET TMT COUNT
 868  002006' 062764  000012  005054'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC
 869  002014' 004767  001410                  JSR     PC,TMT          ;GO TMT
 870  002020' 000455                          BR      JERR            ;TMT ERROR RETURN
 871  002022' 000412                          BR      1$              ;GOOD TMT DONE RETURN
 872  002024' 012700  002222'                 MOV     #FDTO,R0        ;FULL DUP RETURN, SET UP FOR TIME OUT
 873  002030' 016701  176134                  MOV     TIMELN,R1               ;GIVE EM 1 MINUTE
 874  002034' 066701  176130                  ADD     TIMELN,R1
 875  002040' 012702  000000                  MOV     #0,R2           ;SHOW ITS A REC WAIT
 876  002044' 000167  002324                  JMP     TIMKEP          ;GO WAIT
 877                                  ;
 878  002050' 012764  000001  005074' 1$:     MOV     #1,TMTDON(R4)   ;SHOW TMT IS DONE
 879  002056' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
 880                                  ;
 881                                  ;
 882  002060' 005764  005074'         FDRCDN: TST     TMTDON(R4)      ;IS TMT DONE YET?
 883  002064' 001020                          BNE     4$              ;BR IF YES
 884  002066' 005264  005150'                 INC     FDBKR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
 885  002072' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
 886  002076' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
```

```
  887  002102' 005767  003042                TST     FDBKR4
  888  002106' 001402                       BEQ     2$
  889  002110' 005004                       CLR     R4
  890  002112' 000402                       BR      3$
  891  002114' 012704  000002       2$:     MOV     #2,R4
  892  002120' 005364  005150'      3$:     DEC     FDBKR4(R4)
  893  002124' 000755                       BR      FDRCDN          ;BR AND WAIT SOME MORE
  894  002126' 005264  005034'      4$:     INC     LOOP(R4)        ;COUNT AN ITERATION
  895  002132' 026764  176030  005034'      CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH
  896  002140' 003277                       BGT     PPFD1           ;NO, GO DO MORE
  897  002142' 042774  000400  004762'      BIC     #400,@TMTSAV(R4)        ;KILL REQ TO SEND
  898  002150' 000167  177224               JMP     LNDONE
  899                                       ;
  900                                       ;
  901                                       ;FD TMT ERROR
  902  002154' 012764  000001  005074' JERR:  MOV     #1,TMTDON(R4)   ;DON'T BOTHER WAITING FOR TMT
  903  002162' 012764  177777  005014'      MOV     #177777,ABORT(R4)       ;TELL INT ROUTINES TO GIVE UP
  904  002170' 016401  005020'              MOV     CURLIN(R4),R1   ;GET CURRENT LINE INFO
  905  002174' 105261  000020               INCB    20(R1)          ;COUNT ANOTHER ERROR
  906  002200' 126167  000020  175764       CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
  907  002206' 002724                       BLT     FDRCDN          ;NO GO TRY AGAIN
  908  002210' 042774  000400  004762'      BIC     #400,@TMTSAV(R4)        ;KILL REQ TO SEND
  909  002216' 000167  177252               JMP     DROP            ;GIVE UP ON THIS LINE
  910                                       ;
  911                                       ;
  912                                       ;FULL DUPLEX RECEIVER TIMEOUTS COME HERE
  913  002222' 012764  002154' 005040' FDTO:  MOV     #JERR,RECRTN(R4)        ;SET UP CORRECT RETURN
  914  002230' 000167  000650               JMP     RDTO            ;GO REPORT REC TIME OUT
  915                                       ;
```



```
  916                                       ;
  917                                       ;SUBROUTINE TO RECEIVE DATA
  918                                       ;CALLED WITH A JSR PC,REC
  919                                       ;AFTER SETTING LOCATIONS RECADR(R4),RECCNT(R4),RECTIM(R4),VCTSAV(R4),
  920                                       ;AND DVASAV(R4). RETURNS ON ERROR OR RETURNS
  921                                       ;2 BYTES PAST CALL IF NO ERROR
  922                                       ;
  923  002234' 012664  005040'      REC:    MOV     (R6)+,RECRTN(R4);SAVE RTN ADDR
  924  002240' 016401  004756'              MOV     VCTSAV(R4),R1   ;GET VECTOR ADR
  925  002244' 012711  002500'              MOV     #RECINT,(R1)    ;MOVE IN INT. ROUTINE ADR
  926  002250' 060421                       ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
  927  002252' 116711  175534               MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
  928  002256' 005064  005014'              CLR     ABORT(R4)       ;CLEAR ABORT FLAG
  929  002262' 016403  004752'              MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
  930  002266' 005464  004776'              NEG     RECCNT(R4)      ;GET RIGHT FORM OF COUNT
  931  002272' 005002                       CLR     R2              ;ASSUME USING PRI REGS, SET BUMPER TO 0
  932  002274' 032713  000004               BIT     #4,(R3)         ;ARE WE USING PRI OR SEC REG?
  933  002300' 001402                       BEQ     1$              ;BR IF PRI, BUMPER =0
  934  002302' 012702  000004               MOV     #4,R2           ;SET BUMPER TO 4 IF SEC.
  935  002306' 112763  000001  000005 1$:   MOVB    #1,5(R3)        ;POINT TO RXCC-P
  936  002314' 150263  000005               BISB    R2,5(R3)        ;BUMP TO WHICH SET
  937  002320' 016463  004776' 000006       MOV     RECCNT(R4),6(R3)        ;WRITE RXCC-P
  938  002326' 112763  000011  000005       MOVB    #11,5(R3)       ;POINT TO SYNC REG
  939  002334' 016763  175634  000006       MOV     SYNC,6(R3)      ;WRITE IN SYNCS
  940  002342' 112763  000012  000005       MOVB    #12,5(R3)       ;POINT TO MISC REG
  941  002350' 012763  004000  000006       MOV     #4000,6(R3)     ;SET CHAR SIZE TO 8 BITS
  942  002356' 042763  177752  000004       BIC     #177752,4(R3)   ;POINT TO RXBA-P
  943                                                               ;AND CLEAR ERR BITS FOR REC
  944  002364' 150263  000005               BISB    R2,5(R3)        ;BUMP TO PRI OR SEC
  945  002370' 016467  004772' 002562       MOV     RECADR(R4),ADR  ;PREPARE FOR GETPA CALL
  946  002376' 104413  005160'              GETPA$, ADR             ;GET PHYSICAL ADDRESS FROM 16-BIT ADR
  947  002402' 006367  002556               ASL     EA              ;LINE UP EA BITS
  948  002406' 152763  000020  000005       BISB    #20,5(R3)               ;PUT IN WRITE ENABLE BIT
  949  002414' 156763  002544  000005       BISB    EA,5(R3)        ;PUT EA BITS IN REG
  950  002422' 016763  002534  000006       MOV     PA,6(R3)        ;WRITE ALL 18 BITS
  951  002430' 005764  005110'              TST     FDFLAG(R4)              ;FULL DUPLEX?
  952  002434' 001407                       BEQ     2$              ;BR IF NOT
  953  002436' 012713  000041               MOV     #41,(R3)                ;KICK OFF REC
  954  002442' 016402  005040'              MOV     RECRTN(R4),R2   ;GET RETURN ADDRESS
  955  002446' 062702  000004               ADD     #4,R2           ;FD RETURN
  956  002452' 010207                       MOV     R2,R7           ;RETURN
  957  002454' 012700  003104'      2$:     MOV     #RDTO,R0        ;RETURN ADDR
  958  002460' 016401  005002'              MOV     RECTIM(R4),R1   ;HOW LONG TO WAIT
  959  002464' 012702  000000               MOV     #0,R2           ;SHOWS ITS A REC WAIT
  960  002470' 012713  000041               MOV     #41,(R3)        ;KICK OFF REC
  961  002474' 000167  001674               JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
  962                                       ;
  963                                       ;RECEIVER INTERRUPT ROUTINE
  964                                       ;
  965  002500' 000403               RECINT: BR      1$
  966                                       ;-----------------------------------------------------------------
  967  002502' 000004  002522' 000000'      PIRQ$,RECDN+4,BEGIN             ; RTI AND QUEUE UP TO CONTINUE AT RECDN+4
  968                                       ;-----------------------------------------------------------------
  969  002510'                       1$:
  970                                       ;-----------------------------------------------------------------
  971  002510' 000004  002516' 000000'      PIRQ$,RECDN,BEGIN               ; RTI AND QUEUE UP TO CONTINUE AT RECDN
```

```
                                        ;------------------------------------------------------------
                                ;
                                ;
                                ;
002516' 005004                  RECON:  CLR     R4              ;SHOW WE'RE LINE 0
002520' 000402                          BR      1$
002522' 012704  000002                  MOV     #2,R4           ;SHOW WE'RE LINE 1
002526' 005074  004752'         1$:     CLR     @DVASAV(R4)     ;SHUT REC UP
002532' 005764  005014'                 TST     ABORT(R4)       ;GIVE UP?
002536' 001401                          BEQ     2$              ;BR IF NOT
002540' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
002542' 005064  005114'         2$:     CLR     RCVTO(R4)       ;RESET TIME OUT RETRY COUNT
002546' 005064  005222'                 CLR     RETTAB(R4)      ;KILL IMPENDING TIMEOUT
002552' 004767  000476                  JSR     PC,CHKREC       ;CHECK FOR REC, HDW ERRORS
002556' 005767  000000G         3$:     TST     CKMSG           ;IS CHECK MSG ROUTINE BUSY?
002562' 001420                          BEQ     6$              ;BR IF NOT
002564' 005264  005154'                 INC     CHKR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
002570' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
002574' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
002600' 005767  002350                  TST     CHKR4
002604' 001402                          BEQ     4$
002606' 005004                          CLR     R4
002610' 000402                          BR      5$
002612' 012704  000002          4$:     MOV     #2,R4
002616' 005364  005154'         5$:     DEC     CHKR4(R4)
002622' 000755                          BR      3$
002624' 016402  004766'         6$:     MOV     AMSGBF(R4),R2   ;GET REC DATA
002630' 026762  177776G 000006          CMP     MSG-2,6(R2)     ;IS HEADER CRC OK?
002636' 001407                          BEQ     7$              ;BR IF OK
002640' 005767  175320                  TST     REPORT          ;REPORT ERR NOW?
002644' 001055                          BNE     X5              ;BR IF LATER
002646' 012767  003000  175200          MOV     #003000,ASTAT   ;SHOW ERR TYPE
002654' 000466                          BR      RECERR
002656'                         7$:
002656' 117467  005020' 000040          MOVB    @CURLIN(R4),9$  ;GET LINE #
002664' 042767  177760  000032          BIC     #177760,9$      ;LEAVE ONLY LINE BITS
002672' 005767  175266                  TST     REPORT          ;REPORT BAD DATA?
002676' 001403                          BEQ     8$              ;BR IF YES
002700' 052767  100000  000016          BIS     #100000,9$      ;SET FLAG TO CKMSG
002706' 062702  000010          8$:     ADD     #10,R2          ;GET ADDR OF MSG
002712' 010267  000010                  MOV     R2,9$+2
002716' 004567  000002G                 JSR     R5,CKMSG+2      ;GO CHECK MSG
002722' 000000'                         BEGIN
002724' 000000                  9$:     0
002726' 000000                          0
002730' 000417                          BR      X4              ;BAD RETURN
002732' 066702  000000G                 ADD     MSGSIZ,R2       ;POINT IT TO MSG CRC, DATA WAS OK
002736' 126722  000000G                 CMPB    CRCMSG,(R2)+    ;CHECK 1ST CRC BYTE
002742' 001025                          BNE     CER
002744' 126722  000001G                 CMPB    CRCMSG+1,(R2)+  ;CHECK SECOND
002750' 001022                          BNE     CER
002752' 062764  000002  005040'         ADD     #2,RECRTN(R4)   ;GOOD RETURN
002760' 005064  005222'         RRET:   CLR     RETTAB(R4)      ;KILL IMPENDING TIMEOUTS
002764' 000174  005040'                 JMP     @RECRTN(R4)     ;AND GO BACK
                                ;
                                ;
```



```
002770' 005767  175170          X4:     TST     REPORT          ;REPORT ERR NOW OR LATER?
002774' 001001                          BNE     X5              ;BR IF LATER
002776' 000770                          BR      RRET            ;OTHERWISE RETURN
                                ;
003000' 117402  005020'         X5:     MOVB    @CURLIN(R4),R2  ;GET CURRENT LINE #
003004' 042702  177760                  BIC     #177760,R2      ;
003010' 105262  000252'                 INCB    ECOUNT(R2)      ;COUNT AN ERROR FOR THAT LINE
003014' 000761                          BR      RRET            ;AND RETURN
                                ;
                                ;
003016' 005767  175142          CER:    TST     REPORT          ;REPORT NOW OR LATER?
003022' 001366                          BNE     X5              ;BR IF LATER
003024' 012767  004000  175022          MOV     #004000,ASTAT   ;INDICATE ERROR TYPE
003032' 117402  005020'         RECERR: MOVB    @CURLIN(R4),R2  ;GENERATE LINE #
003036' 042702  177760                  BIC     #177760,R2      ;
003042' 050267  175006                  BIS     R2,ASTAT        ;ADD IN LINE #
003046' 005264  005134'                 INC     CRCR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
                                        ;************************************************************
003052' 104404  000000'                 ERROR$,BEGIN            ;CRC ERROR
                                        ;************************************************************
003056' 005767  002052                  TST     CRCR4
003062' 001402                          BEQ     1$
003064' 005004                          CLR     R4
003066' 000402                          BR      2$
003070' 012704  000002          1$:     MOV     #2,R4
003074' 005364  005134'         2$:     DEC     CRCR4(R4)
003100' 000167  177654                  JMP     RRET            ;BAD RETURN
                                ;
                                ;READ TIME OUT ROUTINE
                                ;
003104' 012764  177777  005014' RDTO:   MOV     #177777,ABORT(R4);TELL INT ROUTINES TO GIVE UP
003112' 004767  000136                  JSR     PC,CHKREC       ;CHECK FOR REC HDW ERRORS
003116' 005264  005114'                 INC     RCVTO(R4)       ;COUNT ANOTHER RETRY
003122' 026467  005114' 175042          CMP     RCVTO(R4),ERRLVL        ;ENOUGH TRYS?
003130' 001042                          BNE     4$              ;BR IF NO
003132' 017467  004752' 174712          MOV     @DVASAV(R4),ACSR;GET REC STATUS
003140' 005074  004752'                 CLR     @DVASAV(R4)             ;KILL REC
003144' 016467  004752' 174676          MOV     DVASAV(R4),CSRA ;PUT ADDR IN FOR ERR CALL
003152' 016401  005020'                 MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE BYTE
003156' 111102                          MOVB    (R1),R2         ;GET VALUE OF PHY LINE #
003160' 042702  177760                  BIC     #177760,R2      ;THROW OTHER BITS AWAY
003164' 062702  001000                  ADD     #1000,R2        ;INDICATE REC TIMEOUT ERR
003170' 010267  174660                  MOV     R2,ASTAT        ;SET UP FOR ER CALL
003174' 005264  005140'                 INC     RCTOR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
                                        ;************************************************************
003200' 104404  000000'                 ERROR$,BEGIN            ;REC TIMED OUT
                                        ;************************************************************
003204' 005767  001730                  TST     RCTOR4
003210' 001402                          BEQ     1$
003212' 005004                          CLR     R4
003214' 000402                          BR      2$
003216' 012704  000002          1$:     MOV     #2,R4
003222' 005364  005140'         2$:     DEC     RCTOR4(R4)
003226' 005064  005114'                 CLR     RCVTO(R4)       ;RESET RETRY COUNT
003232' 000174  005040'         3$:     JMP     @RECRTN(R4)     ;ERROR RTN
                                ;
```

```
1084  003236' 016401  005020'          4$:     MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
1085  003242' 105361  000020                  DECB    20(R1)          ;DEC ERROR COUNT TO CORRECT FOR
1086                                                                  ;INSUING INC
1087  003246' 005364  005034'                  DEC     LOOP(R4)        ;CORRECT FOR LATER CALLING THIS A GOOD RUN
1088  003252' 000767                           BR      3$              ;RETURN
1089                                  ;
1090                                  ;
1091                                  ;SUBROUTINE CHECKS FOR ANY OF 3 HARDWARE SET ERR BITS
1092                                  ;AND REPORTS THE ERR AND DOES AN ERROR RETURN TO THE
1093                                  ;ROUTINE WHICH CALLED FOR A RECEIVE
1094                                  ;
1095                                  ;
1096  003254' 016403  004752'          CHKREC: MOV     DVASAV(R4),R3   ;GET DEVICE REG
1097  003260' 012702  011000                   MOV     #11000,R2       ;PRESET ERR #
1098  003264' 032763  000040  000004           BIT     #40,4(R3)       ;NON-EXIST MEM ERROR?
1099  003272' 001401                           BEQ     1$              ;BR IF NOT
1100  003274' 000416                           BR      3$              ;ELSE GO REPORT
1101  003276' 012702  040000           1$:     MOV     #40000,R2       ;PRESET ERR #
1102  003302' 032763  000010  000004           BIT     #10,4(R3)       ;RX LATENCY ERROR?
1103  003310' 001401                           BEQ     2$              ;BR IF NOT
1104  003312' 000407                           BR      3$              ;ELSE GO REPORT
1105  003314' 012702  012000           2$:     MOV     #12000,R2       ;PRESET ERR #
1106  003320' 032763  000002  000004           BIT     #2,4(R3)        ;RX CLOCK LOSS ERROR?
1107  003326' 001001                           BNE     3$              ;BR IF YES, ELSE
1108  003330' 000207                           RTS     PC              ;RETURN OK
1109  003332' 005726                   3$:     TST     (R6)+           ;POP JSR RETURN OFF STACK
1110                                                                  ;SINCE WE WON'T GO BACK THAT WAY
1111  003334' 005767  174624                   TST     REPORT          ;REPORT ERRORS NOW?
1112  003340' 001217                           BNE     X5              ;BR IF NOT
1113  003342' 010367  174502                   MOV     R3,CSRA ;PUT IN FOR ERROR CALL
1114  003346' 011367  174500                   MOV     (R3),ACSR       ;"
1115  003352' 016401  005020'                  MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
1116  003356' 111101                           MOVB    (R1),R1         ;GET VALUE OF CUR LINE
1117  003360' 042702  177760                   BIC     #177760,R2      ;LEAVE LINE # ONLY
1118  003364' 060102                           ADD     R1,R2           ;ADD IN ERR #
1119  003366' 010267  174462                   MOV     R2,ASTAT        ;PUT IN FOR ERROR CALL
1120  003372' 005264  005130'                  INC     DTERR4(R4)      ;SEE NOTE 1 IN FRONT OFF LISTING
1121                                          ;*****************************************************************
1122  003376' 104404  000000'                  ERRORS,BEGIN            ;REC ERROR
1123                                          ;*****************************************************************
1124  003402' 005767  001522                   TST     DTERR4
1125  003406' 001402                           BEQ     4$
1126  003410' 005004                           CLR     R4
1127  003412' 000402                           BR      5$
1128  003414' 012704  000002           4$:     MOV     #2,R4
1129  003420' 005364  005130'          5$:     DEC     DTERR4(R4)
1130  003424' 000174  005040'                  JMP     @RECRTN(R4)     ;GO BACK TO ERROR RTN
1131                                  ;
1132                                  ;
1133                                  ;SUBROUTINE TO TRANSMIT DATA
1134                                  ;CALLED WITH A JSR PC,TMT
1135                                  ;AFTER SETTING LOCATIONS TXADR1(R1), TXCNT1(R4), TMTIM(R4),
1136                                  ;VCTSAV(R4), DVASAV(R4), SUBROUTINE CTS MUST BE CALLED FIRST.
1137                                  ;RETURNS ON ERROR, RETURNS 2 BYTES PAST NORMAL RETURN
1138                                  ;IF NO ERROR
1139                                  ;
```



```
1140  003430' 012664  005070'          TMT:    MOV     (SP)+,TMTRET(R4)  ;SAVE RTN ADDRESS
1141  003434' 005064  005074'                   CLR     TMTDON(R4)      ;INITIALIZE FLAG
1142  003440' 016401  004756'                   MOV     VCTSAV(R4),R1   ;GET REC VECTOR ADDRESS
1143  003444' 062701  000004                    ADD     #4,R1           ;MAKE IT TMT VECTOR
1144  003450' 012711  003762'                   MOV     #TMTINT,(R1)    ;MOVE IN INT ROUTINE ADDRESS
1145  003454' 062021                            ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
1146  003456' 116711  174330                    MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
1147  003462' 005064  005014'                   CLR     ABORT(R4)       ;CLEAR TIME OUT ABORT FLAG
1148  003466' 016403  004752'                   MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
1149  003472' 062764  000016  005054'           ADD     #14.,TXCNT1(R4) ;MAKE SURE MY LAST CHAR GETS SENT, SO
1150                                                                    ;SEND 4 GARBAGE AT END, PLUS 10 SYNCS IN FRONT
1151  003500' 005464  005054'                   NEG     TXCNT1(R4)      ;GET COUNT IN RIGHT FORM
1152  003504' 005002                            CLR     R2              ;ASSUME USING PRI REGS, SET BUMPER TO 0
1153  003506' 032763  000004  000002            BIT     #4,2(R3)        ;ARE WE USING PRI OR SEC?
1154  003514' 001402                            BEQ     11$             ;BR IF USING PRI, BUMPER =0
1155  003516' 012702  000004                    MOV     #4,R2           ;ELSE SET BUMPER TO 4
1156  003522' 112763  000003  000005  11$:     MOVB    #3,5(R3)        ;POINT TO TXCC-P
1157  003530' 150263  000005                    BISB    R2,5(R3)        ;BUMP TO WHICH SET
1158  003534' 016463  005054' 000006            MOV     TXCNT1(R4),6(R3)        ;WRITE TXCC-P
1159  003542' 112763  000012  000005            MOVB    #12,5(R3)       ;POINT TO MISC REG
1160  003550' 012763  004000  000006            MOV     #4000,6(R3)     ;SET TO 8 BITS PER CHAR
1161  003556' 112763  000011  000005            MOVB    #11,5(R3)       ;POINT TO SYNC REG
1162  003564' 016763  174404  000006            MOV     SYNC,6(R3)      ;WRITE SYNC REG
1163  003572' 042763  177725  000004            BIC     #177725,4(R3)   ;CLEAR ALL BUT REC ERR BITS
1164  003600' 112763  000002  000005            MOVB    #2,5(R3)        ;POINT TO TXBA-P
1165  003606' 150263  000005                    BISB    R2,5(R3)        ;POINT TO WHICH SET
1166  003612' 016467  005044' 001340            MOV     TXADR1(R4),ADR  ;PREPARE FOR GETPA CALL
1167  003620' 162767  000012  001332            SUB     #10.,ADR        ;MOVE PTR TO START OF SYNC FIELD
1168  003626' 104413  005160'           GETPA$, ADR             ;GET PHYSICAL ADDRESS FROM 16-BIT ADR
1169  003632' 006367  001326                    ASL     EA              ;LINE UP EA BITS
1170  003636' 152763  000020  000005            BISB    #20,5(R3)               ;PUT IN WR ENABLE BIT
1171  003644' 156763  001314  000005            BISB    EA,5(R3)        ;MOV EA BITS INTO PLACE
1172  003652' 016763  001304  000006            MOV     PA,6(R3)        ;WRITE ALL 18 BITS
1173  003660' 005764  005110'                   TST     FDFLAG(R4)      ;ARE WE FULL DUPLEX?
1174  003664' 001423                            BEQ     5$              ;BR IF NOT, ELSE
1175  003666' 100012                            BPL     4$              ;SKIP IF MASTER
1176  003672' 012700  003710'                   MOV     #3$,R0          ;CONTINUE AT 3$ AFTER GOING TO TIMKEP
1177                                                                    ;TO LET OTHER LINE CONTINUE
1178  003674' 012701  000000                    MOV     #0,R1           ;COME BACK ASAP
1179  003700' 012702  000004                    MOV     #4,R2           ;SHOW ITS A TMT WAIT
1180  003704' 000167  000464                    JMP     TIMKEP          ;LET OTHER LINE GO
1181  003710' 016403  004752'           3$:     MOV     DVASAV(R4),R3   ;PUT DEVICE ADDR BACK IN R3
1182  003714' 112763  000043  000002    4$:     MOVB    #43,2(R3)       ;SET TMT INT ENABLE
1183  003722' 016402  005070'                   MOV     TMTRET(R4),R2   ;GET BAD RET ADDR
1184  003726' 062702  000004                    ADD     #4,R2           ;MAKE IT THE FULL DUPLEX RETURN
1185  003732' 010207                            MOV     R2,R7           ;AND RETURN
1186                                  ;
1187                                  ;
1188  003734' 012700  004060'           5$:     MOV     #TXTO,R0        ;TIME OUT RETURN ADDR
1189  003740' 016401  005064'                   MOV     TMTTIM(R4),R1   ;HOW LONG TO WAIT
1190  003744' 012702  000004                    MOV     #4,R2           ;SHOW ITS TRANSMITTER
1191  003750' 112763  000043  000002            MOVB    #43,2(R3)       ;SET TMT INT ENABLE AND GO BIT
1192  003756' 000167  000412                    JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
1193                                  ;TMT INTERRUPT ROUTINE
1194                                  ;
1195  003762' 000403                  TMTINT: BR      1$
```

```
                                        ;-------------------------------------------------------------------
003764' 000004  004004' 000000'         PIRQ$,TIDONE+4,$BEGIN           ; RTI AND QUEUE UP TO CONTINUE AT TIDONE+4
                                        ;-------------------------------------------------------------------
003772'                         1$:
                                        ;-------------------------------------------------------------------
003772' 000004  004000' 000000'         PIRQ$,TIDONE,BEGIN              ; RTI AND QUEUE UP TO CONTINUE AT TIDONE
                                        ;-------------------------------------------------------------------
                                        ;
004000' 005004                  TIDONE: CLR     R4              ;SET TO LINE 0
004002' 000402                          BR      1$
004004' 012704  000002                  MOV     #2,R4           ;SET TO LINE 1
004010' 005064  005226'         1$:     CLR     RETTAB+4(R4)    ;KILL IMPENDING TIMEOUT
004014' 005764  005014'                 TST     ABORT(R4)       ;SHOULD WE GIVE UP?
004020' 001401                          BEQ     2$              ;BR IF NO, ELSE
004022' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
004024' 016403  004752'         2$:     MOV     DVASAV(R4),R3   ;GET DEVICE ADDR
004030' 042763  000047  000002          BIC     #47,2(R3)       ;KILL TMT'ER
004036' 032763  000025  000004          BIT     #25,4(R3)       ;ANY HDW ERR'S ?
004044' 001005                          BNE     TXTO            ;BR IF YES
004046' 062764  000002  005070'         ADD     #2,TMTRET(R4)   ;MAKE IT A GOOD RETURN
004054' 000174  005070'                 JMP     @TMTRET(R4)     ;GO BACK OK
                                ;
                                ;
                                ;
                                ;ROUTINE IF TRANSMIT TIMES OUT
                                ;
004060' 012764  177777  005014' TXTO:   MOV     #177777,ABORT(R4)       ;TELL INT ROUTINES TO GIVE UP
004066' 016403  004752'                 MOV     DVASAV(R4),R3   ;GET ADDR OF STATUS
004072' 062703  000002                  ADD     #2,R3           ;MAKE IT TMT CSR
004076' 011367  173750                  MOV     (R3),ACSR       ;SAVE STATUS
004102' 042713  000041                  BIC     #41,(R3)                ;KILL TRANSMISSION AND INT
004106' 010367  173736                  MOV     R3,CSRA         ;STORE FOR ERR CALL
004112' 016302  000002                  MOV     2(R3),R2        ;GET ERR REG CONTENTS
004116' 012703  005000                  MOV     #5000,R3        ;ASSUME ITS A TIME OUT
004122' 032702  000020                  BIT     #20,R2          ;TX NON-EXIST MEM?
004126' 001403                          BEQ     1$              ;BR IF NOT
004130' 012703  006000                  MOV     #6000,R3        ;ELSE SHOW IT
004134' 000413                          BR      3$              ;AND GO REPORT
004136' 032702  000004          1$:     BIT     #4,R2           ;TX LATENCY ERR?
004142' 001403                          BEQ     2$              ;BR IF NOT
004144' 012703  007000                  MOV     #7000,R3        ;ELSE SHOW IT
004150' 000405                          BR      3$              ;AND GO REPORT
004152' 032702  000001          2$:     BIT     #1,R2           ;TX CLOCK LOST ERROR?
004156' 001402                          BEQ     3$              ;BR IF NOT
004160' 012703  002000                  MOV     #2000,R3        ;ELSE SHOW IT AND REPORT IT
004164' 016401  005020'         3$:     MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
004170' 111102                          MOVB    (R1),R2         ;GET VALUE
004172' 042702  177760                  BIC     #177760,R2      ;LEAVE ONLY LINE #
004176' 060302                          ADD     R3,R2   ;INDICATE TMT TIMEOUT
004200' 010267  173650                  MOV     R2,ASTAT        ;SET UP FOR ERROR CALL
004204' 005264  005144'                 INC     TXTOR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
                                        ;*****************************************************************
004210' 104404  000000'                 ERROR$,BEGIN            ;TMT ERROR SEE ASTAT INDICATOR
                                        ;*****************************************************************
004214' 005767  000724                  TST     TXTOR4
004220' 001432                          BEQ     4$
```



```
004222' 005004                          CLR     R4
004224' 000402                          BR      5$
004226' 012704  000002          4$:     MOV     #2,R4
004232' 005364  005144'         5$:     DEC     TXTOR4(R4)
004236' 000174  005070'                 JMP     @TMTRET(R4)     ;BAD RETURN
                                ;
                                ;
                                ;
                                ;
                                ;
                                ;SUB GETS CLEAR TO SEND FOR LINE IN DVASAV(R4) AND VCTSAV(R4)
                                ;
004242' 012664  005104'         CTS:    MOV     (R6)+,CTSRET(R4)        ;SAVE RET ADDRESS
004246' 016401  004756'                 MOV     VCTSAV(R4),R1   ;GET VECTOR ADDR
004252' 062701  000004                  ADD     #4,R1           ;MAKE IT B VECTOR
004256' 012711  004320'                 MOV     #CTSINT,(R1)    ;MOV IN INT ROUTINE ADDR
004262' 060421                          ADD     R4,(R1)+        ;SHOWS WHICH LINE
004264' 116711  173522                  MOVB    BR1,(R1)        ;GET BR1 LEVEL
004270' 016403  004752'                 MOV     DVASAV(R4),R3   ;GET DEVICE ADDR
004274' 032763  020000  000002          BIT     #20000,2(R3)  ;IS CLEAR TO SEND UP?
004302' 001402                          BEQ     RTSSET          ;BR IF NOT
004304' 016407  005104'                 MOV     CTSRET(R4),R7   ;RETURN
004310' 012763  001422  000002  RTSSET: MOV     #1422,2(R3)  ;SET REQ TO SEND AND INT, ENABLE
                                                                ;DTR  AND IDLE BIT TOO.
004316' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                ;INT ROUTINE
004320' 000403                  CTSINT: BR      1$              ;DIFFERENT PIRQ'S
                                        ;-------------------------------------------------------------------
004322' 000004  004336' 000000'         PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
                                        ;-------------------------------------------------------------------
004330'                         1$:
                                        ;-------------------------------------------------------------------
004330' 000004  004344' 000000'         PIRQ$,3$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 3$
                                        ;-------------------------------------------------------------------
004336' 012704  000002          2$:     MOV     #2,R4           ;SET UP FOR LINE 1
004342' 000401                          BR      4$
004344' 005004                  3$:     CLR     R4              ;SET UP FOR LINE 0
004346' 016403  004752'         4$:     MOV     DVASAV(R4),R3   ;GET DEVICE ADDR
004352' 032763  020000  000002          BIT     #20000,2(R3)  ;IS CLEAR TO SEND SET?
004360' 001753                          BEQ     RTSSET          ;GO TRY AGAIN IF NOT
004362' 042774  000020  004752'         BIC     #20,@DVASAV(R4) ;DISABLE DATA SET CHG INT'S
004370' 016407  005104'                 MOV     CTSRET(R4),R7   ;RETURN
                                ;
                                ;
                                ;
                                ;ALL ROUTINES COME HERE WHILE WAITING FOR I/O
                                ;ROUTINE EITHER KICKS OFF SECOND LINE, NOTIFYS
                                ;ROUTINES OF TIMEOUTS, OR JUST DOES BREAKS.
                                ;I/O ROUTINES THAT FINISH UP BEFORE TIMEOUT VALUE
                                ;MUST CLEAR LOCATION RETTAB(R4) +0 IF RECEIVER OR +4
                                ;IF TMT.
                                ;CALLED WITH A JMP TO TIMKEP AFTER LOADING
                                ; R0 = WHERE TO RETURN IF TIME OUT
                                ; R1 = HOW LONG TO WAIT (IN SECONDS)
                                ; R2 = 0 IF ITS A REC TIMEING OR 4 IF ITS A TMT WAIT
                                ; R4 = 0 FOR ACTIVE LINE 0, 2 FOR ACTIVE LINE 1
```

```
1308                                    ;
1309                                    ;
1310  004374' 060402                    TIMKEp: ADD     R4,R2           ;FIGURE OUT WHICH TABLE ENTRY
1311  004376' 010162  005212'                   MOV     R1,TIMTAB(R2)   ;STORE HOW LONG TO WAIT
1312  004402' 067762  000000G 005212'          ADD     #TIME,TIMTAB(R2)        ;ADD PRESENT TIME
1313  004410' 010462  005232'                   MOV     R4,R4TAB(R2)    ;SAVE WHICH ACTIVE LINE
1314  004414' 010062  005222'                   MOV     R0,RETTAB(R2)   ;WHERE TO RET TO, PLUS
1315                                                                    ;IF THIS ENTRY IN TABLE IS
1316                                                                    ;0 THEN NO TIMEOUT WILL
1317                                                                    ;OCCUR FOR THIS ENTRY SET
1318  004420' 022767  000002  000364            CMP     #2,LINACT       ;ARE 2 LINES GOING?
1319  004426' 001410                            BEQ     TIMCHK          ;BR IF YES
1320  004430' 005704                            TST     R4              ;ARE WE LINE 0?
1321  004432' 001402                            BEQ     1$              ; BR IF YES
1322  004434' 005004                            CLR     R4              ; IF NO MAKE IT 0
1323  004436' 000402                            BR      2$
1324  004440' 012704  000002           1$:     MOV     #2,R4           ;WE WERE 0 SO NOW DO 1
1325  004444' 000167  174344           2$:     JMP     GETNXT          ;GO DO OTHER LINE
1326                                    ;
1327                                    ;
1328  004450' 012702  005222'           TIMCHK: MOV     #RETTAB,R2      ;GET ADDR OF TIME STATUSES
1329  004454' 005722                    1$:     TST     (R2)+           ;ANY WAIT GOING ON?
1330  004456' 001026                            BNE     4$              ;BR IF YES
1331  004460' 020227  005232'           11$:    CMP     R2,#RETTAB+10   ;ARE WE DONE LOOKING
1332  004464' 001373                            BNE     1$              ;BR BACK IF NO
1333  004466' 005767  000430                    TST     PASTIM          ;TIME FOR END OF PASS
1334  004472' 001402                            BEQ     2$              ;BR IF NO
1335  004474' 000167  000064                    JMP     EPASS           ;GO REPORT END OF PASS
1336                                    ;
1337  004500' 005767  000414           2$:     TST     BRKFLG          ;ARE WE ALREADY IN A BREAK?
1338  004504' 001401                            BEQ     3$              ;BR IF NO
1339  004506' 104400                            EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
1340                                    ;
1341  004510' 012767  000001  000402    3$:     MOV     #1,BRKFLG               ;SHOW WERE DOING BREAK
1342  004516' 104407  000000'                   BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
1343  004522' 104407  000000'                   BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
1344  004526' 005067  000366                    CLR     BRKFLG          ;SHOW WERE BACK
1345  004532' 000746                            BR      TIMCHK
1346                                    ;
1347  004534' 005742                    4$:     TST     -(R2)           ;MOVE R2 BACK
1348  004536' 027762  000000G 177770            CMP     @TIME,-10(R2)   ;IS TIME UP?
1349  004544' 003002                            BGT     5$              ;BR IF YES
1350  004546' 005722                            TST     (R2)+           ;BUMP R2 BACK AGAIN
1351  004550' 000743                            BR      11$             ;GO CHECK OTHERS
1352                                    ;
1353  004552' 016204  000010           5$:     MOV     10(R2),R4       ;SET R4 CORRECTLY FOR WHICH LINE
1354  004556' 011203                            MOV     (R2),R3         ;SAVE RETURN
1355  004560' 005012                            CLR     (R2)            ;CLEAR THIS ENTRY
1356  004562' 010307                            MOV     R3,R7           ;JMP TO TIME OUT ROUTINE
1357                                    ;
1358                                    ;
1359                                    ;
1360                                    ;EPASS DOES END OF PASS CALL AND PRINTS ERR SUMMARY
1361                                    ;IF LOCATION REPORT IS NOT 0
1362                                    ;
1363  004564' 005767  173374           EPASS:  TST     REPORT          ;DO ERR SUM OR NOT
```



```
1364  004570' 001461                            BEQ     5$              ;BR IF NO
1365  004572' 005267  000146                    INC     MSGCNT          ;COUNT A PASS
1366  004576' 026767  000142  173360            CMP     MSGCNT,REPORT   ;TIME FOR A MSG?
1367  004604' 001053                            BNE     5$              ;IF NOT EVEN 8, SKIP MSG
1368  004606' 005067  000132                    CLR     MSGCNT
1369  004612' 012700  000212'                   MOV     #PHYTAB,R0      ;FIND WHICH LINES ARE RUNNING
1370  004616' 012701  011402'                   MOV     #SUM1+4,R1      ;POINTS TO MSG
1371  004622' 105710                    1$:     TSTB    (R0)            ;IS LINE IN USE?
1372  004624' 100403                            BMI     2$              ;BR IF YES
1373  004626' 012711  054130                    MOV     #54130,(R1)     ;PUT XX IN # OF ERRS
1374  004632' 000427                            BR      4$
1375  004634' 116002  000040           2$:     MOVB    32.(R0),R2      ;GET # OF ERRS
1376  004640' 105060  000040                    CLRB    32.(R0)         ;RESET ERR COUNT
1377  004644' 022702  000077                    CMP     #77,R2          ;ANY GOOD MSGS
1378  004650' 002416                            BLT     3$              ;BR IF NO
1379  004652' 010203                            MOV     R2,R3           ;PUT IN MSG
1380  004654' 042703  177770                    BIC     #177770,R3      ;LEAVE ONLY LOW DIGIT
1381  004660' 062703  000060                    ADD     #60,R3  ;MAKE IT ASCII
1382  004664' 110361  000001                    MOVB    R3,1(R1)        ;STORE IT AWAY
1383  004670' 006202                            ASR     R2              ;GET NEXT DIGIT
1384  004672' 006202                            ASR     R2
1385  004674' 006202                            ASR     R2
1386  004676' 062702  000060                    ADD     #60,R2          ;MAKE IT ASCII
1387  004702' 110211                            MOVB    R2,(R1) ;PUT IT IN MSG
1388  004704' 000402                            BR      4$
1389  004706' 012711  042102           3$:     MOV     #42102,(R1)     ;PUT IN AN 'BD' FOR BAD
1390  004712' 005200                    4$:     INC     R0              ;BUMP LINE POINTER
1391  004714' 062701  000010                    ADD     #10,R1          ;POINT TO NXT MSG
1392  004720' 020027  000232'                   CMP     R0,#PHYTAB+20   ;DONE
1393  004724' 001336                            BNE     1$              ;GO DO MORE
1394  004726' 104411  004740' 000000'           MSGN$,MSUM,BEGIN        ;ASCII MESSAGE CALL WITH COMMON HEADER
1395
1396                                    ;
1397  004734'                           5$:
1398  004734' 104402  000000'                   ENDPSS,BEGIN            ;SIGNAL END OF PASS, CONTINUE AT RESTRT
1399                                    ;
1400  004740' 011312'                   MSUM:   SUM
1401  004742' 177777                            -1
1402                                    ;
1403  004744' 000000                    MSGCNT: 0                       ;COUNTS END PASSES FOR MESSAGE CALL
1404  004746' 000000                    MASK:   0                       ;USED FOR BIC'ING IN LOOPS
1405  004750' 000000                    NXTLIN: 0                       ;ADR OF NEXT LIN
1406  004752' 000000                    DVASAV: 0                       ;HOLDS DVA FOR ACTIVE L#0
1407  004754' 000000                            0                       ;HOLDS DVA FOR ACTIVE L#1
1408  004756' 000000                    VCTSAV: 0                       ;HOLDS VCT FOR ACTIVE L#0
1409  004760' 000000                            0                       ;HOLDS VECTOR FOR ACTIVE L#1
1410  004762' 000000                    TMTSAV: 0                       ;HOLDS ADDR OF TMT CSR FOR LINE 0
1411  004764' 000000                            0                       ;SAME FOR LINE 1
1412  004766' 005242'                   AMSGBF: MSGBF0                  ;ADDRESS OF MSG BUFFER 0
1413  004770' 007262'                           MSGBF1                  ;ADDRESS OF MSG BUFFER 1
1414  004772' 000000                    RECADR: 0                       ;CURRENT REC WORD POINTER 0
1415  004774' 000000                            0                       ;CURRENT REC WORD POINTER 1
1416  004776' 000000                    RECCNT: 0                       ;REC BYTE COUNT FOR 0
1417  005000' 000000                            0                       ;REC BYTE COUNT FOR 1
1418  005002' 000000                    RECTIM: 0                       ;TEMP HOLDS TIME OUT VALUE FOR A REC
1419  005004' 000000                            0                       ;SAME BUT FOR LINE 1
```

```
1420  005006' 000000          SYNCNT: 0                       ;COUNTS HOW MANY SYN CHARACTERS REC
1421  005010' 000000                  0                       ;SHOULD WAIT FOR.
1422  005012' 000000          LINACT: 0                       ;HOLD THE NUMBER OF LINES RUNNING,0-2
1423  005014' 000000          ABORT:  0                       ;ABORT FLAG FOR L#0
1424  005016' 000000                  0                       ;ABORT FLAG FOR L#1
1425  005020' 000000          CURLIN: 0                       ;POINTS TO STATUS BYTE FOR CURRENT L#0
1426  005022' 000000                  0                       ;POINTS TO STATUS BYTE FOR CURRENT L#1
1427  005024' 002516'         RECD:   RECDN                   ;WHERE TO PIRQ AFTER REC DONE
1428  005026' 002522'                 RECDN+4                 ;
1429  005030' 000000          ERRWD:  0                       ;HOLDS ERR WORDS TEMPORARILY
1430  005032' 000000                  0                       ;HOLDS ERR WORDS TEMPORARILY
1431  005034' 000000          LOOP:   0                       ;HOLDS LOOP COUNT FOR LINE # 0
1432  005036' 000000                  0                       ;HOLDS LOOP COUNT FOR LINE # 1
1433  005040' 000000          RECRTN: 0                       ;HOLDS RETURN ADDR FOR LINE #0
1434  005042' 000000                  0                       ;HOLDS RETURN ADDR FOR LINE #1
1435  005044' 000000          TXADR1: 0                       ;HOLDS 1ST ADDR TO TMT FROM, LINE 0
1436  005046' 000000                  0                       ;SAME FOR LINE 1
1437  005050' 000000          TXADR2: 0                       ;HOLDS 2ND ADDR TO TMT FROM, LINE 0
1438  005052' 000000                  0                       ;SAME FOR LINE 1
1439  005054' 000000          TXCNT1: 0                       ;HOLDS 1ST TMT COUNT FOR LINE 0
1440  005056' 000000                  0                       ;SAME FOR LINE 1
1441  005060' 000000          TXCNT2: 0                       ;HOLDS 2ND TMT COUNT FOR LINE 0
1442  005062' 000000                  0                       ;SAME FOR LINE 1
1443  005064' 000000          TMTTIM: 0                       ;HOLDS TIME OUT TIME FOR LINE 0
1444  005066' 000000                  0                       ;SAME FOR LINE 1
1445  005070' 000000          TMTRET: 0                       ;HOLDS RTN ADDR FOR TMT, LINE 0
1446  005072' 000000                  0                       ;SAME FOR LINE 1
1447  005074' 000000          TMTDON: 0                       ;FLAG FOR WHEN A TMT IS DONE
1448  005076' 000000                  0                       ;SAME FOR OTHER LINE
1449  005100' 000000          TSYNCT: 0                       ;HOLDS # OF SYN CHAR TO SEND, LINE 0
1450  005102' 000000                  0                       ;SAME FOR LINE 1
1451  005104' 000000          CTSRET: 0                       ;HOLDS RETURN ADDR FOR CTS
1452  005106' 000000                  0                       ;SAME FOR LINE 1
1453  005110' 000000          FDFLAG: 0                       ;0 IF HALF DUPLEX, 1 IF FULL
1454  005112' 000000                  0
1455  005114' 000000          RCVTO:  0                       ;COUNTS REC TIME OUT RETRYS
1456  005116' 000000                  0
1457  005120' 000000          BRKFLG: 0                       ;IS A ONE DURING BREAKS
1458  005122' 000000          PASTIM: 0                       ;HAS A 1 WHEN ITS TIME FOR END OF PASS
1459  005124' 000000          DROPR4: 0       ;EXIT FLAGS
1460  005126' 000000                  0
1461  005130' 000000          DTERR4: 0
1462  005132' 000000                  0
1463  005134' 000000          CRCR4:  0
1464  005136' 000000                  0
1465  005140' 000000          RCTOR4: 0
1466  005142' 000000                  0
1467  005144' 000000          TXTOR4: 0
1468  005146' 000000                  0
1469  005150' 000000          FDBKR4: 0
1470  005152' 000000                  0
1471  005154' 000000          CHKR4:  0
1472  005156' 000000                  0
1473                          ;
1474  005160' 000000          ADR:    0       ;TABLE FOR GETPA CALL
1475  005162' 000000          PA:     0
```



```
1476  005164' 000000          EA:     0
1477                          ;
1478
1479                          ;
1480                          ;
1481                          ;R1 HAS ADDR OR VECTOR
1482                          ;R0 HAS LINE # IN LOW BITS, RETURNS WITH ADDR OR VECTOR IN R1
1483  005166'                 GETADR:
1484  005166' 010046                  MOV     R0,-(R6)        ;SAVE R0
1485  005170' 042700  177760          BIC     #177760,R0      ;SAVE ONLY LINE #
1486  005174' 001404          1$:     BEQ     2$              ;DONE
1487  005176' 062701  000010          ADD     #10,R1          ;BUMP TO NEXT LINE
1488  005202' 005300                  DEC     R0              ;ADDED ENOUGH TIMES?
1489  005204' 000773                  BR      1$              ;GO BACK
1490
1491  005206' 012600          2$:     MOV     (R6)+,R0        ;RESTORE R0
1492
1493
1494  005210' 000207                  RTS     PC              ;RETURN WITH DVA IN R1
1495                          ;
1496                          ;
1497  005212' 000000          TIMTAB: 0                       ;WHEN TO TIME OUT FOR REC,LINE 0
1498  005214' 000000                  0                       ;WHEN TO TIME OUT FOR REC,LINE 1
1499  005216' 000000                  0                       ;WHEN TO TIME OUT FOR TXT,LINE 0
1500  005220' 000000                  0                       ;WHEN TO TIME OUT FO TXT,LINE 1
1501  005222' 000000          RETTAB: 0                       ;WHERE TO RETURN FOR REC LINE 0
1502
1503
1504
1505  005224' 000000                  0                       ;WHERE TO RETURN FOR REC LINE 1
1506  005226' 000000                  0                       ;WHERE TO RETURN FOR TXT LINE 0
1507  005230' 000000                  0                       ;WHERE TO RETURN FOR TXT LINE 1
1508  005232' 000000          R4TAB:  0                       ;HOLD R4 FOR RETURNS
1509
1510
1511
1512
1513  005234' 000000                  0
1514
1515  005236' 000000                  0
1516  005240' 000000                  0
1517                          ;
1518                          ;
1519  005242' 001010          MSGBF0: .BLKW   520.            ;REC BUFFER FOR LINE 0
1520
1521
1522  007262' 001010          MSGBF1: .BLKW   520.            ;REC BUFFER FOR LINE 1
1523  011302' 177777          FILL:   177777                  ;4 CHARACTERS OF FILLER BYTES
1524  011304' 177777                  177777
1525  011306' 011600'         ADROP:  MDROP
1526  011310' 177777                  -1
1527
1528
1529                          .EVEN
1530  011312' 005015  044523  041516  SUM:    .ASCII  <15><12>'SINCE LAST MSG PER LINE, IN OCTAL '<15><12>
1531  011320' 020105  040511  052123
```

```
 1532  011326' 046440  043523  050040
 1533  011334' 051105  046040  047111
 1534  011342' 026105  044440  020116
 1535  011350' 041517  040524  020114
 1536  011356' 005015
 1537  011360' 044514  042516  021440          .ASCII  'LINE #  ERRS'<15><12>
 1538  011366' 020040  051105  051522
 1539  011374' 005015
 1540  011376' 030060  026455  030060  SUM1:   .ASCII  '00--00'<15><12>
 1541  011404' 005015
 1542  011406' 030460  026455  030060          .ASCII  '01--00'<15><12>
 1543  011414' 005015
 1544  011416' 031060  026455  030060          .ASCII  '02--00'<15><12>
 1545  011424' 005015
 1546  011426' 031460  026455  030060          .ASCII  '03--00'<15><12>
 1547  011434' 005015
 1548  011436' 032060  026455  030060          .ASCII  '04--00'<15><12>
 1549  011444' 005015
 1550  011446' 032460  026455  030060          .ASCII  '05--00'<15><12>
 1551  011454' 005015
 1552  011456' 033060  026455  030060          .ASCII  '06--00'<15><12>
 1553  011464' 005015
 1554  011466' 033460  026455  030060          .ASCII  '07--00'<15><12>
 1555  011471' 005015
 1556  011476' 030061  026455  030060          .ASCII  '10--00'<15><12>
 1557  011504' 005015
 1558  011506' 030461  026455  030060          .ASCII  '11--00'<15><12>
 1559  011514' 005015
 1560  011516' 031061  026455  030060          .ASCII  '12--00'<15><12>
 1561  011524' 005015
 1562  011526' 031461  026455  030060          .ASCII  '13--00'<15><12>
 1563  011534' 005015
 1564  011536' 032061  026455  030060          .ASCII  '14--00'<15><12>
 1565  011544' 005015
 1566  011546' 032461  026455  030060          .ASCII  '15--00'<15><12>
 1567  011554' 005015
 1568  011556' 033061  026455  030060          .ASCII  '16--00'<15><12>
 1569  011564' 005015
 1570  011566' 033461  026455  030060          .ASCII  '17--00'<15><12>
 1571  011574' 005015
 1572
 1573  011576'     000                         .BYTE   0
 1574          011600'                         .EVEN
 1575                                  ;
 1576  011600' 044514  042516  053440  MDROP:  .ASCIZ  'LINE WAS DROPPED'
 1577  011606' 051501  042040  047522
 1578  011614' 050120  042105     000
 1579          000001                  .END
```



```
ABORT   005014R          903*   928*   980   1058*  1147*  1208   1222*  1423#
ACSR    000052R          469#  1063*  1114*  1225*
ADDR    000006R          448#   643    722
ADR     005160R          945*   946   1166*  1167*  1168   1474#
ADROP   011306R          808   1525#
AMSGBF  004766R          767    833    859    998   1412#
ASB     000056R          472#
ASTAT   000054R          471#  1003*  1040*  1043*  1070*  1119*  1245*
AWAS    000060R          473#
BDCNV = ******  G          1#
BEGIN   000000R          445#   624    625    678    679    692    746    808    885    886    967    971    989
                         990   1014   1046   1073   1122   1197   1201   1248   1280   1284   1342   1343   1394
                        1398
BIT0  = 000001           482#
BIT1  = 000002           482#
BIT10 = 002000           482#
BIT11 = 004000           482#
BIT12 = 010000           482#
BIT13 = 020000           482#
BIT14 = 040000           482#
BIT15 = 100000           482#
BIT2  = 000004           482#
BIT3  = 000010           482#
BIT4  = 000020           482#
BIT5  = 000040           482#
BIT6  = 000100           482#
BIT7  = 000200           482#
BIT8  = 000400           482#
BIT9  = 001000           482#
BREAK$= 104407           482#   678    679    885    886    989    990   1342   1343
BRKFLG  005120R          711*   750   1337  1341*  1344*  1457#
BP1     000012R          450#   927   1146  1269
BR2     000013R          451#
CDATA$= 104414           482#
CER     003016R         1020   1022   1038#
CHKREC  003254R          985   1059   1096#
CHKR4   005154R          707*   708*   988*   991    996*  1471#
CKMSG = ******  G        482#   986   1013
CRCMSG= ******  G        482#  1019   1021
CRCR4   005134R          699*   700*  1044*  1048   1053*  1463#
CSRA    000050R          467#  1065*  1113*  1227*
CTS     004242R          779    824    858   1264#
CTSINT  004320R         1267   1278#
CTSRET  005104R         1264*  1273   1293   1451#
CURLIN  005020R          717*   799    846    904   1006   1032   1041   1066   1084   1115   1241   1425#
DATCK$= 104417           482#
DATER$= 104405           482#
DERR    001430R          782    795#
DEVSET= ******  G        483#   626
DONE    001134R          715    738#
DROP    001474R          802    806#   849    909
DROPP4  005124R          695*   696*   807*   809    814*  1459#
DTERR4  005130R          697*   698*  1120*  1124   1129*  1461#
DVASAV  004752R          724*   788    929    979*  1063   1064*  1065   1096   1148   1181   1211   1223   1270
                        1289   1292*  1406#
DVID1   000014R          452#   622
```

```
EA      005164R          947*   949    1169*   1171    1476#
ECOUNT  000252R          552#  1034*
ENDPS$= 104402           482#  1398
END$  = 104403           482#   624     692     746
EPASS   004564R          752   1335    1363#
ERRCNT  000030R          459#
ERRLVL  000172R          491#   801     848     906    1061
ERRN$ = 104410           482#
ERROR$= 104404           482#  1046    1073    1122    1248
ERRTAB  000232R          533#
ERRWD   005030R         1429#
EXIT$ = 104400           482#   755     879     902    1210    1276    1339
FDBKR4  005150R          705*   706*    884*    887     892*   1469#
FDFLAG  005110R          757*   761*    857*    951    1173    1453#
FDRCDN  002060R          864    882#    893     907
FDTO    002222R          872    913#
FERR    001444R          773    798#
FERR1   001446R          797    799#
FILL    011302R         1523#
GERR    001670R          827    845#
GETADR  005166R          723    728    1483#
GETNXT  001014R          714#   721     792    1325
GETPAS= 104413           482#   946    1168
GWBUF$= 104412           482#
HERR    001676R          838    846#
INIT    000022R          456#
JERR    002154R          863    870     902#    913
LINACT  005012R          713*   730*    738     787*   1318    1422#
LNDONE  001400R          787#   815     842     898
LOOP    005034R          731*   784*    785     939*    840     894*    895    1087*   1431#
LPCNT   000166R          489#   785     840     895
MAP22$= 104415           482#
MASK    004746R          627*   631     633     637     639*   1484#
MD    = ******  GX       734
MDATAB  000272R          576#
MDMTAB  000312R          603#
MDROP   011600R         1525   1576#
MODNAM  000000R          446#
MODSP   000164R          457    480#
MSG   = ******  G        482#   999
MSGBF0  005242R         1412   1519#
MSGBF1  007262R         1413   1522#
MSGCNT  034744R          670*  1365*   1366    1368*   1483#
MSGHED= ******  G        482#   672     776     821     865
MSGNS = 104411           482#   808    1394
MSGSIZ= ******  G        482#   768     777     822     834     860     867    1018
MSGS$ = 104416           482#
MSG$  = 104406           482#
MSUM    004740R         1394   1400#
NXTLIN  004750R          712*   714     716     719*   1485#
OACNV = ******  G          1#
ONEREC  001310R          493    774#    804
ONETMT  001606R          494    829#
OPEN  = 000000           447    453     460     461     462     463     464     465     466     467     469     471     472
                         473    482#
PA      005162R          950   1172    1475#
```



```
PASCNT  000026R          458#
PASTIM  005122R          710*   748*   1333    1458#
PHYTAB  000212R          511#   628     635     668     712     714     741     744    1369    1392
PIRQ$ = 000004           482#   967     971    1197    1201    1280    1284
POPSP = 005726           482#
POPSP2= 022626           482#
PP      001212R          735    757#
PPFD    001720R          760    855#
PPFD1   001740R          859#   896
PPHDM   001536R          763    820#    841
PPHDM2  001650R          832    839#    850
PPHDS   001244R          766#   786
PPHDS2  001364R          775    784#    805
PRTY  = 000000           482#
PRTY0 = 000000           482#
PRTY1 = 000040           482#
PRTY2 = 000100           482#
PRTY3 = 000140           482#
PRTY4 = 000200           482#
PRTY5 = 000240           450    451     482#
PRTY6 = 000300           482#
PRTY7 = 000340           482#
PS    = 177776           482#
PSW   = 177776           482#
PUSH  = 005746           482#
PUSH2 = 024646           482#
PWRCNT= ******  G        482#   690
QUE$  = 104401           482#
RCIOR4  005140R          701*   702*   1071*   1075    1080*   1465#
RCVTO   005114R          732*   983*   1060*   1061    1081*   1455#
RDTO    003104R          914    957    1058#
REC     002234R          772    837     862     923#
RECADR  004772R          767*   833*    859*    945    1414#
RECCNT  004776R          768*   769*    834*    835*    860*    861*    930*    937    1416#
RECD    005024R         1427#
RECDN   002516R          967    971     976#   1427    1428
RECERR  003032R         1004   1041#
RECINT  002500R          925    965#
RECRTN  005040R          913*   923*    954    1023*   1025    1082    1130    1433#
RECTIM  005002R          770*   771*    836*    958    1418#
REPORT  000164R          486#  1001    1008    1028    1038    1111    1363    1366
RESTRT  000662R          474    677     690#
RETTAB  005222R          984*  1024*   1207*   1314*   1328    1331    1501#
RPET    002760R         1024#  1030    1035    1054
RSTRT   000062R          474#
RTSSET  002310R         1272   1274#   1291
R4TAB   005232R         1313*  1508#
SBADR   000052R          468#
SPARE1  000202R          496#
SPARE2  000204R          497#
SPARE3  000206R          498#
SPARE4  000210R          499#
SPOINT  000024R          457#   694
SPSIZ = 000040             1#   475
SP1     000016R          453#   640
START   000332R          456    622#
```

```
STAT    000020R          455#
SUM     011312R         1400    1530#
SUM1    011376R         1370    1540#
SVR0    000032R          460#
SVR1    000034R          461#
SVR2    000036R          462#
SVR3    000040R          463#
SVR4    000042R          464#
SVR5    000044R          465#
SVR6    000046R          466#
SYNC    000174R          492#     673     939    1162
SYNCNT  005006R         1420#
TIDONE  004000R         1197    1201    1204#
TIMCHK  004450R          740    1319    1328#   1345
TIME  = ******  G        482#    1312    1348
TIMELN  000170R          490#     770     771     780     825     836     873     874
TIMKEP  004374R          876     961    1180    1192    1310#
TIMTAB  005212R         1311*   1312*   1497#
TMT     003430R          781     826     869    1140#
TMTDON  005474R          878*    882     902*   1141*   1447#
TMTINT  003762R         1144    1195#
TMTRET  005070R         1140*   1183    1215*   1216    1256    1445#
TMTSAV  004762R          726*    783*    795*    828*    845*    897*    908*   1410#
TMTTIM  005064R          780*    825*   1189    1443#
TPX   = 000000           482#
TRAPX = 000020           482#
TSYNCT  005100R         1449#
TXADR1  005044R          776*    821*    865*   1166    1435#
TXADR2  005050R         1437#
TXCNT1  005054R          777*    778*    822*    823*    867*    868*   1149*   1151*   1158    1439#
TXCNT2  005060R         1441#
TXTO    004060R         1188    1214    1222#
TXTOR4  005144R          703*    704*   1246*   1250    1255*   1467#
VCTSAV  004756R          729*    924    1142    1265    1408#
VECTOR  000010R          449#    727
WAIT  = ******  G        482#    676
WASADR  000054R          470#
XFLAG   000005R          447#
X4      002770R         1017    1028#
X5      003000R         1002    1029    1032#   1039    1112
.     = 011621R         1519#   1522#   1574#

. ABS.  000003     000
        011621     001


% ERRORS DETECTED:  1
% DEFAULT GLOBALS GENERATED:  1

 *NQAA,NQAA/SOL/CRF:SYM=DDXCOM,NQAA
 RUN-TIME:  7 12 .1 SECONDS
 RUN-TIME RATIO: 70/20=3.4
 CORE USED:  8K  (15 PAGES)
```

```
                        .REM %

                                IDENTIFICATION
                                --------------

                        PRODUCT CODE:   MAINDEC-11-DXNPB-A-D

                        PRODUCT NAME:   DUP-11 DECNET DEC/X11 EXERCISER MODULE

                        DATE:   21-JUN-76

                        MAINTAINER:     DEC/X11 DIAGNOSTIC GROUP
```

## 1. ABSTRACT

THIS TYPE OF DEC/X11 MODULE, REFERRED TO AS DECNET DEC/X11 MODULE, WILL DETECT SYSTEM INTERACTION ERRORS WHEN RUN IN CONJUNCTION WITH OTHER DEC/X11 MODULES EXERCISING OTHER DEVICES ON THE SYSTEM. ERRORS ARE REPORTED ON THE SYSTEM CONSOLE AS THEY OCCUR OR AS A SUMMARY REPORT, AT THE OPERATORS OPTION. IN ADDITION, THESE MODULES MAY BE USED TO VERIFY THE CONDITION OF THE ASSOCIATED MODEMS AND COMMUNICATION LINKS. IT WILL ONLY TEST COMMUNICATIONS PATHS CAPABLE OF RUNNING DDCMP. THIS MEANS THE DEVICES MUST USE NO PARITY, 8 LEVEL CODE, AND TERMINALS CANNOT BE TESTED.

NOTE

THIS DOCUMENT ASSUMES FAMILIARITY WITH THE PHILOSOPHY AND OPERATION OF DEC/X11. THE DETAILS OF LINKING, LOADING, AND RUNNING MODULES UNDER DEC/X11 CAN BE FOUND IN THE DEC/X11 USERS DOCUMENTATION AND REFERENCE GUIDE, MAINDEC-11-DXQBA. THE READER SHOULD ALSO HAVE READ THE DECNET DEC/X11 USER'S GUIDE, MAINDEC-11-DXQBC, WITHOUT THROUGHLY UNDERSTANDING THAT DOCUMENT, THERE IS LITTLE HOPE OF SUCCESSFUL TESTING.

## 2. HARDWARE REQUIREMENTS

THESE MODULES REQUIRE A COMPLETE PATH FROM A COMMUNICATIONS DEVICE TO A COMMUNICATIONS DEVICE. THIS MAY BE ACCOMPLISHED THROUGH A VARIETY OF METHODS:

A. 1 PROCESSOR, 1 COMM. DEVICE, AND A LOOP BACK DEVICE WITH MODEM SIMULATION CAPABILITIES. (MUST RUN FULL DUPLEX)

B. 1 PROCESSOR, 2 COMM. DEVICES, 2 MODEMS

C. 2 PROCESSORS, 2 COMM. DEVICES, 2 MODEMS

UP TO SIXTEEN LINES CAN BE HANDLED BY ONE MODULE BUT ONLY 2 LINES ARE KEPT RUNNING SIMULTANEOUSLY.

## 3. SOFTWARE REQUIREMENTS

A. THE DEC/X11 MONITOR MUST BE QABK OR A LATER REVISION.

B. THE N1A? MODULE MUST BE CONFIGURED DIRECTLY AFTER THE MONITOR*

C. ALL NETWORK MODULES MUST BE CONFIGURED NEXT.

D. THE N2A? MODULE MUST BE CONFIGURED NEXT.*

E. ONE OF THE 3 CLOCK MODULES MUST BE CONFIGURED NEXT. EITHER KWA?, KWB?, OR KWX?

F. ANY OTHER DESIRED DEC/X11 MODULES CAN THEN BE CONFIGURED IN ANY ORDER.

4. PRELIMINARY TESTING

THE APPROPRIATE STAND ALONE DIAGNOSTICS MUST BE RUN TO VERIFY THE CPU, AND THE COMMUNICATIONS DEVICE. THEN A DEC/X11 EXERCISER USING THE CONVENTIONAL (MAINTENANCE MODE) DEC/X11 MODULES FOR THE COMMUNICATIONS DEVICES MUST BE RUN. IF BOTH THESE STEPS PASS SUCCESSFULLY, DECNET DEC/X11 MAY BE ATTEMPTED.

5. PROGRAMMING CONSIDERATIONS

A. OTHER THAN THE REQUIREMENTS MENTIONED IN 3 REGARDING THE ORDER IN WHICH MODULES MUST BE REQUIRED, THESE DECNET DEC/X11 MODULES WILL BE COMPLETELY COMPATIBLE WITH THE DEC/X11 MONITORS AND ALL OTHER MODULES. THE ONLY OTHER EXECPTION IS THAT FOR A GIVEN COMMUNICATIONS DEVICE, THERE WILL NOW BE 2 DIFFERENT DEC/X11 MODULES CAPABLE OF BEING RUN, THE OLDER MAINTANCE MODE TYPE REQUIRING NO CONNECTIONS TO THE COMMUNICATION DEVICE, AND THE NEW DECNET TYPE REQUIRING A COMPLETED COMMUNICATIONS PATH. IT WILL BE POSSIBLE (AND SOMETIMES DESIREABLE) TO CONFIGURE 2 DIFFERENT DEC/X11 MODULES IN ONE RUN TIME EXERCISER WHICH TEST THE SAME PHYSICAL DEVICE. IF THIS IS DONE, ONLY ONE OF THE 2 MODULES MAY BE SELECTED FOR THE SAME RUN.

B. USING THESE DECNET DEC/X11 MODULES, IT IS POSSIBLE TO HAVE MANY NODES CONNECTED IN A NETWORK, ALL RUNNING DEC/X11 TOGETHER. ALL THESE EXERCISERS WOULD HAVE BEEN STARTED AT DIFFERENT TIMES. DUE TO DIFFERENT HARDWARE ON EACH SYSTEM, RUN TIMES AND END OF PASS TIMES MIGHT VARY CONSIDERABLY. THEREFORE IT IS NECCESSARY THAT THESE DECNET DEC/X11 MODULES BE EXTREMELY PATIENT AND HAVE A HIGH RETRY CAPABILITY IN ALL OPERATIONS. THIS HOWEVER, MEANS IT SOMETIMES MAY TAKE A LONG TIME (ON THE ORDER OF 10 MINUTES) BEFORE AN OPERATOR FINDS OUT A LINE IS DEFINITELY NOT RUNNING.

C. DUE TO THE EXCESSIVE PROGRAMMING COMPLICATIONS AND LARGE

---

* THESE ARE DESCRIBED IN THE DECNET DEC/X11 USER'S GUIDE.

BUFFERS REQUIRED TO RUN MANY LINES FROM ONE MODULE ALL AT THE SAME TIME, IT WAS DECIDED TO ONLY RUN 2 AT ANY ONE TIME. IF THE MODULE IS SET TO RUN 16 LINES, IT WILL RUN 1 PAIR FOR THE NUMBER OF TRANSFERS INDICATED BY LOCATION LPCNT, THEN IT WILL RUN A 2ND PAIR, THEN 3RD, ETC UP TO THE 8TH PAIR. IT WILL THEN REPORT AN END OF PASS AND GO BACK TO THE FIRST PAIR, THEN 2ND, ETC. THIS FORCES SOME CONSIDERATIONS TO BE MADE WHEN CONNECTING LINES TOGETHER SO THAT A LINE IN THE ACTIVE MODE IS NOT WAITING ON A LINE IN THE DORMANT MODE. PLEASE READ THE DECNET DEC/X11 USER'S GUIDE, ESPECIALLY THE SECTION ON 2 LINES AT A TIME CONSIDERATIONS.

D. THESE MODULES WILL, AS A GENERAL RULE, USE THE MINIMUM AMOUNT OF FEATURES OF A COMMUNICATION DEVICE REQUIRED TO SEND AND RECEIVE MESSAGES. THIS MEANS THAT CRC CALCULATING HARDWARE, CHARACTER RECOGNITION HARDWARE, ETC, WILL NOT BE USED. THESE FEATURES ARE FULLY TESTED BY STAND ALONE DIAGNOSTICS.

## 6. OPERATING PROCEDURES

A. THE MODULE IS CONFIGURED, LOADED, AND RUN AS SPECIFIED IN THE DEC/X11 REFERENCE GUIDE.

B. CONSOLE SWITCHES ARE STANDARD FOR DEC/X11

C. THE N1A? MODULE WILL SOLICIT FROM THE OPERATOR WHICH TYPE OF MESSAGE HE WANTS TO USE FOR DECNET ACTIVITY. THE OPTIONS INCLUDE:

- ALL ONES
- ALL ZEROS
- ALTERNATING ONES AND ZEROS
- A DIGIT (4 BIT) REPEATING COUNT PATTERN
- A PRECANNED ASCII MESSAGE
- USER SPECIFIED BY TYPING IN

D. THE N2A? MODULE WILL MAKE ANY AUTOMATIC PHONE CONNECTIONS USING ANY DN-11'S PRESENT AND WILL REQUEST THE OPERATOR TO DIAL ALL MANUAL CONNECTIONS. ALL OPERATORS AT EACH NODE UNDER TEST MUST REACH THIS POINT BEFORE ANY ONE NODE CONTINUES INTO THE EXERCISING PHASE.

E. THE N2A? MODULE WILL REQUEST A CARRIAGE RETURN WHEN THE OPERATOR WANTS TO BEGIN EXERCISING HIS NODE.

F. THERE ARE MANY OPTIONS WHICH MAY BE ALTERED BEFORE RUN TIME, THEY INCLUDE:

(1) LOCATION 164 IS A SWITCH WHICH INDICATES WHETHER OR NOT ALL ERRORS SHOULD BE REPORTED AS THEY HAPPEN. IF LOCATION 164 IS NON-ZERO (EQUAL TO N), THEN ERRORS ARE TOTALLED (ON A LINE BY LINE BASIS); EVERY N PASSES THIS

TOTAL NUMBER OF ERRORS FOR THE PREVIOUS N PASSES WILL BE REPORTED. THE VALUE OF N IS THE VALUE OF LOCATION 164. THE MAXIMUM NUMBER OF ERRORS THAT CAN BE TALLEYED IS 256 PER LINE. THEREFORE IF A LINE IS NOT KNOWN TO BE OF VERY GOOD QUALITY, THE VALUE OF LOCATION 164 TIMES THE VALUE OF LOCATION 166 SHOULD BE LESS THAN 128. (LOCATION 166 IS DEFINED NEXT.) THIS MEANS THE NUMBER OF MESSAGES PER PASS TIMES THE NUMBER OF PASSES PER SUMMARY REPORT SHOULD BE LESS THAN 128. IN THIS WAY, IF EVERY MESSAGE ATTEMPTED HAS AN ERROR ON BOTH TRANSMIT AND RECEIVE, THE ERROR TALLY WILL NOT OVERFLOW. IF THE LINE IS OF KNOWN GOOD QUALITY, THIS RESTRICTION MAY BE RELAXED. THE DEFAULT VALUE IS TO NOT REPORT TOTAL ERRORS.(LOC 164=0) OF COURSE DEC/X11 KEEPS TOTALS FOR THE ENTIRE RUN, REGARDLESS.

(2) LOCATION 166 INDICATES HOW MANY ITERATIONS SHOULD BE DONE BEFORE AN END OF PASS IS CALLED. AN ITERATION IS NORMALLY ONE TRANSMIT AND ONE RECEIVE, IN EITHER ORDER. THE DEFAULT VALUE IS 8.

(3) LOCATION 170 IS A TIME SCALING FACTOR. THERE ARE MANY TIME OUT LOOPS IN THE MODULES, SOME SHOULD LOGICALLY BE LONGER THAN OTHERS. THE VALUE IN LOCATION 170 IS THE NUMBER OF SECONDS TO WAIT FOR THE SMALLEST TIME OUTS. SOME TIME OUTS WILL BE A MULTIPLE OF THIS FACTOR. DEFAULT IS 30 SECONDS.

(4) LOCATION 172 IS AN ERROR TOLERATION LEVEL, ON A SINGLE MESSAGE BASIS. SO IF IN AN ATTEMPT TO DO ONE ITERATION (USUALLY A TRANSMIT THEN RECIEVE OR VICE-VERSA) THE NUMBER OF ERRORS REACHES THE VALUE IN LOCATION 172, THAT LINE IS DROPPED FROM TESTING. IF BEFORE THIS LEVEL IS REACHED, AN ITERATION IS CORRECTLY COMPLETED, THEN THE ERROR COUNT WHICH IS COMPARED TO LOCATION 172 IS RESET TO ZERO. THE DEFAULT IS 10.

(5) LOCATION 174 CONTAINS 2 IDENTICAL BYTES FOR USE AS SYNC CHARACTERS. THE DEFAULT IS 113226 WHICH IS 2 BYTES OF 226. IF THIS IS CHANGED, BOTH BYTES MUST BE IDENTICAL TO EACH OTHER. THIS LOCATION IS A DON'T CARE FOR ASYNCHRONOUS DEVICES, BECAUSE THEY "SYNC" ON THE FIRST BYTE OF THE ACTUAL MESSAGE, NOT ON PRECEDING CHARACTERS AS SYNCHRONOUS DEVICES DO.

(6) LOCATION 176 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN RECEIVE-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, SLAVE. (BIT 7 = 1, BITS 4,5, AND 6 = 0'S IN THE PHYSICAL LINE TABLE. SEE ITEM 6-F-9 BELOW). THEN PATCH THE LOCATION POINTED TO BY LOCATION 176 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, SLAVE BY THIS MODULE TO RUN IN RECEIVE ONLY MODE. THIS LOCATION MAY BE PATCHED BACK TO A 401 TO RESUME NORMAL REC-TMT TESTING.

(7) LOCATION 200 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN TRANSMIT-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, MASTER (BITS 7 AND 5 = 1'S, BITS 4 AND 6 = 0'S IN THE PHYSICAL-LINE TABLE. SEE ITEM 6-F-9 BELOW.) THEN PATCH THE LOCATION POINTED TO BY LOCATION 200 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, MASTER BY THIS MODULE TO RUN IN TRANSMIT ONLY MODE. THE LOCATION MAY BE PATCHED BACK TO 401 TO RESUME NORMAL TMT-REC TESTING.

G. LOCATIONS DVA, VCT, BR1 AND BR2 MUST BE SET TO PROPERLY MATCH THE FIRST DEVICE TO BE TESTED BY THIS MODULE.

## 7. ERROR REPORTING

A. ERRORS ARE EITHER TOTALLED AND PRINTED AS SUMMARIES ON A LINE BY LINE BASIS, OR THEY ARE PRINTED AS THEY OCCUR. SEE SECTION 6F1. THEY ARE 2 CLASSES OF ERRORS DATA ERRORS AND OTHER ERRORS.

B. DATA ERRORS ARE ACTUALLY REPORTED BY THE N1A? MODULE BECAUSE THIS IS WHERE THE SHARED DATA COMPARE ERROR ROUTINE IS LOCATED. SO ALL DECNET DEC/X11 MODULE DATA COMPARE ERRORS LOOK LIKE THE STANDARD DEC/X11 DATA COMPARE ERROR MESSAGE EXCEPT:

(1) THE MODULE NAME TYPED IS FFXX, WHERE FF IS FILLED IN TO BE THE NAME OF THE DECNET DEC/X11 MODULE WHICH CALLED THE N1A? MODULE TO DO THE COMPARE. IN OTHER WORDS FF REPRESENTS THE MODULE THAT "HAD" THE ERROR.

(2) THE ERROR NUMBER IS A COUNT OF THE DATA COMPARE ERRORS IN THE CURRENT MESSAGE BEING CHECKED, IT IS THEREFORE RESET TO ZERO EVERY TIME A NEW MESSAGE IS TO BE CHECKED.

(3) ACSR CONTAINS THE ADDRESS OF THE DECNET DEC/X11 MODULE WHICH CALLED THIS ROUTINE TO HAVE ITS MESSAGE CHECKED.

(4) SBADR CONTAINS THE LINE # OF THE DEVICE WHICH HAD THE DATA ERROR.

(5) WASADR CONTAINS THE COUNT (IN OCTAL) OF WHICH CHARACTER IN THE MESSAGE THIS BAD CHARACTER WAS.

(6) THE "SHOULD BE AND "WAS" ITEMS ARE NORMAL.

C. OTHER ERRORS ARE REPORTED USING THE NORMAL ERROR CALL IN

DEC/X11 CSRA AND ACSR ARE STANDARD. STATC CONTAINS A CODED

ERROR NUMBER, WITH THE CODE BEING CONSISTENT IN ALL DECNET DEC/X11 MODULES. THE LOW ORDER 2 DIGITS OF STATC CONTAIN WHICH LINE HAD THE ERROR THE OTHER DIGITS CONTAIN THIS CODE:

| | |
|---|---|
| 0010XX | REC TIME OUT |
| 0020XX | TMT CLOCK LOSS |
| 0030XX | HEADER CRC ERROR |
| 0040XX | MSG CRC ERROR |
| 0050XX | TMT TIME OUT |
| 0060XX | TMT NON-EXISTENT MEMORY |
| 0070XX | TMT LATENCY |
| 0100XX | REC DATA PARITY ERROR |
| 0110XX | REC NON-EXISTENT MEMORY |
| 0120XX | REC CLOCK LOSS |
| 0200XX | REC FRAMING ERROR |
| 0400XX | REC OVERRUN OR LATENCY ERROR |

THIS METHOD SHOULD ELIMINATE THE CONSTANT NEED FOR HAVING A LISTING ON HAND TO DECIDE WHAT THE ERRORS WHERE. THE FIRST TIME THE RUN TIME EXERCISER IS RUN AFTER LOADING, THE N1A? MODULE WILL TYPE THIS ERROR CODE LIST OUT TO THE SYSTEM CONSOLE FOR THE OPERATORS SUBSEQUENT USE.

D. ALL ERRORS HAVE A RETRY LIMIT (SEE SECTION 6F4) WHICH CAN BE MODIFIED.

E. ON POWER-FAIL, POWER-UP CONDITIONS, DECNET DEC/X11 MODULES WILL DROP THEMSELVES. THIS IS DONE BECAUSE THE PHONE CONNECTIONS WILL HAVE BEEN LOST AND OPERATOR INTERVENTION WILL BE REQUIRED.

8. TEST SEQUENCE

LINES ARE TESTED BY PAIRS IN SEQUENTIAL ORDER. THIS ORDER MAY BE MODIFIED BY MODIFYING THE PHYSICAL LINE TABLE. (SEE DECNET DEC/X11 USER'R GUIDE). THERE ARE SEVERAL TESTING OPTIONS WITH SLIGHTLY DIFFERENT FUNCTIONS MAKING UP AN ITERATION:

A. POINT-TO-POINT SLAVE. IN THIS CASE THE LINE WAITS TO RECEIVE A MESSAGE, WHEN IT DOES, IT CHECKS IT, AND THEN TRANSMITS A MESSAGE BACK. IT WILL REPORT ERRORS ON EXCESSIVE WAITS.

B. POINT-TO-POINT MASTER. IN THIS CASE THE LINE SENDS A MESSAGE AND WAITS TO RECEIVE ONE IN RETURN. IT THEN CHECKS THE RECEIVED MESSAGE. IF IT TIMES-OUT, IT TRANSMITS AGAIN, IT WILL CONTINUE TO TIME OUT AND RE-TRANSMIT UNTIL EITHER A MESSAGE IS RECEIVED, OR IT TIMES OUT COUNT EQUALS LOCATION

172, THE ERROR TOLERATION LEVEL. (SEE SECTION 6F4). IT WILL THEN REPORT AN ERROR AND BEGIN AGAIN. AFTER THE NUMBER OF

CONSECUTIVE TIME-OUT ERRORS REPORTED EQUALS LOCATION 172, THE LINE WILL BE DROPPED.

C. POINT-TO-POINT SLAVE RECEIVE ONLY. IN THIS MODE THE LINE WAITS ONLY TO RECEIVE, CHECKS THE MESSAGE, AND WAITS TO RECEIVE AGAIN.

D. POINT-TO-POINT MASTER-TRANSMIT ONLY. IN THIS MODE THE LINE JUST TPANSMITS AND THEN TRANSMITS AGAIN ,ETC.

E. MULTI-DROP MASTER. IN THIS MODE, THE LINE WILL SEND A DDCMP MESSAGE OUT FOR THE ADDRESS CONTAINED IN THE MULTI-DROP MENU TABLE, WAIT TO RECEIVE A MESSAGE BACK, AND THEN CHECK THE DATA.
* NOT YET IMPLEMENTED

F. MULTI-DROP SLAVE. IN THIS MODE, THE LINE WILL WAIT FOR A MESSAGE WITH THE MATCHING ADDRESS. THEN IT WILL TRANSMIT A MESSAGE.
* NOT YET IMPLEMENTED

G. THESE BASIC ITERATIONS ARE REPEATED UP TO THE NUMBER OF TIMES IN LOCATION 166. (SEE SECTION 6F2) FOR EACH LINE SELECTED FOR TEST, AND THE END-OF-PASS IS DECLARED.

9. PASS TIMES

PASS TIMES ARE EXTREMELY CONFIGURATION DEPENDENT. DIFFERENT BAUD RATES, MESSAGE SIZES, AND TESTING MODES ALL GREATLY AFFECT THE PASS TIME. BELOW ARE SOME BALL PARK TIMES, BUT ANY ONE SYSTEM COULD VARY TREMENDOUSLY FROM THESE:
-RUNNING ALONE, 1 LINE ON AN 11/05 AT 9600 BAUD,HALF DUPLEX, POINT-TO-POINT, WITH A 512 CHARACTER MESSAGE, TAKES 15 SECONDS.

NOTE 1 AT VARIOUS MONITOR CALLS THROUGHOUT THIS MODULE, IT IS LIKELY THAT WHEN 2 LINES ARE RUNNING, A MONITOR CALL CAN BE MADE AND BEFORE CONTORL IS RETURNED TO THE MODULE, THE OTHER LINE MAY CAUSE THE EXACT SAME CALL TO BE MADE. IN FACT, WHEN RUNNING 2 LINES FULL DUPLEX, IT MAY BE POSSIBLE TO HAVE 4 CALLS TO THE MONITOR FROM ONE SPOT IN THE MODULE, BEFORE THE FIRST CALL IS RETURNED FROM. THIS IS WHY AT SUCH CALLS, THE CODE SETS COUNTERS BEFORE DOING CALLS, SO THAT WHEN IT IS RETURNED TO, IT CAN KEEP WHICH LINE DID WHAT STRAIGHT.

```
                        NOTE 2  WHEN RUNNING 2 LINES LOOPED TOGETHER FROM THE SAME MODULE
                        THE LOW ORDER ONE (PER DVC, LOC,14) SHOULD BE SLAVE AND THE HIGH ORDER
                        ONE SHOULD BE MASTER.
                        %
```

```
                                        IOMOD <NPBA >, 1,4,5,5
                                        MODULE  140000,NPBA ,1,4,5,5,
                                        .TITLE  NPBA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  4    9/6/75
                                        .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 050116  040502      040 MODNAM: .ASCII  /NPBA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000001                  ADDR:   1+0             ;1ST DEVICE ADDR.
000010' 000004                  VECTOR: 4+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' 000332'                 INIT:   START           ;MODULE START ADDR.
000024' 000164'                 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.
000062' 000540'                 RSTRT:  RESTRT          ;RESTART ADDRESS AFTER END OF PASS
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000164'                         MODSP:
                                ;*****************************************************************************
                                        .GLOBL  WAIT, TIME, MSGSIZ, CKMSG, CRCMSG,MSGHED, MSG,PWRCNT
                                        .GLOBL  DEVSET
                                ;
                                ;
000164' 000000                  REPORT: 0               ;FLAG, IF =#, THEN ERRORS ON RECEIVED MSG'S
                                                        ;ARE TOTALLED AND PRINTED EVERY # PASSED
                                                        ;IF =0, ERRORS ARE ALL REPORTED AS THEY HAPPEN.
000166' 000010                  LPCNT:  10              ;ITERATIONS PER END PASS CALL
000170' 000036                  TIMELN: 30.             ;TIME OUT FACTOR, IN SECONDS
000172' 000012                  ERRLVL: 10.             ;NUMBER OF ERRORS TOLERATED
000174' 113226                  SYNC:   113226          ;2 BYTES OF SYNC CHARACTER
000176' 001156'                         ONEREC          ;ADDRESS OF WHERE 240 GOES
000200' 001500'                         ONETMT          ;ADDRESS OF WHERE 240 GOES
```



```
000202' 000000                  SPARE1: 0
000204' 000000                  SPARE2: 0
000206' 000000                  SPARE3: 0
000210' 000000                  SPARE4: 0
                                ;
                                ;
                                ;PHYSICAL LINE TABLE. BYTE 0 CONTAINS THE PHYSICAL
                                ;LINE # FOR THE FIRST LINE TO BE RUN, BYTE 1 WILL BE
                                ;THE PHYSICAL LINE RUN SECOND, ETC.
                                ;EACH BYTE IN THE TABLE HAS THE FOLLOWING BIT MEANINGS:
                                ;BIT 0-3        PHYSICAL LINE #
                                ;BIT 4          0 FOR POINT-TO POINT, 1 FOR MULTI-DROP
                                ;BIT 5          0 FOR SLAVE, 1 FOR MASTER (DERIVED FROM SR1)
                                ;BIT 6          0 FOR HALF DUPLEX, 1 FOR FULL
                                ;BIT 7          0 DO NOT TEST, 1 TEST (DERIVED FROM DVID1)
000212'     000                 PHYTAB: .BYTE   0
000213'     001                         .BYTE   1
000214'     002                         .BYTE   2
000215'     003                         .BYTE   3
000216'     004                         .BYTE   4
000217'     005                         .BYTE   5
000220'     006                         .BYTE   6
000221'     007                         .BYTE   7
000222'     010                         .BYTE   10
000223'     011                         .BYTE   11
000224'     012                         .BYTE   12
000225'     013                         .BYTE   13
000226'     014                         .BYTE   14
000227'     015                         .BYTE   15
000230'     016                         .BYTE   16
000231'     017                         .BYTE   17

                                ;
                                ;
                                ;ERRTAB ,KEEPS TRACK OF RETRY ERROR COUNTS FOR
                                ;EACH LINE
                                ;
000232'     000                 ERRTAB: .BYTE   0
000233'     000                         .BYTE   0
000234'     000                         .BYTE   0
000235'     000                         .BYTE   0
000236'     000                         .BYTE   0
000237'     000                         .BYTE   0
000240'     000                         .BYTE   0
000241'     000                         .BYTE   0
000242'     000                         .BYTE   0
000243'     000                         .BYTE   0
000244'     000                         .BYTE   0
000245'     000                         .BYTE   0
000246'     000                         .BYTE   0
000247'     000                         .BYTE   0
000250'     000                         .BYTE   0
000251'     000                         .BYTE   0
                                ;
                                ;
                                ;ECOUNT TABLE KEEPS TRACK OF # OF BAD MSG'S
```

```
000252'  000                ECOUNT: .BYTE   0
000253'  000                        .BYTE   0
000254'  000                        .BYTE   0
000255'  000                        .BYTE   0
000256'  000                        .BYTE   0
000257'  000                        .BYTE   0
000260'  000                        .BYTE   0
000261'  000                        .BYTE   0
000262'  000                        .BYTE   0
000263'  000                        .BYTE   0
000264'  000                        .BYTE   0
000265'  000                        .BYTE   0
000266'  000                        .BYTE   0
000267'  000                        .BYTE   0
000270'  000                        .BYTE   0
000271'  000                        .BYTE   0
                              ;
                              ;
                              ;MULTI-DROP STATION ADDRESS TABLE
                              ;BYTE 0 IS THE MULTI-DROP STATION ADDRESS
                              ;FOR LOGICAL LINE 0, BYTE 1 IS FOR LOGICAL LINE 1, ETC.
                              ;THIS ENTRY ONLY HAS MEANING IF CORRESPONDING
                              ;BYTE IN PHYSICAL LINE TABLE HAS BITS 4 AND 7 SET
                              ;AND BIT 5 OFF, IE MULTI-DROP SLAVE TO BE TESTED
000272'  000                MDATAB: .BYTE   0
000273'  001                        .BYTE   1
000274'  002                        .BYTE   2
000275'  003                        .BYTE   3
000276'  004                        .BYTE   4
000277'  005                        .BYTE   5
000300'  006                        .BYTE   6
000301'  007                        .BYTE   7
000302'  010                        .BYTE   10
000303'  011                        .BYTE   11
000304'  012                        .BYTE   12
000305'  013                        .BYTE   13
000306'  014                        .BYTE   14
000307'  015                        .BYTE   15
000310'  016                       .BYTE   16
000311'  017                       .BYTE   17
                              ;
                              ;MULTI-DROP STATION ADDRESS MENU TABLE
                              ;THIS TABLE INDICATES TO ANY LINE RUNNING AS
                              ;MULTI-DROP MASTER, WHICH STATION ADDRESS TO
                              ;ATTEMPT TO EXEPCISE ON THE MULTI-DROP LINE
                              ;IT ALLOWS UP TO 15 DIFFERENT STATIONS TO BE
                              ;TESTED ON A LINE.  EACH BYTE WHICH IS NOT=-1
                              ;INDICATES A STATION ADDRESS TO BE TESTED, EACH
                              ;STATION IS TESTED, BY THE ORDER OF THE TABLE,
                              ;UNTIL THE FIRST OCCURANCE OF A -1 (BINARY 11111111)
                              ;
000312'  377                MDMTAB: .BYTE   -1
000313'  377                        .BYTE   -1
000314'  377                        .BYTE   -1
000315'  377                        .BYTE   -1
000316'  377                        .BYTE   -1
```



```
000317'  377                        .BYTE   -1
000320'  377                        .BYTE   -1
000321'  377                        .BYTE   -1
000322'  377                        .BYTE   -1
000323'  377                        .BYTE   -1
000324'  377                        .BYTE   -1
000325'  377                        .BYTE   -1
000326'  377                        .BYTE   -1
000327'  377                        .BYTE   -1
000330'  377                        .BYTE   -1
000331'  377                        .BYTE   -1
                              ;
                              ;
000332' 016700  177456      START:  MOV     DVID1,R0        ;GET DEVICES TO BE TESTED
000336' 001032                      BNE     1$              ;ANY ACTIVE?
000340' 104403  000000'             END$,BEGIN              ;
000344' 012705  000000'     1$:     MOV     #BEGIN,R5       ;PREPARE FOR NXT JSR
000350' 004567  000000G             JSR     R5,DEVSET       ;GO ASK FOR PARAMETERS
000354' 012767  000200  004276      MOV     #200,MASK       ;SET UP MASK
000362' 012701  000232'     2$:     MOV     #PHYTAB+16.,R1  ;GET END OF DEVICE TABLE
000366' 006100              3$:     ASL     R0              ;CHECK EACH BIT
000370' 103003                      BCC     4$              ;BR IF NOT ON
000372' 156741  004262              BISB    MASK,-(R1)      ;SET THIS LINE ACTIVE
000376' 000402                      BR      5$
000400' 146741  004254      4$:     BICB    MASK,-(R1)      ;CLEAR THIS LINE
000404' 105061  000040      5$:     CLRB    32.(R1)         ;CLEAR ECOUNT TABLE
000410' 020127  000211'             CMP     R1,#PHYTAB-1    ;DONE?
000414' 001364                      BNE     3$              ;BR IF NOT
000416' 026727  004236  000040      CMP     MASK,#40        ;BEEN HERE BEFORE?
000424' 001406                      BEQ     6$              ;BR IF YES
000426' 012767  000040  004224      MOV     #40,MASK        ;SET UP MASK
000434' 016700  177356              MOV     SR1,R0          ;GET SLAVE/MASTER INFO
000440' 000750                      BR      2$              ;GO SET UP
000442' 005201              6$:     INC     R1              ;GET PHYTAB ADDR
000444' 016700  177336              MOV     ADDR,R0         ;GET DEVICE ADDR
000450' 105721              7$:     TSTB    (R1)+           ;DEVICE ACTIVE?
000452' 100013                      BPL     8$              ;BR IF NOT
000454' 012760  000400  000004      MOV     #400,4(R0)      ;RESET DEVICE.
000462' 012710  000002              MOV     #2,(R0)         ;SET DATA TERM READY
000466' 112767  000202  177501      MOVB    #202,SYNC+1     ;SET BITS FOR DEC MODE AND
                                                            ;INHIBIT AUTO-CRC
000474' 016760  177474  000002      MOV     SYNC,2(R0)      ;WRITE PARAMETER CONTROL REGISTER
000502' 062700  000010      8$:     ADD     #10,R0          ;BUMP R0
000506' 020127  000232'             CMP     R1,#PHYTAB+16.  ;DONE?
000512' 001356                      BNE     7$              ;BR IF NOT
000514' 005067  004136      9$:     CLR     MSGCNT          ;COUNTS WHEN TO DO SUMMARY
000520' 005767  000000G             TST     WAIT            ;LINES SET UP?
000524' 001405                      BEQ     RESTRT          ;BR IF YES
000526' 104107  000000'             BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
000532' 104407  0C0000'             BREAK$,REGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000536' 000766                      BR      9$              ;WAIT TILL WAIT IS CLEARED BY NET2
                              ;
                              ;
                              ;THRUUGH-OUT THE PROGRAM MANY LOCATIONS ARE REFERENCED
                              ;OFF OF (R4).  THIS IS A MEANS OF KEEPING TWO SETS OF
                              ;VARIABLES, ONE FOR EACH OF THE ACTIVE LINES.
```

```
                                        ;R4 ALWAYS CONTAINS A 0 IF THE PROGRAM IS CURRENTLY
                                        ;REFERENCING LINE 0 OR A 2 IF IT IS REFERENCING
                                        ;LINE 1.
                                        ;
000540' 005767  000000G         RESTRT: TST     PWRCNT          ;HAS THERE BEEN A POWERFAIL
000544' 001402                          BEQ     1$              ;CONTINUE IF NOT,
000546' 104403  000000'                 END$,BEGIN              ;
000552' 005004                  1$:     CLR     R4              ;FIRST ACTIVE LINE
000554' 016706  177244                  MOV     $POINT,R6       ;INIT STACK
000560' 005067  004246                  CLR     DROPR4          ;CLEAR ALL EXIT FLAGS
000564' 005067  004244                  CLR     DROPR4+2
000570' 005067  004242                  CLR     DTERR4
000574' 005067  004240                  CLR     DTERR4+2
000600' 005067  004236                  CLR     CRCR4
000604' 005067  004234                  CLR     CRCR4+2
000610' 005067  004232                  CLR     RCTOR4
000614' 005067  004230                  CLR     RCTOR4+2
000620' 005067  004226                  CLR     TXTOR4
000624' 005067  004224                  CLR     TXTOR4+2
000630' 005067  004222                  CLR     FDBKR4
000634' 005067  004220                  CLR     FDBKR4+2
000640' 005067  004216                  CLR     CHKR4
000644' 005067  004214                  CLR     CHKR4+2
                                        
000650' 005067  004154                  CLR     PASTIM          ;CLR PASS TIME FLAG
000654' 005067  004146                  CLR     BRKFLG          ;CLR FLAG THAT SAYS WE'RE IN BREAK
000660' 012767  000212' 003774          MOV     #PHYTAB,NXTLIN  ;POINTS TO NEXT LINE TO TEST
000666' 005067  004026                  CLR     LINACT          ;SHOW NO LINES ACTIVE
000672' 026727  003764  000232' GETNXT: CMP     NXTLIN,#PHYTAB+16.;DONE ALL LINES?
000700' 001440                          BEQ     DONE            ;BR IF SO
000702' 016700  003754                  MOV     NXTLIN,R0       ;GET ADDR OF NEXT LINE
000706' 010064  004726'                 MOV     R0,CURLIN(R4)   ;SAVE IT
000712' 105060  000020                  CLRB    20(R0)          ;CLEAR RETRY COUNT FOR THIS LINE
000716' 005267  003740                  INC     NXTLIN          ;BUMP FOR NEXT TIME
000722' 111000                          MOVB    (R0),R0         ;GET PHYSICAL LINE BYTE
000724' 002362                          BGE     GETNXT          ;BR BACK IF BIT 7 NOT ON
000726' 016701  177054                  MOV     ADDR,R1         ;SET UP TO CALC ADDR
000732' 004767  004130                  JSR     PC,GETADR       ;GO CALC ADDR
000736' 010164  004664'                 MOV     R1,DVASAV(R4)   ;SAVE DVA FOR THIS LINE
000742' 016701  177042                  MOV     VECTOR,R1               ;SET UP TO CALC VECTOR ADDR
000746' 004767  004114                  JSR     PC,GETADR       ;GO CALC
000752' 010164  004670'                 MOV     R1,VCTSAV(R4)   ;SAVE IT
000756' 005267  003736                  INC     LINACT          ;SHOW A LINE'S RUNNING
000762' 005064  004742'                 CLR     LOOP(R4)        ;CLEAR ITERATION COUNT
000766' 005364  005022'                 CLR     RCVTO(R4)       ;CLR REC TIME OUT COUNT
000772' 032700  000020                  BIT     #20,R0          ;MULTI DROP OR NOT?
A 000776' 001377                        BNE     MD              ;BR IF IT IS
001000' 000427                          BR      PP              ;BR IF NOT TO P TO P
                                ;
                                ;
001002' 005767  003712          DONE:   TST     LINACT          ;ANY LINES RUNNING?
001006' 001402                          BEQ     1$              ;BR IF NO
001010' 000167  003346                  JMP     TIMCHK          ;IF YES GO CHECK FOR TIME OUT
001014' 012701  000212'         1$:     MOV     #PHYTAB,R1      ;MUST CHECK IF MAYBE ALL LINES DROPPED
001020' 105721                  2$:     TSTB    (R1)+           ;IS THIS LINE STILL SELECTED?
001022' 100405                          BMI     3$              ;BR OUT IF YES
```



```
001024' 022701  000232'                 CMP     #PHYTAB+16.,R1  ;CHECKED ALL 16?
001030' 001373                          BNE     2$              ;BR BACK IF NO
001032' 104403  000000'                 END$,BEGIN              ;
                                ;
001036' 012767  000001  003764  3$:     MOV     #1,PASTIM       ;SHOW IT IS TIME FOR ENDPAS
                                                                ;FOR BREAK LOOP
001044' 005767  003756                  TST     BRKFLG          ;ARE WE IN A BREAK LOOP?
001050' 001002                          BNE     4$              ;BR IF WE ARE
001052' 000167  003420                  JMP     EPASS           ;GO DO END OF PASS
                                ;
001056'                         4$:
001056' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                ;POINT TO POINT
001060' 012764  000001  005016' PP:     MOV     #1,FDFLAG(R4)   ;SAY WERE FULL DUPLEX
001066' 032700  000100                  BIT     #100,R0         ;HALF OR FULL DUPLEX?
001072' 001402                          BEQ     1$              ;BR IF HALF
001074' 000167  000512                  JMP     PPFD            ;JMP TO FULL DUPLEX ROUTINE
001100' 005064  005016'         1$:     CLR     FDFLAG(R4)      ;CORRECT FLAG CAUSE WERE NOT FD
001104' 032700  000040                  BIT     #40,R0          ;SLAVE OR MASTER?
001110' 001141                          BNE     PPHDM           ;BR IF MASTER
                                ;
                                ;POINT TO POINT, HALF DUPLEX, SLAVE
001112'                         PPHDS:
001112' 016464  004674' 004700'         MOV     AMSGBF(R4),RECADR(R4)  ;SET UP INPUT AREA
001120' 016764  000000G 004704'         MOV     MSGSIZ,RECCNT(R4)  ;SET UP COUNT
001126' 062764  000012  004704'         ADD     #12,RECCNT(R4)  ;CORRECT FOR HEADER AND CRC
001134' 016764  177030  004710'         MOV     TIMELN,RECTIM(R4)  ;HOW LONG TO WAIT
001142' 066764  177022  004710'         ADD     TIMELN,RECTIM(R4)  ;DOUBLE IT
001150' 004767  000766                  JSR     PC,REC          ;GO RECEIVE DATA
001154' 000462                          BR      FERR            ;BR TO ERROR ROUTINE
001156' 000401                  ONEREC: BR      1$              ;MAKE THIS A NOP=240 TO
001160' 000432                          BR      PPHDS2          ;DO ONE-WAY-IN (REC. ONLY)
001162' 012764  000000G 004752' 1$:     MOV     #MSGHED,TXADR1(R4)  ;GET MESS ADR
001170' 016764  000000G 004762'         MOV     MSGSIZ,TXCNT1(R4)  ;GET MSG SIZE
001176' 062764  000012  004762'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC'S
001204' 012764  000004  004766'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
001212' 012764  011202' 004756'         MOV     #FILL,TXADR2(R4)
001220' 004767  002744                  JSR     PC,CTS          ;GO GET CLEAR TO SEND
001224' 016764  176740  004772'         MOV     TIMELN,TMTTIM(R4)  ;GIVE 'EM 1/2 MINUTE
001232' 004767  002172                  JSR     PC,TMT          ;GO TRANSMIT
001236' 000423                          BR      DERR            ;BR TO ERROR ROUTINE
001240' 042774  000004  004664'         BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
001246' 005264  004742'         PPHDS2: INC     LOOP(R4)        ;COUNT AN ITERATION
001252' 026764  176710  004742'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH!
001260' 003314                          BGT     PPHDS           ;NO, GO DO ANOTHER TIME
001262' 005367  003432          LNDONE: DEC     LINACT
001266' 016403  004664'                 MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
001272' 042713  177775                  BIC     #177775,(R3)    ;KILL ALL BUT DTR
001276' 005063  000004                  CLR     4(R3)           ;KILL TRANSMITTER TOO
001302' 000167  177364                  JMP     GETNXT          ;YES DO NEXT LINE
                                ;
                                ;
001306' 042774  000004  004664' DERR:   BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
001314' 012703  177777                  MOV     #-1,R3          ;SET FLAG FOR WHICH RETURN
001320' 000401                          BR      FERR1
001322' 005003                  FFRR:   CLR     R3              ;SET FLAG FOR WHICH RETURN
```

```
 773  001324' 016401  004726'          FERR1:  MOV     CURLIN(R4),R1   ;GET LINE # ADDR
 774  001330' 105261  000020                  INCB    20(R1)          ;COUNT AN ERROR
 775  001334' 126167  000020  176630          CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
 776  001342' 002003                          BGE     DROP            ;BR IF YES
 777  001344' 005703                          TST     R3              ;DO A TMT OR DO A REC?
 778  001346' 001703                          BEQ     ONEREC          ;BR IF TMT
 779  001350' 000736                          BR      PPHDS2          ;GO TRY AGAIN
 780  001352' 142711  000200          DROP:   BICB    #200,(R1)       ;DROP THE LINE FROM TEST
 781  001356' 005264  005032'                 INC     DROPR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
 782  001362' 104411  011206' 000000'         MSGN$,ADROP,BEGIN       ;ASCII MESSAGE CALL WITH COMMON HEADER
 783  001370' 005767  003436                  TST     DROPR4
 784  001374' 001402                          BEQ     1$
 785  001376' 005004                          CLR     R4
 786  001400' 000402                          BR      2$
 787  001402' 012704  000002          1$:     MOV     #2,R4
 788  001406' 005364  005032'          2$:     DEC     DROPR4(R4)
 789  001412' 000723                          BR      LNDONE
 790
 791                                  ;
 792                                  ;POINT TO POINT HALF DUPLEX MASTER
 793                                  ;
 794  001414'                         PPHDM:
 795  001414' 012764  000000G 004752'         MOV     #MSGHED,TXADR1(R4)      ;GET MSG ADDRESS
 796  001422' 016764  000000G 004762'         MOV     MSGSIZ,TXCNT1(R4)       ;GET MSG SIZE
 797  001430' 062764  000012  004762'         ADD     #12,TXCNT1(R4)          ;CORRECT FOR CRC AND HEADER
 798  001436' 012764  000004  004766'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
 799  001444' 012764  011202' 004756'         MOV     #FILL,TXADR2(R4)
 800  001452' 004767  002512                  JSR     PC,CTS                  ;GO GET CLEAR TO SEND
 801  001456' 016764  176506  004772'         MOV     TIMELN,TMTTIM(R4)       ;GIVE 'EM 1/2 MINUTE
 802  001464' 004767  001740                  JSR     PC,TMT                  ;GO TRANSMIT
 803  001470' 000434                          BR      GERR                    ;ERROR RETURN
 804  001472' 042774  000004  004664'         BIC     #4,@DVASAV(R4)          ;KILL REQ TO SEND
 805  001500' 000401                  ONETMT: BR      1$                      ;MAKE THIS A NOP=240 FOR ONE
 806                                                                          ;WAY OUT, TMT ONLY, OTHER SIDE
 807                                                                          ;OF LINE MUST DO ONE WAY IN,
 808  001502' 000417                          BR      PPHDM2                  ;USED FOR ONE WAY OUT ONLY
 809  001504' 016464  004674' 004700' 1$:     MOV     AMSGBF(R4),RECADR(R4)   ;SET UP INPUT AREA
 810  001512' 016764  000000G 004704'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
 811  001520' 062764  000012  004704'         ADD     #12,RECCNT(R4)          ;CORRECT FOR HEADER AND CRC
 812  001526' 016764  176436  004710'         MOV     TIMELN,RECTIM(R4)       ;GIVE EM 1/2 MINUTES
 813  001534' 004767  000402                  JSR     PC,REC                  ;GO RECEIVE
 814  001540' 000413                          BR      HERR                    ;ERROR RETURN
 815  001542' 005264  004742'          PPHDM2: INC     LOOP(R4)                ;COUNT AN ITERATION
 816  001546' 026764  176414  004742'         CMP     LPCNT,LOOP(R4)          ;DONE ENOUGH?
 817  001554' 003317                          BGT     PPHDM                   ;BR IF NOT
 818  001556' 000167  177500                  JMP     LNDONE
 819                                  ;
 820                                  ;
 821  001562' 042774  000004  004664'  GERR:  BIC     #4,@DVASAV(R4)          ;KILL REQ TO SEND
 822  001570' 016401  004726'          HERR:  MOV     CURLIN(R4),R1           ;GET LINE # ADDRESS
 823  001574' 105261  000020                  INCB    20(R1)                  ;COUNT AN ERROR
 824  001600' 126167  000020  176364          CMPB    20(R1),ERRLVL           ;TOO MANY ERRORS?
 825  001606' 002261                          BGE     DROP                    ;BR IF YES
 826  001610' 000754                          BR      PPHDM2                  ;GO TRY AGAIN
 827                                  ;
 828                                  ;
```



```
 829                                  ;POINT -TO-POINT FULL DUPLEX
 830                                  ;
 831  001612' 032700  000040          PPFD:   BIT     #40,R0          ;MASTER OR SLAVE?
 832  001616' 001003                          BNE     1$              ;BR IF MASTER
 833  001620' 012764  177777  005016'         MOV     #177777,FDFLAG(R4)      ;SHOW ITS SLAVE
 834  001626' 004767  002336          1$:     JSR     PC,CTS          ;GO GET CLEAR TO SEND
 835  001632' 016464  004674' 004700' PPFD1:  MOV     AMSGBF(R4),RECADR(R4)   ;GET INPUT AREA
 836  001640' 016764  000000G 004704'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
 837  001646' 062764  000012  004704'         ADD     #12,RECCNT(R4)  ;CORRECT FOR CRC AND HEADER
 838  001654' 004767  000262                  JSR     PC,REC          ;GO GET RECEIVING STARTED
 839  001660' 000500                          BR      JERR            ;REC ERROR RETURN
 840  001662' 000441                          BR      FDRCDN          ;REC DONE
 841  001664' 012764  000000G 004752'         MOV     #MSGHED,TXADR1(R4)      ;FULL DUPLEX RETURN FROM REC
 842                                                                  ;GET TMT ADDR
 843  001672' 016764  000000G 004762'         MOV     MSGSIZ,TXCNT1(R4)       ;GET TMT COUNT
 844  001700' 062764  000012  004762'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC
 845  001706' 012764  000004  004766'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
 846  001714' 012764  011202' 004756'         MOV     #FILL,TXADR2(R4)
 847  001722' 004767  001502                  JSR     PC,TMT          ;GO TMT
 848  001726' 000455                          BR      JERR            ;TMT ERROR RETURN
 849  001730' 000412                          BR      1$              ;GOOD TMT DONE RETURN
 850  001732' 012700  002130'                 MOV     #FDTO,R0        ;FULL DUP RETURN, SET UP FOR TIME OUT
 851  001736' 016701  176226                  MOV     TIMELN,R1       ;GIVE EM 1 MINUTE
 852  001742' 066701  176222                  ADD     TIMELN,R1
 853  001746' 012702  000000                  MOV     #0,R2           ;SHOW ITS A REC WAIT
 854  001752' 000167  002330                  JMP     TIMKEP          ;GO WAIT
 855                                  ;
 856  001756' 012764  000001  005002' 1$:     MOV     #1,TMTDON(R4)   ;SHOW TMT IS DONE
 857  001764' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT,
 858                                  ;
 859                                  ;
 860  001766' 005764  005002'          FDRCDN: TST     TMTDON(R4)      ;IS TMT DONE YET?
 861  001772' 001020                          BNE     4$              ;BR IF YES
 862  001774' 005264  005056'                 INC     FDBKR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
 863  002000' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
 864  002004' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
 865  002010' 005767  003042                  TST     FDBKR4
 866  002014' 001402                          BEQ     2$
 867  002016' 005004                          CLR     R4
 868  002020' 000402                          BR      3$
 869  002022' 012704  000002          2$:     MOV     #2,R4
 870  002026' 005364  005056'          3$:     DEC     FDBKR4(R4)
 871  002032' 000755                          BR      FDRCDN          ;BR AND WAIT SOME MORE
 872  002034' 005264  004742'          4$:     INC     LOOP(R4)        ;COUNT AN ITERATION
 873  002040' 026764  176122  004742'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH
 874  002046' 003271                          BGT     PPFD1           ;NO, GO DO MORE
 875  002050' 042774  000004  004664'         BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
 876  002056' 000167  177200                  JMP     LNDONE
 877                                  ;
 878                                  ;
 879                                  ;FD TMT ERROR
 880  002062' 012761  000001  005002' JERR:   MOV     #1,TMTDON(R4)   ;DON'T BOTHER WAITING FOR TMT
 881  002070' 012764  177777  004722'         MOV     #177777,ABORT(R4)       ;TELL INT. ROUTINES TO GIVE UP
 882  002076' 016401  004726'                 MOV     CURLIN(R4),R1   ;GET CURRENT LINE INFO
 883  002102' 105261  000020                  INCB    20(R1)          ;COUNT ANOTHER ERROR
 884  002106' 126167  000020  176056          CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
```

```
885  002114' 002724                          BLT     FDRCDN          ;NO GO TRY AGAIN
886  002116' 042774  000004  004664'         BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
887  002124' 000167  177222                  JMP     DROP            ;GIVE UP ON THIS LINE
888                                          ;
889                                          ;
890                                          ;FULL DUPLEX RECEIVER TIMEOUTS COME HERE
891  002130' 012764  002062' 004746' FDTO:   MOV     #JERR,RECRTN(R4)        ;SET UP CORRECT RETURN
892  002136' 000167  001120                  JMP     RDTO            ;GO REPORT REC TIME OUT
893                                          ;
```



```
894                                          ;
895                                          ;SUBROUTINE TO RECEIVE DATA
896                                          ;CALLED WITH A JSR PC,REC
897                                          ;AFTER SETTING LOCATIONS RECADR(R4),RECCNT(R4),RECTIM(R4),VCTSAV(R4),
898                                          ;AND DVASAV(R4). RETURNS ON ERROR OR RETURNS
899                                          ;2 BYTES PAST CALL IF NO ERROR
900                                          ;
901  002142' 012664  004746'         REC:    MOV     (R6)+,RECRTN(R4);SAVE RTN ADDR
902  002146' 016401  004670'                 MOV     VCTSAV(R4),R1   ;GET VECTOR ADR
903  002152' 012711  002252'                 MOV     #RECINT,(R1)    ;MOVE IN INT. ROUTINE ADR
904  002156' 060421                          ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
905  002160' 116711  175626                  MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
906  002164' 005064  004722'                 CLR     ABORT(R4)       ;CLEAR ABORT FLAG
907  002170' 012764  000002  004714'         MOV     #2,SYNCNT(R4)   ;MUST GET 1 SYN CHARACTERS
908  002176' 005764  005016'                 TST     FDFLAG(R4)      ;FULL DUPLEX?
909  002202' 001410                          BEQ     1$              ;BR IF NOT
910  002204' 052774  000120  004664'         BIS     #120,@DVASAV(R4)  ;KICK OFF REC
911  002212' 016402  004746'                 MOV     RECRTN(R4),R2   ;GET RETURN ADDRESS
912  002216' 062702  000004                  ADD     #4,R2           ;FD RETURN
913  002222' 010207                          MOV     R2,R7           ;RETURN
914  002224' 012700  003262'         1$:     MOV     #RDTO,R0        ;RETURN ADDR
915  002230' 016401  004710'                 MOV     RECTIM(R4),R1   ;HOW LONG TO WAIT
916  002234' 012702  000000                  MOV     #0,R2           ;SHOWS ITS A REC WAIT
917  002240' 052774  000120  004664'         BIS     #120,@DVASAV(R4);START THE RECIEVER.
918  002246' 000167  002034                  JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
919                                          ;
920                                          ;RECEIVER INTERRUPT ROUTINE
921                                          ;
922  002252' 000405                  RECINT: BR      1$
923  002254' 010446                          MOV     R4,-(R6)        ;SAVE R4
924  002256' 010346                          MOV     R3,-(R6)        ;SAVE R3
925  002260' 012704  000002                  MOV     #2,R4           ;SET UP FOR SECOND LINE
926  002264' 000403                          BR      2$
927  002266' 010446                  1$:     MOV     R4,-(R6)        ;SAVE R4
928  002270' 010346                          MOV     R3,-(R6)        ;SAVE R3
929  002272' 005001                          CLR     R4              ;SET UP FOR FIRST LINE
930  002274' 016403  004664'         2$:     MOV     DVASAV(R4),R3   ;GET THE DEVICE ADDRESS
931  002300' 005764  004722'                 TST     ABORT(R4)       ;GIVE UP?
932  002304' 001403                          BEQ     3$              ;BR IF NO
933  002306' 042713  000560                  BIC     #560,(R3)       ;DISABLE REC INT.S
934  002312' 000460                          BR      7$              ;AND LEAVE
935  002314' 011364  004736'         3$:     MOV     (R3),ERRWD(R4)  ;SAVE STATUS
936  002320' 016346  000002                  MOV     2(R3),-(R6)     ;SAVE CHARACTER AND ERR BITS
937  002324' 005764  004714'                 TST     SYNCNT(R4)      ;ARE WE IN SYNC
938  002330' 001421                          BEQ     5$              ;BR IF YES
939  002332' 122716  000220                  CMPB    #220,(R6)       ;IS IT A DLE?
940  002336' 001406                          BEQ     4$              ;BR IF YES
941  002340' 126716  175630                  CMPB    SYNC,(R6)       ;IS IT A SYNC CHAR?
942  002344' 001440                          BEQ     6$              ;YES, IGNORE IT
943  002346' 042713  000020                  BIC     #20,(R3)        ;RESYNC IF NOT
944  002352' 000435                          BR      6$              ;LEAVE
945  002354' 005064  004714'         4$:     CLR     SYNCNT(R4)      ;SHOW WE'RE IN SYN
946  002360' 005764  005016'                 TST     FDFLAG(R4)      ;FD SLAVE?
947  002364' 100003                          BPL     5$              ;BR IF NOT
948  002366' 012764  000010  005006'         MOV     #10,TSYNCT(R4)  ;LET TMT GO IF FULL DUP
949  002374' 005767  175564          5$:     TST     REPORT          ;REPORT ERR NOW OR LATER?
```

```
                                BNE     51$             ;BR IF LATER
                                TSTB    ERRWD(R4)       ;IS REC DONE BIT SET?
                                BPL     RERR            ;BR IF NOT
                                TST     (R6)            ;DATA ERROR?
                                BPL     51$             ;BR IF NO DATA ERR
                                MOV     (R6)+,ERRWD(R4) ;SAVE INFORMATION
                                BIC     #100000,ERRWD(R4)       ;USE BIT 15 FOR FLAG TO RERR
                                BR      RERR+2
                        51$:    MOVB    (R6),@RECADR(R4)        ;STORE THE CHAR IN BUFFER
                                INC     RECADR(R4)      ;FOR NEXT TIME
                                DEC     RECCNT(R4)      ;COUNT IT
                                BLE     8$              ;BR IF DONE
                        6$:     TST     (R6)+           ;FIX STACK
                                BIS     #520,(R3)       ; READ ANOTHER
                        7$:     MOV     (R6)+,R3        ;RESTORE R3
                                MOV     (R6)+,R4        ;RESTORE R4
                                RTI                     ;RETURN
                        8$:     MOV     RECD(R4),RPIRQ+2
                                TST     (R6)+           ;FIX STACK
                                BIC     #560,(R3)       ;DISABLE REC INT.S
                                MOV     (R6)+,R3        ;RESTORE REGS
                                MOV     (R6)+,R4
                        RPIRQ:
                        ;---------------------------------------------------------------
                        PIRQ$,RECDN,BEGIN               ; RTI AND QUEUE UP TO CONTINUE AT RECDN
                        ;---------------------------------------------------------------
                        ;
                        RERR:   TST     (R6)+           ;FIX STACK
                                MOV     (R6)+,R3        ;RESTORE REG
                                BIC     #560,@DVASAV(R4)        ;RESET THE RECEIVER
                                MOV     #2$,1$+2                ;SET UP TO PIRQ TO
                                ADD     R4,1$+2         ;THE RIGHT PLACE
                                ADD     R4,1$+2
                                MOV     (R6)+,R4        ;RESTORE REG
                        1$:
                        ;---------------------------------------------------------------
                        PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
                        ;---------------------------------------------------------------
                        2$:     CLR     R4              ;SHOW ITS LINE 0
                                BR      3$              ;
                                MOV     #2,R4           ;SHOW ITS LINE 1
                        3$:     MOV     DVASAV(R4),CSRA ;SET UP FOR ERR CALL
                                MOV     ERRWD(R4),ACSR  ;SET UP FOR ERR CALL
                                MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE BYTE
                                MOVB    (R1),R2         ;GET VALUE OF LINE #
                                BIC     #177760,R2      ;THROW OTHER BITS AWAY
                                TST     ERRWD(R4)       ;IS IT A DATA OR LINE CHANGE ERR
                                BPL     DATAER          ;BR IF DATA ERR
                                MOV     R2,ASTAT        ;SET UP ERR INDICATION
                                ADD     #002000,ASTAT   ;SHOW WHICH TYPE
                        ;*****************************************************************
                                ERROR$,BEGIN            ;DATA SET CHANGE
                        ;*****************************************************************
                                EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                        ;
                        ;
```



```
                        DATAER: MOV     R2,ASTAT        ;SET UP ERR LINE #
                                MOVB    ERRWD+1(R4),ASTAT+1;GET ERR TYPE
                                BIC     #107400,ASTAT   ;LEAVE ONLY ERR BITS
                                INC     DTERR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
                        ;*****************************************************************
                                ERROR$,BEGIN            ;DATA ERR
                        ;*****************************************************************
                                TST     DTERR4
                                BEQ     1$
                                CLR     R4
                                BR      2$
                        1$:     MOV     #2,R4
                        2$:     DEC     DTERR4(R4)
                        ;
                        ;CHECK  ASTAT IN ERROR MESSAGE
                        ;0100XX = REC DATA PARITY ERROR
                        ;0200XX = FRAMING ERROR
                        ;0400XX = OVERRUN
                        ;XX = PHYSICAL LINE #
                        ;
                                JMP     RRET            ;GO DO AN ERROR EXIT
                        ;
                        ;
                        ;
                        RECDN:  CLR     R4              ;SHOW WE'RE LINE 0
                                BR      1$
                                MOV     #2,R4           ;SHOW WE'RE LINE 1
                        1$:     BIC     #177771,@DVASAV(R4)     ;SHUT UP DEVICE
                                CLR     RCVTO(R4)       ;RESET TIME OUT RETRY COUNT
                        3$:     TST     CKMSG           ;IS CHECK MSG ROUTINE BUSY?
                                BEQ     6$              ;BR IF NOT
                                INC     CHKR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
                                BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
                                BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
                                TST     CHKR4
                                BEQ     4$
                                CLR     R4
                                BR      5$
                        4$:     MOV     #2,R4
                        5$:     DEC     CHKR4(R4)
                                BR      3$
                        6$:     MOV     AMSGBF(R4),R2   ;GET REC DATA
                                CMP     MSG-2,6(R2)     ;IS HEADER CRC OK?
                                BEQ     7$              ;BR IF OK
                                TST     REPORT          ;REPORT ERROR NOW?
                                BNE     X5              ;BR IF LATER
                                MOV     #003000,ASTAT   ;SHOW ERR TYPE
                                BR      RECERR
                        7$:
                                MOVB    @CURLIN(R4),9$  ;GET LINE #
                                BIC     #177760,9$      ;LEAVE ONLY LINE BITS
                                TST     REPORT          ;REPORT BAD DATA?
                                BEQ     8$              ;BR IF YES
                                BIS     #100000,9$      ;SET FLAG TO CKMSG
                        8$:     ADD     #10,R2          ;GET ADDR OF MSG
                                MOV     R2,9$+2
```

```
1062  003074' 004567  000002G          JSR     R5,CKMSG+2      ;GO CHECK MSG
1063  003100' 000000'                  BEGIN
1064  003102' 000000                   9$:     0
1065  003104' 000000                           0
1066  003106' 000417                   BR      X4              ;BAD RETURN
1067  003110' 062702  000000G          ADD     MSGSIZ,R2       ;POINT IT TO MSG CRC, DATA WAS OK
1068  003114' 126722  000000G          CMPB    CRCMSG,(R2)+    ;CHECK 1ST CRC BYTE
1069  003120' 001025                   BNE     CER
1070  003122' 126722  000001G          CMPB    CRCMSG+1,(R2)+  ;CHECK SECOND
1071  003126' 001022                   BNE     CER
1072  003130' 062764  000002  004746'  ADD     #2,RECRTN(R4)   ;GOOD RETURN
1073  003136' 005064  005122'  RRET:   CLR     RETTAB(R4)      ;KILL IMPENDING TIMEOUTS
1074  003142' 000174  004746'          JMP     @RECRTN(R4)     ;AND GO BACK
1075                                   ;
1076                                   ;
1077  003146' 005767  175012           X4:     TST     REPORT          ;REPORT ERR NOW OR LATER?
1078  003152' 001001                   BNE     X5              ;BR IF LATER
1079  003154' 000770                   BR      RRET            ;OTHERWISE RETURN
1080                                   ;
1081  003156' 117402  004726'          X5:     MOVB    @CURLIN(R4),R2  ;GET CURRENT LINE #
1082  003162' 042702  177760           BIC     #177760,R2      ;
1083  003166' 105262  000252'          INCB    ECOUNT(R2)      ;COUNT AN ERROR FOR THAT LINE
1084  003172' 000761                   BR      RRET            ;AND RETURN
1085                                   ;
1086                                   ;
1087  003174' 005767  174764           CER:    TST     REPORT          ;REPORT NOW OR LATER?
1088  003200' 001366                   BNE     X5              ;BR IF LATER
1089  003202' 012767  004000  174644   MOV     #004000,ASTAT   ;INDICATE ERROR TYPE
1090  003210' 117402  004726'          RECERR: MOVB    @CURLIN(R4),R2  ;GENERATE LINE #
1091  003214' 042702  177760           BIC     #177760,R2      ;
1092  003220' 050267  174630           BIS     R2,ASTAT        ;ADD IN LINE #
1093  003224' 005264  005042'          INC     CRCR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
1094                                   ;*****************************************************************
1095  003230' 104404  000000'          ERROR$,BEGIN            ;CRC ERROR
1096                                   ;*****************************************************************
1097  003234' 005767  001602           TST     CRCR4
1098  003240' 001402                   BEQ     1$
1099  003242' 005004                   CLR     R4
1100  003244' 000402                   BR      2$
1101  003246' 012704  000002           1$:     MOV     #2,R4
1102  003252' 005364  005042'          2$:     DEC     CRCR4(R4)
1103  003256' 000167  177654           JMP     RRET            ;BAD RETURN
1104                                   ;
1105                                   ;READ TIME OUT ROUTINE
1106                                   ;
1107  003262' 012764  177777  004722'  RDTO:   MOV     #177777,ABORT(R4);TELL INT ROUTINES TO GIVE UP
1108  003270' 005264  005022'          INC     RCVTO(R4)       ;COUNT ANOTHER RETRY
1109  003274' 026467  005022'  174670   CMP     RCVTO(R4),ERRLVL  ;ENOUGH TRYS?
1110  003302' 001043                   BNE     4$              ;BR IF NO
1111  003304' 017467  004664'  174540   MOV     @DVASAV(R4),ACSR;GET REC STATUS
1112  003312' 042774  177771  004664'  BIC     #177771,@DVASAV(R4)     ;KILL ALL BUT DTR
1113  003320' 016467  004664'  174522   MOV     DVASAV(R4),CSRA ;PUT ADDR IN FOR ERR CALL
1114  003326' 016401  004726'          MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE BYTE
1115  003332' 111102                   MOVB    (R1),R2         ;GET VALUE OF PHY LINE #
1116  003334' 042702  177760           BIC     #177760,R2      ;THROW OTHER BITS AWAY
1117  003340' 062702  001000           ADD     #1000,R2        ;INDICATE REC TIMEOUT ERR
```



```
1118  003344' 010267  174504           MOV     R2,ASTAT        ;SET UP FOR ER CALL
1119  003350' 005264  005046'          INC     RCTOR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
1120                                   ;*****************************************************************
1121  003354' 104404  000000'          ERROR$,BEGIN            ;REC TIMED OUT
1122                                   ;*****************************************************************
1123  003360' 005767  001462           TST     RCTOR4
1124  003364' 001402                   BEQ     1$
1125  003366' 005004                   CLR     R4
1126  003370' 000402                   BR      2$
1127  003372' 012704  000002           1$:     MOV     #2,R4
1128  003376' 005364  005046'          2$:     DEC     RCTOR4(R4)
1129  003402' 005064  005022'          CLR     RCVTO(R4)       ;RESET RETRY COUNT
1130  003406' 000174  004746'          3$:     JMP     @RECRTN(R4)     ;ERROR RTN
1131                                   ;
1132  003412' 016401  004726'          4$:     MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
1133  003416' 105361  000020           DECB    20(R1)          ;DEC ERROR COUNT TO CORRECT FOR
1134                                                           ;INSUING INC
1135  003422' 005364  004742'          DEC     LOOP(R4)        ;CORRECT FOR LATER CALLING THIS A GOOD RUN
1136  003426' 000767                   BR      3$              ;RETURN
1137                                   ;
1138                                   ;
1139                                   ;SUBROUTINE TO TRANSMIT DATA
1140                                   ;CALLED WITH A JSR PC,TMT
1141                                   ;AFTER SETTING LOCATIONS TXADR1(R1), TXADR2(R4), TXCNT1(R4),TXCNT2(R4), TMTIM(R4),
1142                                   ;VCTSAV(R4), DVASAV(R4), SUBROUTINE CTS MUST BE CALLED FIRST.
1143                                   ;RETURNS ON ERROR, RETURNS 2 BYTES PAST NORMAL RETURN
1144                                   ;IF NO ERROR
1145                                   ;
1146  003430' 012664  004776'          TMT:    MOV     (SP)+,TMTRET(R4)  ;SAVE RTN ADDRESS
1147  003434' 005064  005002'          CLR     TMTDON(R4)      ;INITIALIZE FLAG
1148  003440' 016401  004670'          MOV     VCTSAV(R4),R1   ;GET REC VECTOR ADDRESS
1149  003444' 062701  000004           ADD     #4,R1           ;MAKE IT TMT VECTOR
1150  003450' 012711  003604'          MOV     #TMTINT,(R1)    ;MOVE IN INT ROUTINE ADDRESS
1151  003454' 060421                   ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
1152  003456' 116711  174330           MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
1153  003462' 005064  004722'          CLR     ABORT(R4)       ;CLEAR TIME OUT ABORT FLAG
1154  003466' 012764  000010  005006'  MOV     #10.,TSYNCT(R4) ;SEND TEN SYN CHARACTERS
1155  003474' 016403  004664'          MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
1156  003500' 005764  005016'          TST     FDFLAG(R4)      ;ARE WE FULL DUPLEX?
1157  003504' 001417                   BEQ     2$              ;BR IF NOT, ELSE
1158  003506' 100003                   BPL     1$              ;SKIP IF MASTER
1159  003510' 012764  177777  005006'  MOV     #177777,TSYNCT(R4)      ;MAKE TMT WAIT FOR A REC'D SYNC
1160  003516' 012763  000120  000004   1$:     MOV     #120,4(R3)      ;SET TMT INT ENABLE
1161  003524' 012763  000400  000006           MOV     #400,6(R3)      ;XMIT START OF MESSAGE.
1162  003532' 016402  004776'          MOV     TMTRET(R4),R2   ;GET BAD RET ADDR
1163  003536' 062702  000004           ADD     #4,R2           ;MAKE IT THE FULL DUPLEX RETURN
1164  003542' 010207                   MOV     R2,R7           ;AND RETURN
1165                                   ;
1166                                   ;
1167  003544' 012700  004036'          2$:     MOV     #TXTO,R0        ;TIME OUT RETURN ADDR
1168  003550' 016401  004772'          MOV     TMTTIM(R4),R1   ;HOW LONG TO WAIT
1169  003554' 012702  000004           MOV     #4,R2           ;SHOW ITS TRANSMITTER
1170  003560' 016403  004664'          MOV     DVASAV(R4),R3   ;GET THE STATUS REG
1171  003564' 012763  000120  000004   MOV     #120,4(R3)      ;SET TMT INT ENABLE AND SEND BIT
1172  003572' 012763  000400  000006   MOV     #400,6(R3)      ;XMIT START OF MESSAGE.
1173  003600' 000167  000502           JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
```

```
1174                                    ;TMT INTERRUPT ROUTINE
1175                                    ;
1176  003604' 000405                    TMTINT: BR      1$
1177  003606' 010446                            MOV     R4,-(SP)        ;SAVE R4
1178  003610' 010346                            MOV     R3,-(SP)        ;SAVE R3
1179  003612' 012704  000002                    MOV     #2,R4           ;SET TO LINE 1
1180  003616' 000403                            BR      2$
1181  003620' 010446                    1$:     MOV     R4,-(SP)        ;SAVE R4
1182  003622' 010346                            MOV     R3,-(SP)        ;SAVE R3
1183  003624' 005004                            CLR     R4              ;SET TO LINE 0
1184  003626' 005764  004722'           2$:     TST     ABORT(R4)       ;GIVE UP?
1185  003632' 100417                            BMI     TIEXIT          ;BR IF YES
1186  003634' 016403  004664'                   MOV     DVASAV(R4),R3   ;GET DVA
1187  003640' 005764  004762'                   TST     TXCNT1(R4)      ;FIRST BUFFER TMT'ED?
1188  003644' 001015                            BNE     FTBUF           ;BR IF NOT
1189  003646' 005764  004766'                   TST     TXCNT2(R4)      ;SECOND BUFFER TMT'ED?
1190  003652' 001437                            BEQ     TIDONE          ;BR IF YES
1191  003654' 117463  004756' 000006            MOVB    @TXADR2(R4),6(R3)  ;SEND NEXT BYTE
1192  003662' 005264  004756'                   INC     TXADR2(R4)      ;MOV ADDRESS POINTER
1193  003666' 005364  004766'                   DEC     TXCNT2(R4)      ;COUNT IT
1194  003672' 012603                    TIEXIT: MOV     (R6)+,R3        ;RESTORE R3
1195  003674' 012604                            MOV     (R6)+,R4        ;RESTORE R4
1196  003676' 000002                            RTI                     ;RETURN
1197  003700' 005764  005006'           FTBUF:  TST     TSYNCT(R4)      ;SENT ENOUGH FILL CHARS?
1198  003704' 001407                            BEQ     2$              ;BR IF YES
1199                                                                    ;IF NEGITIVE
1200  003706' 100402                            BMI     1$              ;SEND TILL RECINT CLEARS TSYNCT
1201  003710' 005364  005006'                   DEC     TSYNCT(R4)      ;ELSE COUNT IT
1202  003714' 116763  174254  000006    1$:     MOVB    SYNC,6(R3)      ;SEND IF YES.
1203  003722' 000763                            BR      TIEXIT          ;AND LEAVE
1204  003724' 117463  004752' 000006    2$:     MOVB    @TXADR1(R4),6(R3)  ;SEND NEXT BYTE
1205  003732' 042763  000400  000006            BIC     #400,6(R3)      ;ASSURE CLEAR TSOM BIT.
1206  003740' 005264  004752'                   INC     TXADR1(R4)      ;MOVE ADDRESS POINTER
1207  003744' 005364  004762'                   DEC     TXCNT1(R4)      ;COUNT IT
1208  003750' 000750                            BR      TIEXIT          ;AND LEAVE
1209                                    ;
1210                                    ;ALL CHAR'S HAVE BEEN TMT'ED
1211  003752' 042763  000120  000004    TIDONE: BIC     #120,4(R3)      ;KILL TMT'ER
1212  003760' 012603                            MOV     (R6)+,R3        ;RESTORE R3
1213  003762' 005064  005126'                   CLR     RETTAB+4(R4)    ;KILL IMPENDING TIMEOUT
1214  003766' 062764  000002  004776'           ADD     #2,TMTRET(R4)   ;MAKE IT A GOOD RETURN
1215  003774' 005704                            TST     R4              ;LINE 0 OR 1
1216  003776' 001004                            BNE     1$              ;BR IF LINE 1
1217  004000' 012604                            MOV     (R6)+,R4        ;RESTORE R4
1218                                            ;---------------------------------------------------------------
1219  004002' 000004  004020' 000000'           PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
1220                                            ;---------------------------------------------------------------
1221  004010' 012604                    1$:     MOV     (R6)+,R4        ;RESTORE R4
1222                                            ;---------------------------------------------------------------
1223  004012' 000004  004026' 000000'           PIRQ$,3$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 3$
1224                                            ;---------------------------------------------------------------
1225  004020' 005004                    2$:     CLR     R4              ;SET TO LINE 0
1226  004022' 000174  004776'                   JMP     @TMTRET(R4)     ;RETURN
1227  004026' 012704  000002            3$:     MOV     #2,R4           ;SET TO LINE 1
1228  004032' 000174  004776'                   JMP     @TMTRET(R4)     ;RETURN
1229                                    ;
```



```
1230                                    ;
1231                                    ;ROUTINE IF TRANSMIT TIMES OUT
1232                                    ;
1233  004036' 012764  177777  004722'   TXTO:   MOV     #177777,ABORT(R4);TELL INT ROUTINES TO GIVE UP
1234  004044' 016407  004664' 173776            MOV     DVASAV(R4),CSRA ;GET ADDR OF STATUS
1235  004052' 062767  000004  173770            ADD     #4,CSRA         ;MAKE IT TMT CSR
1236  004060' 017767  173764  173764            MOV     @CSRA,ACSR      ;SAVE STATUS
1237  004066' 042777  177767  173754            BIC     #177767,@CSRA   ;KILL TRANSMISSION AND INT
1238  004074' 116763  174074  000006            MOVB    SYNC,6(R3)      ;IDLE WITH SYN CHARACTER.
1239  004102' 052763  000400  000006            BIS     #400,6(R3)
1240  004110' 016401  004726'                   MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
1241  004114' 111102                            MOVB    (R1),R2         ;GET VALUE
1242  004116' 042702  177760                    BIC     #177760,R2      ;LEAVE ONLY LINE #
1243  004122' 062702  005000                    ADD     #5000,R2        ;INDICATE TMT TIMEOUT
1244  004126' 010267  173722                    MOV     R2,ASTAT        ;SET UP FOR ERROR CALL
1245  004132' 005264  005052'                   INC     TXTOR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
1246                                            ;***************************************************************
1247  004136' 104404  000000'                   ERROR$,BEGIN            ;TMT TIMEOUT
1248                                            ;***************************************************************
1249  004142' 005767  000704                    TST     TXTOR4
1250  004146' 001402                            BEQ     1$
1251  004150' 005004                            CLR     R4
1252  004152' 000402                            BR      2$
1253  004154' 012704  000002            1$:     MOV     #2,R4
1254  004160' 005364  005052'           2$:     DEC     TXTOR4(R4)
1255  004164' 000174  004776'                   JMP     @TMTRET(R4)     ;BAD RETURN
1256                                    ;
1257                                    ;
1258                                    ;
1259                                    ;
1260                                    ;
1261                                    ;SUB GETS CLEAR TO SEND FOR LINE IN DVASAV(R4) AND VCTSAV(R4)
1262
1263  004170' 012664  005012'   CTS:     MOV     (R6)+,CTSRET(R4)  ;SAVE RET ADDRESS
1264  004174' 016401  004670'           MOV     VCTSAV(R4),R1   ;GET VECTOR ADDR
1265  004200' 012711  004236'           MOV     #CTSINT,(R1)    ;MOV IN INT ROUTINE ADDR
1266  004204' 060421                    ADD     R4,(R1)+        ;SHOWS WHICH LINE
1267  004206' 116711  173600            MOVB    BR1,(R1)        ;GET BR1 LEVEL
1268  004212' 032774  020000  004664'   BIT     #20000,@DVASAV(R4)  ;IS CLEAR TO SEND UP?
1269  004220' 001402                    BEQ     RTSSET          ;BR IF NOT
1270  004222' 016407  005012'           MOV     CTSRET(R4),R7   ;RETURN
1271  004226' 012774  000046  004664' RTSSET: MOV     #46,@DVASAV(R4)  ;SET REQ TO SEND
1272  004234' 104400                    EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
1273                                    ;INT ROUTINE
1274  004236' 000403                    CTSINT: BR      1$              ;DIFFERENT PIRQ'S
1275                                            ;---------------------------------------------------------------
1276  004240' 000004  004254' 000000'           PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
1277                                            ;---------------------------------------------------------------
1278  004246'                           1$:
1279                                            ;---------------------------------------------------------------
1280  004246' 000004  004262' 000000'           PIRQ$,3$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 3$
1281                                            ;---------------------------------------------------------------
1282  004254' 012704  000002            2$:     MOV     #2,R4           ;SET UP FOR LINE 1
1283  004260' 000401                            BR      4$
1284  004262' 005004                    3$:     CLR     R4              ;SET UP FOR LINE 0
1285  004264' 032774  020000  004664'   4$:     BIT     #20000,@DVASAV(R4)  ;IS CLEAR TO SEND SET?
```

```
1286  004272' 001755                          BEQ     RTSSET          ;GO TRY AGAIN IF NOT
1287  004274' 042774  000040  004664'         BIC     #40,@DVASAV(R4) ;DISABLE DATA SET CHG INT'S
1288  004302' 016407  005012'                 MOV     CTSRET(R4),R7   ;RETURN
1289                                  ;
1290                                  ;
1291                                  ;
1292                                  ;ALL ROUTINES COME HERE WHILE WAITING FOR I/O
1293                                  ;ROUTINE EITHER KICKS OFF SECOND LINE, NOTIFYS
1294                                  ;ROUTINES OF TIMEOUTS, OR JUST DOES BREAKS.
1295                                  ;I/O ROUTINES THAT FINISH UP BEFORE TIMEOUT VALUE
1296                                  ;MUST CLEAR LOCATION RETTAB(R4) +0 IF RECEIVER OR +4
1297                                  ;IF TMT.
1298                                  ;CALLED WITH A JMP TO TIMKEP AFTER LOADING
1299                                  ; R0 = WHERE TO RETURN IF TIME OUT
1300                                  ; R1 = HOW LONG TO WAIT (IN SECONDS)
1301                                  ; R2 = 0 IF ITS A REC TIMEING OR 4 IF ITS A TMT WAIT
1302                                  ; R4 = 0 FOR ACTIVE LINE 0, 2 FOR ACTIVE LINE 1
1303                                  ;
1304                                  ;
1305  004306' 060402                 TIMKEP: ADD     R4,R2           ;FIGURE OUT WHICH TABLE ENTRY
1306  004310' 010162  005112'                 MOV     R1,TIMTAB(R2)   ;STORE HOW LONG TO WAIT
1307  004314' 067762  000000G 005112'        ADD     @TIME,TIMTAB(R2)        ;ADD PRESENT TIME
1308  004322' 010462  005132'                 MOV     R4,R4TAB(R2)    ;SAVE WHICH ACTIVE LINE
1309  004326' 010062  005122'                 MOV     R0,RETTAB(R2)   ;WHERE TO RET TO, PLUS
1310                                                                  ;IF THIS ENTRY IN TABLE IS
1311                                                                  ;0 THEN NO TIMEOUT WILL
1312                                                                  ;OCCUR FOR THIS ENTRY SET
1313  004332' 022767  000002  000360          CMP     #2,LINACT       ;ARE 2 LINES GOING?
1314  004340' 001410                          BEQ     TIMCHK          ;BR IF YES
1315  004342' 005704                          TST     R4              ;ARE WE LINE 0?
1316  004344' 001402                          BEQ     1$              ; BR IF YES
1317  004346' 005004                          CLR     R4              ; IF NO MAKE IT 0
1318  004350' 000402                          BR      2$
1319  004352' 012704  000002          1$:     MOV     #2,R4           ;WE WERE 0 SO NOW DO 1
1320  004356' 000167  174310          2$:     JMP     GETNXT          ;GO DO OTHER LINE
1321                                  ;
1322                                  ;
1323  004362' 012702  005122'         TIMCHK: MOV     #RETTAB,R2      ;GET ADDR OF TIME STATUSES
1324  004366' 005722                  1$:     TST     (R2)+           ;ANY WAIT GOING ON?
1325  004370' 001026                          BNE     4$              ;BR IF YES
1326  004372' 020227  005132'         11$:    CMP     R2,#RETTAB+10   ;ARE WE DONE LOOKING
1327  004376' 001373                          BNE     1$              ;BR BACK IF NO
1328  004400' 005767  000424                  TST     PASTIM          ;TIME FOR END OF PASS
1329  004404' 001402                          BEQ     2$              ;BR IF NO
1330  004406' 000167  000064                  JMP     EPASS           ;GO REPORT END OF PASS
1331                                  ;
1332  004412' 005767  000410          2$:     TST     BRKFLG          ;ARE WE ALREADY IN A BREAK?
1333  004416' 001401                          BEQ     3$              ;BR IF NO
1334  004420' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
1335                                  ;
1336  004422' 012767  000001  000376  3$:     MOV     #1,BRKFLG       ;SHOW WERE DOING BREAK
1337  004430' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
1338  004434' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
1339  004440' 005067  000362                  CLR     BRKFLG          ;SHOW WERE BACK
1340  004444' 000746                          BR      TIMCHK
1341                                  ;
```



```
1342  004446' 005742                  4$:     TST     -(R2)           ;MOVE R2 BACK
1343  004450' 027762  000000G 177770         CMP     @TIME,-10(R2)   ;IS TIME UP?
1344  004456' 003002                          BGT     5$              ;BR IF YES
1345  004460' 005722                          TST     (R2)+           ;BUMP R2 BACK AGAIN
1346  004462' 000743                          BR      11$             ;GO CHECK OTHERS
1347                                  ;
1348  004464' 016204  000010          5$:     MOV     10(R2),R4       ;SET R4 CORRECTLY FOR WHICH LINE
1349  004470' 011203                          MOV     (R2),R3         ;SAVE RETURN
1350  004472' 005012                          CLR     (R2)            ;CLEAR THIS ENTRY
1351  004474' 010307                          MOV     R3,R7           ;JMP TO TIME OUT ROUTINE
1352                                  ;
1353                                  ;
1354                                  ;
1355                                  ;EPASS DOES END OF PASS CALL AND PRINTS ERR SUMMARY
1356                                  ;IF LOCATION REPORT IS NOT 0
1357                                  ;
1358  004476' 005767  173462          EPASS:  TST     REPORT          ;DO ERR SUM OR NOT
1359  004502' 001461                          BEQ     5$              ;BR IF NO
1360  004504' 005267  000146                  INC     MSGCNT          ;COUNT A PASS
1361  004510' 026767  000142  173446          CMP     MSGCNT,REPORT   ;TIME FOR A MSG?
1362  004516' 001053                          BNE     5$              ;IF NOT, BR
1363  004520' 005067  000132                  CLR     MSGCNT
1364  004524' 012700  000212'                 MOV     #PHYTAB,R0      ;FIND WHICH LINES ARE RUNNING
1365  004530' 012701  011306'                 MOV     #SUM1+4,R1      ;POINTS TO MSG
1366  004534' 105710                  1$:     TSTB    (R0)            ;IS LINE IN USE?
1367  004536' 100403                          BMI     2$              ;BR IF YES
1368  004540' 012711  054130                  MOV     #54130,(R1)     ;PUT XX IN # OF ERRS
1369  004544' 000427                          BR      4$
1370  004546' 116002  000040          2$:     MOVB    32.(R0),R2      ;GET # OF ERRS
1371  004552' 105060  000040                  CLRB    32.(R0) ;CLEAR THE COUNT FOR NEXT PASS.
1372  004556' 022702  000077                  CMP     #77,R2          ;ANY GOOD MSGS
1373  004562' 002416                          BLT     3$              ;BR IF NO
1374  004564' 010203                          MOV     R2,R3           ;PUT IN MSG
1375  004566' 042703  177770                  BIC     #177770,R3      ;LEAVE ONLY LOW DIGIT
1376  004572' 062703  000060                  ADD     #60,R3  ;MAKE IT ASCII
1377  004576' 110361  000001                  MOVB    R3,1(R1)        ;STORE IT AWAY
1378  004602' 006202                          ASR     R2              ;GET NEXT DIGIT
1379  004604' 006202                          ASR     R2
1380  004606' 006202                          ASR     R2
1381  004610' 062702  000060                  ADD     #60,R2          ;MAKE IT ASCII
1382  004614' 110211                          MOVB    R2,(R1) ;PUT IT IN MSG
1383  004616' 000402                          BR      4$
1384  004620' 012711  042102          3$:     MOV     #42102,(R1)     ;PUT IN A 'BD' FOR BAD
1385  004624' 005200                  4$:     INC     R0              ;BUMP LINE POINTER
1386  004626' 062701  000010                  ADD     #10,R1          ;POINT TO NXT MSG
1387  004632' 020027  000232'                 CMP     R0,#PHYTAB+20   ;DONE
1388  004636' 001336                          BNE     1$              ;GO DO MORE
1389  004640' 104411  004652' 040000'         MSGN$,MSUM,BEGIN        ;ASCII MESSAGE CALL WITH COMMON HEADER
1390
1391                                  ;
1392  004646'                         5$:
1393  004646' 104402  040000'                 ENDPSS,BEGIN            ;SIGNAL END OF PASS. CONTINUE AT RESTRT
1394
1395                                  ;
1396  004652' 011212'                 MSUM:   SUM
1397  004654' 177777                          -1
```

```
1398                                    ;
1399  004656' 000000                   MSGCNT: 0                       ;COUNTS END PASSES FOR MESSAGE CALL
1400  004660' 000000                   MASK:   0                       ;USED FOR BIC'ING IN LOOPS
1401  004662' 000000                   NXTLIN: 0                       ;ADR OF NEXT LIN
1402  004664' 000000                   DVASAV: 0                       ;HOLDS DVA FOR ACTIVE L#0
1403  004666' 000000                           0                       ;HOLDS DVA FOR ACTIVE L#0
1404  004670' 000000                   VCTSAV: 0                       ;HOLDS VCT FOR ACTIVE L#0
1405  004672' 000000                           0                       ;HOLDS VECTOR FOR ACTIVE L#0
1406  004674' 005142'                   AMSGBF: MSGBF0                  ;ADDRESS OF MSG BUFFER 0
1407  004676' 007162'                           MSGBF1                  ;ADDRESS OF MSG BUFFER 1
1408  004700' 000000                   RECADR: 0                       ;CURRENT REC WORD POINTER 0
1409  004702' 000000                           0                       ;CURRENT REC WORD POINTER 1
1410  004704' 000000                   RECCNT: 0                       ;REC BYTE COUNT FOR 0
1411  004706' 000000                           0                       ;REC BYTE COUNT FOR 1
1412  004710' 000000                   RECTIM: 0                       ;TEMP HOLDS TIME OUT VALUE FOR A REC
1413  004712' 000000                           0                       ;SAME BUT FOR LINE 1
1414  004714' 000000                   SYNCNT: 0                       ;COUNTS HOW MANY SYN CHARACTERS REC
1415  004716' 000000                           0                       ;SHOULD WAIT FOR.
1416  004720' 000000                   LINACT: 0                       ;HOLD THE NUMBER OF LINES RUNNING,0-2
1417  004722' 000000                   ABORT:  0                       ;ABORT FLAG FOR L#0
1418  004724' 000000                           0                       ;ABORT FLAG FOR L#0
1419  004726' 000000                   CURLIN: 0                       ;POINTS TO STATUS BYTE FOR CURRENT L#0
1420  004730' 000000                           0                       ;POINTS TO STATUS BYTE FOR CURRENT L#0
1421  004732' 002712'                   RECD:   RECON                   ;WHERE TO PIRQ AFTER REC DONE
1422  004734' 002716'                           RECDN+4                 ;
1423  004736' 000000                   ERRWD:  0                       ;HOLDS ERR WORDS TEMPORARILY
1424  004740' 000000                           0                       ;HOLDS ERR WORDS TEMPORARILY
1425  004742' 000000                   LOOP:   0                       ;HOLDS LOOP COUNT FOR LINE # 0
1426  004744' 000000                           0                       ;HOLDS LOOP COUNT FOR LINE # 1
1427  004746' 000000                   RECRTN: 0                       ;HOLDS RETURN ADDR FOR LINE #0
1428  004750' 000000                           0                       ;HOLDS RETURN ADDR FOR LINE #0
1429  004752' 000000                   TXADR1: 0                       ;HOLDS 1ST ADDR TO TMT FROM, LINE 0
1430  004754' 000000                           0                       ;SAME FOR LINE 1
1431  004756' 000000                   TXADR2: 0                       ;HOLDS 2ND ADDR TO TMT FROM, LINE 0
1432  004760' 000000                           0                       ;SAME FOR LINE 1
1433  004762' 000000                   TXCNT1: 0                       ;HOLDS 1ST TMT COUNT FOR LINE 0
1434  004764' 000000                           0                       ;SAME FOR LINE 1
1435  004766' 000000                   TXCNT2: 0                       ;HOLDS 2ND TMT COUNT FOR LINE 0
1436  004770' 000000                           0                       ;SAME FOR LINE 1
1437  004772' 000000                   TMTTIM: 0                       ;HOLDS TIME OUT TIME FOR LINE 0
1438  004774' 000000                           0                       ;SAME FOR LINE 1
1439  004776' 000000                   TMTRET: 0                       ;HOLDS RTN ADDR FOR TMT, LINE 0
1440  005000' 000000                           0                       ;SAME FOR LINE 1
1441  005002' 000000                   TMTDON: 0                       ;FLAG FOR WHEN A TMT IS DONE
1442  005004' 000000                           0                       ;SAME FOR OTHER LINE
1443  005006' 000000                   TSYNCT: 0                       ;HOLDS # OF SYN CHAR TO SEND, LINE 0
1444  005010' 000000                           0                       ;SAME FOR LINE 1
1445  005012' 000000                   CTSRET: 0                       ;HOLDS RETURN ADDR FOR CTS
1446  005014' 000000                           0                       ;SAME FOR LINE 1
1447  005016' 000000                   FDFLAG: 0                       ;0 IF HALF DUPLEX, 1 IF FULL
1448  005020' 000000                           0
1449  005022' 000000                   RCVTO:  0                       ;COUNTS REC TIME OUT RETRYS
1450  005024' 000000                           0
1451  005026' 000000                   BRKFLG: 0                       ;IS A ONE DURING BREAKS
1452  005030' 000000                   PASTIM: 0                       ;HAS A 1 WHEN ITS TIME FOR END OF PASS
1453  005032' 000000                   DPOPR4: 0       ;EXIT FLAGS
```



```
1454  005034' 000000                           0
1455  005036' 000000                   DTERR4: 0
1456  005040' 000000                           0
1457  005042' 000000                   CRCR4:  0
1458  005044' 000000                           0
1459  005046' 000000                   RCTOR4: 0
1460  005050' 000000                           0
1461  005052' 000000                   TXTOR4: 0
1462  005054' 000000                           0
1463  005056' 000000                   FDBKR4: 0
1464  005060' 000000                           0
1465  005062' 000000                   CHKR4:  0
1466  005064' 000000                           0
1467
1468                                    ;
1469                                    ;
1470                                    ;R1 HAS ADDR OR VECTOR
1471                                    ;R0 HAS LINE # IN LOW BITS, RETURNS WITH ADDR OR VECTOR IN R1
1472  005066'                           GETADR:
1473  005066' 010046                            MOV     R0,-(R6)        ;SAVE R0
1474  005070' 042700  177760                   BIC     #177760,R0      ;SAVE ONLY LINE #
1475  005074' 001404                    1$:     BEQ     2$              ;DONE
1476  005076' 062701  000010                    ADD     #10,R1          ;BUMP TO NEXT LINE
1477  005102' 005300                            DEC     R0              ;ADDED ENOUGH TIMES?
1478  005104' 000773                            BR      1$              ;GO BACK
1479  005106' 012600                    2$:     MOV     (R6)+,R0        ;RESTORE R0
1480  005110' 000207                            RTS     PC              ;RETURN WITH DVA IN R1
1481                                    ;
1482                                    ;
1483  005112' 000000                   TIMTAB: 0                       ;WHEN TO TIME OUT FOR REC,LINE 0
1484  005114' 000000                           0                       ;WHEN TO TIME OUT FOR REC,LINE 1
1485  005116' 000000                           0                       ;WHEN TO TIME OUT FOR TXT,LINE 0
1486  005120' 000000                           0                       ;WHEN TO TIME OUT FO TXT,LINE 1
1487  005122' 000000                   RETTAB: 0                       ;WHERE TO RETURN FOR REC LINE 0
1488  005124' 000000                           0                       ;WHERE TO RETURN FOR REC LINE 1
1489  005126' 000000                           0                       ;WHERE TO RETURN FOR TXT LINE 0
1490  005130' 000000                           0                       ;WHERE TO RETURN FOR TXT LINE 1
1491  005132' 000000                   R4TAB:  0                       ;HOLD R4 FOR RETURNS
1492
1493  005134' 000000                           0
1494
1495  005136' 000000                           0
1496  005140' 000000                           0
1497                                    ;
1498                                    ;
1499  005142' 001010                   MSGBF0: .BLKW   520.             ;REC BUFFER FOR LINE 0
1500  007162' 001010                   MSGBF1: .BLKW   520.             ;REC BUFFER FOR LINE 1
1501
1502
1503
1504  011202' 177777                   FILL:   177777                   ;4 CHARACTERS OF FILLER BYTES
1505  011204' 177777                           177777
1506  011206' 011504'                  ADROP:  MDROP
1507  011210' 177777                           -1
1508                                    .EVEN
1509  011212' 005015  044523  041516   SUM:    .ASCII" <15><12>'SINCE LAST SUMMARY PER LINE, IN OCTAL '<15><12>
```

```
1510  011220' 020105  040514  052123
1511  011226' 051440  046525  040515
1512  011234' 054522  050040  051105
1513  011242' 046040  047111  026105
1514  011250' 044440  020116  041517
1515  011256' 040524  020114  005015
1516  011264' 044514  042516  021440          .ASCII  'LINE #  ERRS'<15><12>
1517  011272' 020040  051105  051522
1518  011300' 005015
1519  011302' 030060  026455  030060  SUM1:   .ASCII  '00--00'<15><12>
1520  011310' 005015
1521  011312' 030460  026455  030060          .ASCII  '01--00'<15><12>
1522  011320' 005015
1523
1524  011322' 031060  026455  030060          .ASCII  '02--00'<15><12>
1525  011330' 005015
1526
1527
1528  011332' 031460  026455  030060          .ASCII  '03--00'<15><12>
1529  011340' 005015
1530
1531
1532  011342' 032060  026455  030060          .ASCII  '04--00'<15><12>
1533  011350' 005015
1534  011352' 032460  026455  030060          .ASCII  '05--00'<15><12>
1535  011360' 005015
1536  011362' 033060  026455  030060          .ASCII  '06--00'<15><12>
1537  011370' 005015
1538  011372' 033460  026455  030060          .ASCII  '07--00'<15><12>
1539  011400' 005015
1540  011402' 030061  026455  030060          .ASCII  '10--00'<15><12>
1541  011410' 005015
1542  011412' 030461  026455  030060          .ASCII  '11--00'<15><12>
1543  011420' 005015
1544  011422' 031061  026455  030060          .ASCII  '12--00'<15><12>
1545  011430' 005015
1546  011432' 031461  026455  030060          .ASCII  '13--00'<15><12>
1547  011440' 005015
1548  011442' 032061  026455  030060          .ASCII  '14--00'<15><12>
1549  011450' 005015
1550  011452' 032461  026455  030060          .ASCII  '15--00'<15><12>
1551  011460' 005015
1552  011462' 033061  026455  030060          .ASCII  '16--00'<15><12>
1553  011470' 005015
1554  011472' 033461  026455  030060          .ASCII  '17--00'<15><12>
1555  011500' 005015
1556
1557  011502'    000                          .BYTE   0
1558          011504'                         .EVEN
1559                                  ;
1560                                  ;
1561  011504' 044514  042516  053440  MDROP:  .ASCIZ  'LINE WAS DROPPED'
1562  011512' 051501  042040  047522
1563  011520' 050120  042105    000
1564          000001                .END
```



```
ABORT   004722R          881*   906*   931   1107*  1153*  1184   1233*  1417#
ACSR    000052R          467#   992*  1111*  1236*
ADDR    000006R          446#   639    697
ADROP   011206R          782   1506#
AMSGBF  004674R          740    809    835   1047   1406#
ASB     000056R          470#
ASTAT   000054R          469#   998*   999*  1006*  1007*  1008*  1052*  1089*  1092*  1118*  1244*
AWAS    000060R          471#
BDCNV = ******  G          1#
BEGIN   000000R          443#   620    621    653    654    667    719    782    863    864    974    986   1001
                        1011   1038   1039   1063   1095   1121   1219   1223   1247   1276   1280   1337   1338
                        1389   1393
BIT0  = 000001           480#
BIT1  = 000002           480#
BIT10 = 002000           480#
BIT11 = 004000           480#
BIT12 = 010000           480#
BIT13 = 020000           480#
BIT14 = 040000           480#
BIT15 = 100000           480#
BIT2  = 000004           480#
BIT3  = 000010           480#
BIT4  = 000020           480#
BIT5  = 000040           480#
BIT6  = 000100           480#
BIT7  = 000200           480#
BIT8  = 000400           480#
BIT9  = 001000           480#
BREAK$= 104407           480#   653    654    863    864   1038   1039   1337   1338
BRKFLG  005026R          686*   723   1332   1336*  1339*  1451#
BR1     000012R          448#   905   1152   1267
BR2     000013R          449#
CDATA$= 104414           480#
CER     003174R         1069   1071   1087#
CHKR4   005062R          682*   683*  1037*  1040   1045*  1465#
CKMSG = ******  G        480#  1035   1062
CRCMSG= ******  G        480#  1060   1078
CRCR4   005042R          674*   675*  1093*  1097   1102*  1457#
CSRA    000050R          465#   991*  1113*  1234*  1235*  1236   1237*
CTS     004170R          754    800    834   1263#
CTSINT  004236R         1265   1274#
CTSRET  005012R         1263*  1270   1288   1445#
CURLIN  004726R          692*   773    822    882    993   1055   1081   1090   1114   1132   1240   1419#
DATAER  002634R          997   1006#
DATCK$= 104417           480#
DATER$= 104405           480#
DERR    001306R          757    769#
DEVSET= ******  G        481#   622
DONE    001002R          690    711#
DROP    001352R          776    780#   825    887
DROPR4  005032R          670*   671*   781*   783    788*  1453#
DTERR4  005036R          672*   673*  1009*  1013   1018*  1455#
DVASAV  004664R          699*   758*   763    769*   804*   821*   875*   886*   910*   917*   930    979*   991
                        1033*  1111   1112*  1113   1155   1170   1186   1234   1268   1271*  1285   1287*  1402#
DVID1   000014R          450#   618
ECOUNT  000252R          549#  1083*
```

```
ENDPS$= 104402          480#  1393
END$  = 104403          480#   620     667     719
EPASS   004476R         725   1330    1359#
ERRCNT  000030R         457#
ERRLVL  000172R         489#   775     824     884    1109
ERRN$ = 104410          480#
ERROR$= 104404          480#  1001    1011    1095    1121    1247
ERRTAB  000232R         530#
ERRWD   004736R         935*   951     955*    956*    992     996    1007    1423#
EXIT$ = 104400          480#   720     857    1003    1272    1334
FDBKR4  005056R         680*   681*    862*    865     870*   1463#
FDFLAG  005016R         730*   734*    833*    908     946    1156    1447#
FDRCDN  001766R         840    860#    871     885
FDTO    002130R         850    891#
FERR    001322R         746    772#
FERR1   001324R         771    773#
FILL    011202R         753    799     846    1504#
FTBUF   003700R        1188   1197#
GERR    001562R         803    821#
GETADR  005066R         698    701    1472#
GETNXT  000672R         689#   696     766    1320
GETPA$= 104413          480#
GWBUF$= 104412          480#
HERR    001570R         814    822#
INIT    000022R         454#
JERR    002062R         839    848     880#    891
LINACT  004720R         688*   703*    711     762*   1313    1416#
LNDONE  001262R         762#   789     818     876
LOOP    004742R         704*   759*    760     815*    816     872*    873    1135*   1425#
LPCNT   000166R         487#   760     816     873
MAP22$= 104415          480#
MASK    004660R         623*   627     629     633     635*   1400#
MD    = ******  GX        707
MDATAB  000272R         573#
MDMTAB  000312R         600#
MDROP   011504R        1500   1561#
MODNAM  000000R         444#
MODSP   000164R         455    478#
MSG   = ******  G       480#  1048
MSGBF0  005142R        1406   1499#
MSGBF1  007162R        1407   1500#
MSGCNT  004656R         650*  1360*   1361    1363*   1399#
MSGHED= ******  G       480#   749     795     841
MSGN$ = 104411          480#   782    1389
MSGSIZ= ******  G       480#   741     750     796     810     836     843    1067
MSGS$ = 104416          480#
MSG$  = 104406          480#
MSUM    004652R        1389   1396#
NXTLIN  004662R         687*   689     691     694*   1401#
OACNV = ******  G         1#
ONEREC  001156R         491    747#    778
ONETMT  001500R         492    805#
OPEN  = 000000          445    451     458     459     460     461     462     463     464     465     467     469     470
                        471    480#
PASCNT  000026R         456#
PASTIM  005030R         685*   721*   1320    1452#
```



```
PHYTAB  000212R         508#   624     631     648     687     689     714     717    1364    1387
PIRQ$ = 000004          480#   974     986    1219    1223    1276    1280
POPSP = 005726          480#
POPSP2= 022626          480#
PP      001660R         708    730#
PPFD    001612R         733    831#
PPFD1   001632R         835#   874
PPHDM   001414R         736    794#    817
PPHDM2  001542R         800    815#    826
PPHDS   001112R         739#   761
PPHDS2  001246R         748    759#    779
PRTY  = 000000          480#
PRTY0 = 000000          480#
PRTY1 = 000040          480#
PRTY2 = 000100          480#
PRTY3 = 000140          480#
PRTY4 = 000200          480#
PRTY5 = 000240          448    449     480#
PRTY6 = 000300          480#
PRTY7 = 000340          480#
PS    = 177776          480#
PSW   = 177776          480#
PUSH  = 005746          480#
PUSH2 = 024646          480#
PWRCNT= ******  G       480#   665
QUES  = 104401          480#
RCTOR4  005046R         676*   677*   1119*   1123    1128*   1459#
RCVTO   005022R         705*  1014*   1108*   1109    1129*   1449#
RDTO    003262R         892    914    1107#
REC     002142R         745    813     838     901#
RECADR  004700R         740*   809*    835*    958*    959*   1408#
RECCNT  004704R         741*   742*    810*    811*    836*    837*    960*   1410#
RECD    004732R         967   1421#
RECDN   002712R         974   1030#   1421    1422
RECERR  003210R        1053   1090#
RECINT  002252R         903    922#
RECRTN  004746R         891*   901*    911    1072*   1074    1130    1427#
RECTIM  004710R         743*   744*    812*    915    1412#
REPORT  000164R         484#   949    1050    1057    1077    1087    1358    1361
RERR    002510R         952    957     977#
RESTRT  000540R         472    652     665#
RETTAB  005122R        1073*  1213*   1309*   1323    1326    1407#
RPIRQ   002502R         967*   972#
RRET    003136R        1026   1073#   1079    1084    1103
RSTRT   000062R         472#
RTSSET  004226R        1269   1271*   1286
R4TAB   005132R        1308*  1491#
SBADR   000052R         466#
SPARE1  000202R         493#
SPARE2  000204R         494#
SPARE3  000206R         495#
SPARE4  000210R         496#
SPOINT  000024R         455#   669
SPSIZ = 000040            1#   473
SR1     000016R         451#   636
START   000332R         454    618#
```

```
STAT    000020R          453#
SUM     011212R         1396    1549#
SUM1    011302R         1365    1519#
SVR0    000032R          458#
SVR1    000034R          459#
SVR2    000036R          460#
SVR3    000040R          461#
SVR4    000042R          462#
SVR5    000044R          463#
SVR6    000046R          464#
SYNC    000174R          490#    644*    646     941    1202    1238
SYNCNT  004714R          907*    937     945*   1414#
TIDONE  003752R         1190    1211#
TIEXIT  003672R         1185    1194#   1203    1208
TIMCHK  004362R          713    1314    1323#   1340
TIME  = ******  G        480#   1307    1343
TIMELN  000176R          488#    743     744     755     801     812     851     852
TIMKEP  004306R          854     918    1173    1305#
TIMTAB  005112R         1306*   1307*   1483#
TMT     003430R          756     802     847    1146#
TMTDON  005002R          856*    860     880*   1147*   1441#
TMTINT  003604R         1150    1176#
TMTRET  004776R         1146*   1162    1214*   1226    1228    1255    1439#
TMTTIM  004772R          755*    801*   1160    1437#
TPX   = 000000           480#
TRAPX = 000020           480#
TSYNCT  005006R          948*   1154*   1159*   1197    1201*   1443#
TXADR1  004752R          749*    795*    841*   1204    1206*   1429#
TXADR2  004756R          753*    799*    846*   1191    1192*   1431#
TXCNT1  004762R          750*    751*    796*    797*    843*    844*   1187    1207*   1433#
TXCNT2  004766R          752*    798*    845*   1189    1193*   1435#
TXTO    004036R         1167    1233#
TXTOR4  005052R          678*    679*   1245*   1249    1254*   1461#
VCTSAV  004670R          702*    902    1148    1264    1404#
VECTOR  000010R          447#    700
WAIT  = ******  G        480#    651
WASADR  000051R          468#
XFLAG   000056R          445#
X4      003146R         1066    1077#
X5      003156R         1051    1078   1081#   1088
.     = 011525R         1499#   1500#  1558#

. ABS.  000000     000
        011525     001

% ERRORS DETECTED:  1
% DEFAULT GLOBALS GENERATED:  1

 *NPBA,NPBA/SOL/CRF:SYM=DDXCOM,NPBA
 RUN-TIME: 6 12 1 SECONDS
 RUN-TIME RATIO: 55/20=2.6
 CORE USED:  8K  (15 PAGES)
```

```
                .REM %

                                        IDENTIFICATION
                                        --------------

                                PRODUCT CODE:   MAINDEC-11-DXNLA-A-D

                                PRODUCT NAME:   DL-11 DECNET DEC/X11 EXERCISER MODULE

                                DATE:   21-JUN-76

                                MAINTAINER:     DEC/X11 DIAGNOSTIC GROUP

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

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

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


                COPYRIGHT (C) 1976 DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
```

## 1. ABSTRACT

THIS TYPE OF DEC/X11 MODULE, REFERRED TO AS DECNET DEC/X11 MODULE, WILL DETECT SYSTEM INTERACTION ERRORS WHEN RUN IN CONJUNCTION WITH OTHER DEC/X11 MODULES EXERCISING OTHER DEVICES ON THE SYSTEM. ERRORS ARE REPORTED ON THE SYSTEM CONSOLE AS THEY OCCUR OR AS A SUMMARY REPORT, AT THE OPERATORS OPTION. IN ADDITION, THESE MODULES MAY BE USED TO VERIFY THE CONDITION OF THE ASSOCIATED MODEMS AND COMMUNICATION LINKS. IT WILL ONLY TEST COMMUNICATIONS PATHS CAPABLE OF RUNNING DDCMP. THIS MEANS THE DEVICES MUST USE NO PARITY, 8 LEVEL CODE, AND TERMINALS CANNOT BE TESTED.

NOTE

THIS DOCUMENT ASSUMES FAMILIARITY WITH THE PHILOSOPHY AND OPERATION OF DEC/X11. THE DETAILS OF LINKING, LOADING, AND RUNNING MODULES UNDER DEC/X11 CAN BE FOUND IN THE DEC/X11 USERS DOCUMENTATION AND REFERENCE GUIDE, MAINDEC-11-DXQBA. THE READER SHOULD ALSO HAVE READ THE DECNET DEC/X11 USER'S GUIDE, MAINDEC-11-DXQBC, WITHOUT THROUGHLY UNDERSTANDING THAT DOCUMENT, THERE IS LITTLE HOPE OF SUCCESSFUL TESTING.

## 2. HARDWARE REQUIREMENTS

THESE MODULES REQUIRE A COMPLETE PATH FROM A COMMUNICATIONS DEVICE TO A COMMUNICATIONS DEVICE. THIS MAY BE ACCOMPLISHED THROUGH A VARIETY OF METHODS:

A. 1 PROCESSOR, 1 COMM. DEVICE, AND A LOOP BACK DEVICE WITH MODEM SIMULATION CAPABILITIES. (MUST RUN FULL DUPLEX)

B. 1 PROCESSOR, 2 COMM. DEVICES, 2 MODEMS

C. 2 PROCESSORS, 2 COMM. DEVICES, 2 MODEMS

UP TO SIXTEEN LINES CAN BE HANDLED BY ONE MODULE BUT ONLY 2 LINES ARE KEPT RUNNING SIMULTANEOUSLY.

## 3. SOFTWARE REQUIREMENTS

A. THE DEC/X11 MONITOR MUST BE QABK OR A LATER REVISION.

B. THE N1A? MODULE MUST BE CONFIGURED DIRECTLY AFTER THE MONITOR*

C. ALL NETWORK MODULES MUST BE CONFIGURED NEXT.

D. THE N2A? MODULE MUST BE CONFIGURED NEXT.*

E. ONE OF THE 3 CLOCK MODULES MUST BE CONFIGURED NEXT. EITHER KWA?, KWB?, OR KWX?

F. ANY OTHER DESIRED DEC/X11 MODULES CAN THEN BE CONFIGURED IN ANY ORDER.

4. PRELIMINARY TESTING

THE APPROPRIATE STAND ALONE DIAGNOSTICS MUST BE RUN TO VERIFY THE CPU, AND THE COMMUNICATIONS DEVICE. THEN A DEC/X11 EXERCISER USING THE CONVENTIONAL (MAINTENANCE MODE) DEC/X11 MODULES FOR THE COMMUNICATIONS DEVICES MUST BE RUN. IF BOTH THESE STEPS PASS SUCCESSFULLY, DECNET DEC/X11 MAY BE ATTEMPTED.

5. PROGRAMMING CONSIDERATIONS

A. OTHER THAN THE REQUIREMENTS MENTIONED IN 3 REGARDING THE ORDER IN WHICH MODULES MUST BE REQUIRED, THESE DECNET DEC/X11 MODULES WILL BE COMPLETELY COMPATIBLE WITH THE DEC/X11 MONITORS AND ALL OTHER MODULES. THE ONLY OTHER EXECPTION IS THAT FOR A GIVEN COMMUNICATIONS DEVICE, THERE WILL NOW BE 2 DIFFERENT DEC/X11 MODULES CAPABLE OF BEING RUN, THE OLDER MAINTANCE MODE TYPE REQUIRING NO CONNECTIONS TO THE COMMUNICATION DEVICE, AND THE NEW DECNET TYPE REQUIRING A COMPLETED COMMUNICATIONS PATH. IT WILL BE POSSIBLE (AND SOMETIMES DESIREABLE) TO CONFIGURE 2 DIFFERENT DEC/X11 MODULES IN ONE RUN TIME EXERCISER WHICH TEST THE SAME PHYSICAL DEVICE. IF THIS IS DONE, ONLY ONE OF THE 2 MODULES MAY BE SELECTED FOR THE SAME RUN.

B. USING THESE DECNET DEC/X11 MODULES, IT IS POSSIBLE TO HAVE MANY NODES CONNECTED IN A NETWORK, ALL RUNNING DEC/X11 TOGETHER. ALL THESE EXERCISERS WOULD HAVE BEEN STARTED AT DIFFERENT TIMES. DUE TO DIFFERENT HARDWARE ON EACH SYSTEM, RUN TIMES AND END OF PASS TIMES MIGHT VARY CONSIDERABLY. THEREFORE IT IS NECCESSARY THAT THESE DECNET DEC/X11 MODULES BE EXTREMELY PATIENT AND HAVE A HIGH RETRY CAPABILITY IN ALL OPERATIONS. THIS HOWEVER, MEANS IT SOMETIMES MAY TAKE A LONG TIME (ON THE ORDER OF 10 MINUTES) BEFORE AN OPERATOR FINDS OUT A LINE IS DEFINITELY NOT RUNNING.

C. DUE TO THE EXCESSIVE PROGRAMMING COMPLICATIONS AND LARGE

---

* THESE ARE DESCRIBED IN THE DECNET DEC/X11 USER'S GUIDE.

BUFFERS REQUIRED TO RUN MANY LINES FROM ONE MODULE ALL AT THE SAME TIME, IT WAS DECIDED TO ONLY RUN 2 AT ANY ONE TIME. IF THE MODULE IS SET TO RUN 16 LINES, IT WILL RUN 1 PAIR FOR THE NUMBER OF TRANSFERS INDICATED BY LOCATION LPCNT, THEN IT WILL RUN A 2ND PAIR, THEN 3RD, ETC UP TO THE 8TH PAIR. IT WILL THEN REPORT AN END OF PASS AND GO BACK TO THE FIRST PAIR, THEN 2ND, ETC. THIS FORCES SOME CONSIDERATIONS TO BE MADE WHEN CONNECTING LINES TOGETHER SO THAT A LINE IN THE ACTIVE MODE IS NOT WAITING ON A LINE IN THE DORMANT MODE. PLEASE READ THE DECNET DEC/X11 USER'S GUIDE, ESPECIALLY THE SECTION ON 2 LINES AT A TIME CONSIDERATIONS.

D. THESE MODULES WILL, AS A GENERAL RULE, USE THE MINIMUM AMOUNT OF FEATURES OF A COMMUNICATION DEVICE REQUIRED TO SEND AND RECEIVE MESSAGES. THIS MEANS THAT CRC CALCULATING HARDWARE, CHARACTER RECOGNITION HARDWARE, ETC, WILL NOT BE USED. THESE FEATURES ARE FULLY TESTED BY STAND ALONE DIAGNOSTICS.

## 6. OPERATING PROCEDURES

A. THE MODULE IS CONFIGURED, LOADED, AND RUN AS SPECIFIED IN THE DEC/X11 REFERENCE GUIDE.

B. CONSOLE SWITCHES ARE STANDARD FOR DEC/X11

C. THE N1A? MODULE WILL SOLICIT FROM THE OPERATOR WHICH TYPE OF MESSAGE HE WANTS TO USE FOR DECNET ACTIVITY. THE OPTIONS INCLUDE:

- ALL ONES
- ALL ZEROS
- ALTERNATING ONES AND ZEROS
- A DIGIT (4 BIT) REPEATING COUNT PATTERN
- A PRECANNED ASCII MESSAGE
- USER SPECIFIED BY TYPING IN

D. THE N2A? MODULE WILL MAKE ANY AUTOMATIC PHONE CONNECTIONS USING ANY DN-11'S PRESENT AND WILL REQUEST THE OPERATOR TO DIAL ALL MANUAL CONNECTIONS. ALL OPERATORS AT EACH NODE UNDER TEST MUST REACH THIS POINT BEFORE ANY ONE NODE CONTINUES INTO THE EXERCISING PHASE.

E. THE N2A? MODULE WILL REQUEST A CARRIAGE RETURN WHEN THE OPERATOR WANTS TO BEGIN EXERCISING HIS NODE.

F. THERE ARE MANY OPTIONS WHICH MAY BE ALTERED BEFORE RUN TIME, THEY INCLUDE:

(1) LOCATION 164 IS A SWITCH WHICH INDICATES WHETHER OR NOT ALL ERRORS SHOULD BE REPORTED AS THEY HAPPEN. IF LOCATION 164 IS NON-ZERO (EQUAL TO N), THEN ERRORS ARE TOTALLED (ON A LINE BY LINE BASIS); EVERY N PASSES THIS

TOTAL NUMBER OF ERRORS FOR THE PREVIOUS N PASSES WILL BE REPORTED. THE VALUE OF N IS THE VALUE OF LOCATION 164. THE MAXIMUM NUMBER OF ERRORS THAT CAN BE TALLEYED IS 256 PER LINE. THEREFORE IF A LINE IS NOT KNOWN TO BE OF VERY GOOD QUALITY, THE VALUE OF LOCATION 164 TIMES THE VALUE OF LOCATION 166 SHOULD BE LESS THAN 128. (LOCATION 166 IS DEFINED NEXT.) THIS MEANS THE NUMBER OF MESSAGES PER PASS TIMES THE NUMBER OF PASSES PER SUMMARY REPORT SHOULD BE LESS THAN 128. IN THIS WAY, IF EVERY MESSAGE ATTEMPTED HAS AN ERROR ON BOTH TRANSMIT AND RECEIVE, THE ERROR TALLY WILL NOT OVERFLOW. IF THE LINE IS OF KNOWN GOOD QUALITY, THIS RESTRICTION MAY BE RELAXED. THE DEFAULT VALUE IS TO NOT REPORT TOTAL ERRORS.(LOC 164=0) OF COURSE DEC/X11 KEEPS TOTALS FOR THE ENTIRE RUN, REGARDLESS.

(2) LOCATION 166 INDICATES HOW MANY ITERATIONS SHOULD BE DONE BEFORE AN END OF PASS IS CALLED. AN ITERATION IS NORMALLY ONE TRANSMIT AND ONE RECEIVE, IN EITHER ORDER. THE DEFAULT VALUE IS 8.

(3) LOCATION 170 IS A TIME SCALING FACTOR. THERE ARE MANY TIME OUT LOOPS IN THE MODULES, SOME SHOULD LOGICALLY BE LONGER THAN OTHERS. THE VALUE IN LOCATION 170 IS THE NUMBER OF SECONDS TO WAIT FOR THE SMALLEST TIME OUTS. SOME TIME OUTS WILL BE A MULTIPLE OF THIS FACTOR. DEFAULT IS 30 SECONDS.

(4) LOCATION 172 IS AN ERROR TOLERATION LEVEL, ON A SINGLE MESSAGE BASIS. SO IF IN AN ATTEMPT TO DO ONE ITERATION (USUALLY A TRANSMIT THEN RECIEVE OR VICE-VERSA) THE NUMBER OF ERRORS REACHES THE VALUE IN LOCATION 172, THAT LINE IS DROPPED FROM TESTING. IF BEFORE THIS LEVEL IS REACHED, AN ITERATION IS CORRECTLY COMPLETED, THEN THE ERROR COUNT WHICH IS COMPARED TO LOCATION 172 IS RESET TO ZERO. THE DEFAULT IS 10.

(5) LOCATION 174 CONTAINS 2 IDENTICAL BYTES FOR USE AS SYNC CHARACTERS. THE DEFAULT IS 113226 WHICH IS 2 BYTES OF 226. IF THIS IS CHANGED, BOTH BYTES MUST BE IDENTICAL TO EACH OTHER. THIS LOCATION IS A DON'T CARE FOR ASYNCHRONOUS DEVICES, BECAUSE THEY "SYNC" ON THE FIRST BYTE OF THE ACTUAL MESSAGE, NOT ON PRECEDING CHARACTERS AS SYNCHRONOUS DEVICES DO.

(6) LOCATION 176 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN RECEIVE-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, SLAVE. (BIT 7 = 1, BITS 4,5, AND 6 = 0'S IN THE PHYSICAL LINE TABLE. SEE ITEM 6-F-9 BELOW). THEN PATCH THE LOCATION POINTED TO BY LOCATION 176 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, SLAVE BY THIS MODULE TO RUN IN RECEIVE ONLY MODE. THIS LOCATION MAY BE PATCHED BACK TO A 401 TO RESUME NORMAL REC-TMT TESTING.

(7) LOCATION 200 IS AN ADDRESS. IF THE OPERATOR WANTS TO RUN THE DEVICE IN TRANSMIT-ONLY MODE, HE SELECTS THE LINE FOR HALF DUPLEX, POINT-TO-POINT, MASTER (BITS 7 AND 5 = 1'S, BITS 4 AND 6 = 0'S IN THE PHYSICAL-LINE TABLE, SEE ITEM 6-F-9 BELOW.) THEN PATCH THE LOCATION POINTED TO BY LOCATION 200 FROM A 401 TO A 240. THIS WILL CAUSE ALL LINES BEING TESTED IN HALF DUPLEX, POINT-TO-POINT, MASTER BY THIS MODULE TO RUN IN TRANSMIT ONLY MODE. THE LOCATION MAY BE PATCHED BACK TO 401 TO RESUME NORMAL TMT-REC TESTING.

G. LOCATIONS DVA, VCT, BR1 AND BR2 MUST BE SET TO PROPERLY MATCH THE FIRST DEVICE TO BE TESTED BY THIS MODULE.

7. ERROR REPORTING

A. ERRORS ARE EITHER TOTALLED AND PRINTED AS SUMMARIES ON A LINE BY LINE BASIS, OR THEY ARE PRINTED AS THEY OCCUR. SEE SECTION 6F1. THEY ARE 2 CLASSES OF ERRORS DATA ERRORS AND OTHER ERRORS.

B. DATA ERRORS ARE ACTUALLY REPORTED BY THE N1A? MODULE BECAUSE THIS IS WHERE THE SHARED DATA COMPARE ERROR ROUTINE IS LOCATED. SO ALL DECNET DEC/X11 MODULE DATA COMPARE ERRORS LOOK LIKE THE STANDARD DEC/X11 DATA COMPARE ERROR MESSAGE EXCEPT:

(1) THE MODULE NAME TYPED IS FFXX, WHERE FF IS FILLED IN TO BE THE NAME OF THE DECNET DEC/X11 MODULE WHICH CALLED THE N1A? MODULE TO DO THE COMPARE. IN OTHER WORDS FF REPRESENTS THE MODULE THAT "HAD" THE ERROR.

(2) THE ERROR NUMBER IS A COUNT OF THE DATA COMPARE ERRORS IN THE CURRENT MESSAGE BEING CHECKED, IT IS THEREFORE RESET TO ZERO EVERY TIME A NEW MESSAGE IS TO BE CHECKED.

(3) ACSR CONTAINS THE ADDRESS OF THE DECNET DEC/X11 MODULE WHICH CALLED THIS ROUTINE TO HAVE ITS MESSAGE CHECKED.

(4) SBADR CONTAINS THE LINE # OF THE DEVICE WHICH HAD THE DATA ERROR.

(5) WASADR CONTAINS THE COUNT (IN OCTAL) OF WHICH CHARACTER IN THE MESSAGE THIS BAD CHARACTER WAS.

(6) THE "SHOULD BE AND "WAS" ITEMS ARE NORMAL.

C. OTHER ERRORS ARE REPORTED USING THE NORMAL ERROR CALL IN

DEC/X11 CSRA AND ACSR ARE STANDARD. STATC CONTAINS A CODED

ERROR NUMBER, WITH THE CODE BEING CONSISTENT IN ALL DECNET DEC/X11 MODULES. THE LOW ORDER 2 DIGITS OF STATC CONTAIN WHICH LINE HAD THE ERROR THE OTHER DIGITS CONTAIN THIS CODE:

| | |
|---|---|
| 0010XX | REC TIME OUT |
| 0020XX | TMT CLOCK LOSS |
| 0030XX | HEADER CRC ERROR |
| 0040XX | MSG CRC ERROR |
| 0050XX | TMT TIME OUT |
| 0060XX | TMT NON-EXISTENT MEMORY |
| 0070XX | TMT LATENCY |
| 0100XX | REC DATA PARITY ERROR |
| 0110XX | REC NON-EXISTENT MEMORY |
| 0120XX | REC CLOCK LOSS |
| 0200XX | REC FRAMING ERROR |
| 0400XX | REC OVERRUN OR LATENCY ERROR |

THIS METHOD SHOULD ELIMINATE THE CONSTANT NEED FOR HAVING A LISTING ON HAND TO DECIDE WHAT THE ERRORS WHERE. THE FIRST TIME THE RUN TIME EXERCISER IS RUN AFTER LOADING, THE N1A? MODULE WILL TYPE THIS ERROR CODE LIST OUT TO THE SYSTEM CONSOLE FOR THE OPERATORS SUBSEQUENT USE.

D. ALL ERRORS HAVE A RETRY LIMIT (SEE SECTION 6F4) WHICH CAN BE MODIFIED.

E. ON POWER-FAIL, POWER-UP CONDITIONS, DECNET DEC/X11 MODULES WILL DROP THEMSELVES. THIS IS DONE BECAUSE THE PHONE CONNECTIONS WILL HAVE BEEN LOST AND OPERATOR INTERVENTION WILL BE REQUIRED.

8. TEST SEQUENCE

LINES ARE TESTED BY PAIRS IN SEQUENTIAL ORDER. THIS ORDER MAY BE MODIFIED BY MODIFYING THE PHYSICAL LINE TABLE. (SEE DECNET DEC/X11 USER'R GUIDE). THERE ARE SEVERAL TESTING OPTIONS WITH SLIGHTLY DIFFERENT FUNCTIONS MAKING UP AN ITERATION:

A. POINT-TO-POINT SLAVE. IN THIS CASE THE LINE WAITS TO RECEIVE A MESSAGE, WHEN IT DOES, IT CHECKS IT, AND THEN TRANSMITS A MESSAGE BACK. IT WILL REPORT ERRORS ON EXCESSIVE WAITS.

B. POINT-TO-POINT MASTER. IN THIS CASE THE LINE SENDS A MESSAGE AND WAITS TO RECEIVE ONE IN RETURN. IT THEN CHECKS THE RECEIVED MESSAGE. IF IT TIMES-OUT, IT TRANSMITS AGAIN, IT WILL CONTINUE TO TIME OUT AND RE-TRANSMIT UNTIL EITHER A MESSAGE IS RECEIVED, OR IT TIMES OUT COUNT EQUALS LOCATION

172, THE ERROR TOLERATION LEVEL. (SEE SECTION 6F4). IT WILL THEN REPORT AN ERROR AND BEGIN AGAIN. AFTER THE NUMBER OF

CONSECUTIVE TIME-OUT ERRORS REPORTED EQUALS LOCATION 172, THE LINE WILL BE DROPPED.

C. POINT-TO-POINT SLAVE RECEIVE ONLY. IN THIS MODE THE LINE WAITS ONLY TO RECEIVE, CHECKS THE MESSAGE, AND WAITS TO RECEIVE AGAIN.

D. POINT-TO-POINT MASTER-TRANSMIT ONLY. IN THIS MODE THE LINE JUST TRANSMITS AND THEN TRANSMITS AGAIN ,ETC.

E. MULTI-DROP MASTER. IN THIS MODE, THE LINE WILL SEND A DDCMP MESSAGE OUT FOR THE ADDRESS CONTAINED IN THE MULTI-DROP MENU TABLE, WAIT TO RECEIVE A MESSAGE BACK, AND THEN CHECK THE DATA.
* NOT YET IMPLEMENTED

F. MULTI-DROP SLAVE. IN THIS MODE, THE LINE WILL WAIT FOR A MESSAGE WITH THE MATCHING ADDRESS. THEN IT WILL TRANSMIT A MESSAGE.
* NOT YET IMPLEMENTED

G. THESE BASIC ITERATIONS ARE REPEATED UP TO THE NUMBER OF TIMES IN LOCATION 166. (SEE SECTION 6F2) FOR EACH LINE SELECTED FOR TEST, AND THE END-OF-PASS IS DECLARED.

9. PASS TIMES

PASS TIMES ARE EXTREMELY CONFIGURATION DEPENDENT. DIFFERENT BAUD RATES, MESSAGE SIZES, AND TESTING MODES ALL GREATLY AFFECT THE PASS TIME. BELOW ARE SOME BALL PARK TIMES, BUT ANY ONE SYSTEM COULD VARY TREMENDOUSLY FROM THESE:
-RUNNING ALONE, 1 LINE ON AN 11/05 AT 9600 BAUD,HALF DUPLEX, POINT-TO-POINT, WITH A 512 CHARACTER MESSAGE, TAKES 15 SECONDS.

NOTE 1 AT VARIOUS MONITOR CALLS THROUGHOUT THIS MODULE, IT IS LIKELY THAT WHEN 2 LINES ARE RUNNING, A MONITOR CALL CAN BE MADE AND BEFORE CONTORL IS RETURNED TO THE MODULE, THE OTHER LINE MAY CAUSE THE EXACT SAME CALL TO BE MADE. IN FACT, WHEN RUNNING 2 LINES FULL DUPLEX, IT MAY BE POSSIBLE TO HAVE 4 CALLS TO THE MONITOR FROM ONE SPOT IN THE MODULE, BEFORE THE FIRST CALL IS RETURNED FROM. THIS IS WHY AT SUCH CALLS, THE CODE SETS COUNTERS BEFORE DOING CALLS, SO THAT WHEN IT IS RETURNED TO, IT CAN KEEP WHICH LINE DID WHAT STRAIGHT.

```
                        NOTE 2  WHEN RUNNING 2 LINES LOOPED TOGETHER FROM THE SAME MODULE,
                        THE LOW ORDER ONE (PER DVC, LOC. 14) SHOULD BE SLAVE AND THE HIGHER
                        ORDER ONE SHOULD BE MASTER.


                        NOTE 3  YOU CAN RUN THE DL'S WITHOUT MODEMS IF A NULL MODEM IS USED
                        AND LOCATION TAGGED "NULL:" IN THE CTS: (CLEAR TO SEND) ROUTINE IS
                        PATCHED FROM 104400 TO A 772.  THIS WILL GENERALLY REQUIRE ALL DL'S
                        BEING RUN BY THIS MODEM TO NOT USE MODEMS.  IF YOU HAVE SOME WITH
                        AND SOME WITHOUT, USE 2 NLA? MODULES.
                        %
```

```
                                ;PARITY MUST BE OFF
                                ;STOP CODE LENGTH MUST BE SAME (1,1.5,OR 2 BITS)
                                ;CHARACTER SIZE MUST BE 8 BITS
000000'                                 IOMOD <NLAA >, 1,4,4,4
000000'                                 MODULE  140000,NLAA ,1,4,4,4,
                                        .TITLE  NLAA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  4    9/6/75
                                        .LIST   BIN
                                ;*************************************************************************
000000'                         BEGIN:
000000' 046116  040501      040 MODNAM: .ASCII  /NLAA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 000001                  ADDR:   1+0             ;1ST DEVICE ADDR.
000010' 000004                  VECTOR: 4+0                     ;1ST DEVICE VECTOR.
000012'     200                 BR1:    .BYTE   PRTY4+0         ;1ST BR LEVEL.
000013'     200                 BR2:    .BYTE   PRTY4+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' 000332'                 INIT:   START                   ;MODULE START ADDR.
000024' 000164'                 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.
000062' 000516'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000164'                         MODSP:
                                ;*************************************************************************
                                        .GLOBL  WAIT, TIME, MSGSIZ, CKMSG, CRCMSG,MSGHED, MSG,PWRCNT
                                        .GLOBL  DEVSET
                                ;
                                ;
000164' 000000                  REPORT: 0               ;FLAG, IF =#, THEN ERRORS ON RECEIVED MSG'S
                                                        ;ARE TOTALLED AND PRINTED EVERY # PASSES
                                                        ;IF =0, ERRORS ARE ALL REPORTED AS THEY HAPPEN.
000166' 000010                  LPCNT:  10              ;ITERATIONS PER END PASS CALL
000170' 000036                  TIMELN: 30.             ;TIME OUT FACTOR, IN SECONDS
000172' 000012                  ERRLVL: 10.             ;NUMBER OF ERRORS TOLERATED
```



```
000174' 113226                  SYNC:   113226          ;2 BYTES OF SYNC CHARACTER
000176' 001134'                         ONEREC          ;ADDRESS OF WHERE 240 GOES
000200' 001456'                         ONETMT          ;ADDRESS OF WHERE 240 GOES

000202' 000000                  SPARE1: 0
000204' 000000                  SPARE2: 0
000206' 000000                  SPARE3: 0
000210' 000000                  SPARE4: 0
                                ;
                                ;
                                ;PHYSICAL LINE TABLE. BYTE 0 CONTAINS THE PHYSICAL
                                ;LINE # FOR THE FIRST LINE TO BE RUN. BYTE 1 WILL BE
                                ;THE PHYSICAL LINE RUN SECOND, ETC.
                                ;EACH BYTE IN THE TABLE HAS THE FOLLOWING BIT MEANINGS:
                                ;BIT 0-3        PHYSICAL LINE #
                                ;BIT 4          0 FOR POINT-TO POINT, 1 FOR MULTI-DROP
                                ;BIT 5          0 FOR SLAVE, 1 FOR MASTER (DERIVED FROM SR1)
                                ;BIT 6          0 FOR HALF DUPLEX, 1 FOR FULL
                                ;BIT 7          0 DO NOT TEST, 1 TEST (DERIVED FROM DVID1)
000212'     000                 PHYTAB: .BYTE   0
000213'     001                         .BYTE   1
000214'     002                         .BYTE   2
000215'     003                         .BYTE   3
000216'     004                         .BYTE   4
000217'     005                         .BYTE   5
000220'     006                         .BYTE   6
000221'     007                         .BYTE   7
000222'     010                         .BYTE   10
000223'     011                         .BYTE   11
000224'     012                         .BYTE   12
000225'     013                         .BYTE   13
000226'     014                         .BYTE   14
000227'     015                         .BYTE   15
000230'     016                         .BYTE   16
000231'     017                         .BYTE   17

                                ;
                                ;
                                ;ERRTAB ,KEEPS TRACK OF RETRY ERROR COUNTS FOR
                                ;EACH LINE
                                ;
000232'     000                 ERRTAB: .BYTE   0
000233'     000                         .BYTE   0
000234'     000                         .BYTE   0
000235'     000                         .BYTE   0
000236'     000                         .BYTE   0
000237'     000                         .BYTE   0
000240'     000                         .BYTE   0
000241'     000                         .BYTE   0
000242'     000                         .BYTE   0
000243'     000                         .BYTE   0
000244'     000                         .BYTE   0
000245'     000                         .BYTE   0
000246'     000                         .BYTE   0
000247'     000                         .BYTE   0
000250'     000                         .BYTE   0
```

```
000251' 000                     .BYTE   0
                        ;
                        ;
                        ;ECOUNT TABLE KEEPS TRACK OF # OF BAD MSG'S
000252' 000             ECOUNT: .BYTE   0
000253' 000                     .BYTE   0
000254' 000                     .BYTE   0
000255' 000                     .BYTE   0
000256' 000                     .BYTE   0
000257' 000                     .BYTE   0
000260' 000                     .BYTE   0
000261' 000                     .BYTE   0
000262' 000                     .BYTE   0
000263' 000                     .BYTE   0
000264' 000                     .BYTE   0
000265' 000                     .BYTE   0
000266' 000                     .BYTE   0
000267' 000                     .BYTE   0
000270' 000                     .BYTE   0
000271' 000                     .BYTE   0
                        ;
                        ;
                        ;MULTI-DROP STATION ADDRESS TABLE
                        ;BYTE 0 IS THE MULTI-DROP STATION ADDRESS
                        ;FOR LOGICAL LINE 0, BYTE 1 IS FOR LOGICAL LINE 1, ETC.
                        ;THIS ENTRY ONLY HAS MEANING IF CORRESPONDING
                        ;BYTE IN PHYSICAL LINE TABLE HAS BITS 4 AND 7 SET
                        ;AND BIT 5 OFF, IE MULTI-DROP SLAVE TO BE TESTED
000272' 000             MDATAB: .BYTE   0
000273' 001                     .BYTE   1
000274' 002                     .BYTE   2
000275' 003                     .BYTE   3
000276' 004                     .BYTE   4
000277' 005                     .BYTE   5
000300' 006                     .BYTE   6
000301' 007                     .BYTE   7
000302' 010                     .BYTE   10
000303' 011                     .BYTE   11
000304' 012                     .BYTE   12
000305' 013                     .BYTE   13
000306' 014                     .BYTE   14
000307' 015                     .BYTE   15
000310' 016                     .BYTE   16
000311' 017                     .BYTE   17
                        ;
                        ;MULTI-DROP STATION ADDRESS MENU TABLE
                        ;THIS TABLE INDICATES TO ANY LINE RUNNING AS
                        ;MULTI-DROP MASTER, WHICH STATION ADDRESS TO
                        ;ATTEMPT TO EXERCISE ON THE MULTI-DROP LINE
                        ;IT ALLOWS UP TO 15 DIFFERENT STATIONS TO BE
                        ;TESTED ON A LINE.  EACH BYTE WHICH IS NOT=-1
                        ;INDICATES A STATION ADDRESS TO BE TESTED, EACH
                        ;STATION IS TESTED, BY THE ORDER OF THE TABLE,
                        ;UNTIL THE FIRST OCCURANCE OF A -1 (BINARY 11111111)
                        ;
000312' 377             MDMTAB: .BYTE   -1
```



```
000313' 377                     .BYTE   -1
000314' 377                     .BYTE   -1
000315' 377                     .BYTE   -1
000316' 377                     .BYTE   -1
000317' 377                     .BYTE   -1
000320' 377                     .BYTE   -1
000321' 377                     .BYTE   -1
000322' 377                     .BYTE   -1
000323' 377                     .BYTE   -1
000324' 377                     .BYTE   -1
000325' 377                     .BYTE   -1
000326' 377                     .BYTE   -1
000327' 377                     .BYTE   -1
000330' 377                     .BYTE   -1
000331' 377                     .BYTE   -1
                        ;
                        ;
                        ;
000332' 016700  177456  START:  MOV     DVID1,R0        ;GET DEVICES TO BE TESTED
000336' 001002                  BNE     1$              ;ANY ACTIVE?
000340' 104403  000000'         END$,BEGIN              ;
000344' 012705  000000'  1$:    MOV     #BEGIN,R5       ;PREPARE FOR NXT JSR
000350' 004567  000000G         JSR     R5,DEVSET       ;GO ASK FOR PARAMETERS
000354' 012767  000200  004214  MOV     #200,MASK       ;SET UP MASK
000362' 012701  000232'  2$:    MOV     #PHYTAB+16.,R1  ;GET END OF DEVICE TABLE
000366' 006300           3$:    ASL     R0              ;CHECK EACH BIT
000370' 103003                  BCC     4$              ;BR IF NOT ON
000372' 156741  004200          BISB    MASK,-(R1)      ;SET THIS LINE ACTIVE
000376' 000402                  BR      5$
000400' 146741  004172  4$:     BICB    MASK,-(R1)      ;CLEAR THIS LINE
000404' 105061  000040  5$:     CLRB    32.(R1)         ;CLEAR ECOUNT TABLE
000410' 020127  000211'         CMP     R1,#PHYTAB-1    ;DONE?
000414' 001364                  BNE     3$              ;BR IF NOT
000416' 026727  004154  000040  CMP     MASK,#40        ;BEEN HERE BEFORE?
000424' 001406                  BEQ     6$              ;BR IF YES
000426' 012767  000040  004142  MOV     #40,MASK        ;SET UP MASK
000434' 016700  177356          MOV     SR1,R0          ;GET SLAVE/MASTER INFO
000440' 000750                  BR      2$              ;GO SET UP
000442' 005201  6$:     INC     R1              ;GET PHYTAB ADDR
000444' 016700  177336          MOV     ADDR,R0         ;GET DEVICE ADDR
000450' 105721  7$:     TSTB    (R1)+           ;DEVICE ACTIVE?
000452' 100002                  BPL     8$              ;BR IF NOT
000454' 012710  000002          MOV     #2,(R0) ;SET DATA TERM READY
000460' 062700  000010  8$:     ADD     #10,R0          ;BUMP R0
000464' 020127  000232'         CMP     R1,#PHYTAB+16.  ;DONE?
000470' 001367                  BNE     7$              ;BR IF NOT
000472' 005067  004076  9$:     CLR     MSGCNT          ;COUNTS WHEN TO DO SUMMARY
000476' 005767  000000G         TST     WAIT            ;LINES SET UP?
000502' 001405                  BEQ     RESTRT          ;BR IF YES
000504' 104407  000000'         BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
000510' 104407  000000'         BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
000514' 000766                  BR      9$              ;WAIT TILL WAIT IS CLEARED BY NET2
                        ;
                        ;
                        ;THROUGH-OUT THE PROGRAM MANY LOCATIONS ARE REFERENCED
                        ;OFF OF (R4).  THIS IS A MEANS OF KEEPING TWO SETS OF
```

```
                                        ;VARIABLES, ONE FOR EACH OF THE ACTIVE LINES.
                                        ;R4 ALWAYS CONTAINS A 0 IF THE PROGRAM IS CURRENTLY
                                        ;REFERENCING LINE 0 OR A 2 IF IT IS REFERENCING
                                        ;LINE 1.
                                        ;
000516' 005767  000000G         RESTRT: TST     PWRCNT          ;HAS THERE BEEN A POWER FAIL?
000522' 001402                          BEQ     1$              ;CONTINUE IF NOT,
000524' 104403  000000'                 END$,BEGIN              ;
000530' 005004                  1$:     CLR     R4              ;FIRST ACTIVE LINE
000532' 016706  177266                  MOV     SPOINT,R6       ;INIT STACK
000536' 005067  004206                  CLR     DROPR4          ;CLEAR ALL EXIT FLAGS
000542' 005067  004204                  CLR     DROPR4+2
000546' 005067  004202                  CLR     DTERR4
000552' 005067  004200                  CLR     DTERR4+2
000556' 005067  004176                  CLR     CRCR4
000562' 005067  004174                  CLR     CRCR4+2
000566' 005067  004172                  CLR     RCTOR4
000572' 005067  004170                  CLR     RCTOR4+2
000576' 005067  004166                  CLR     TXTOR4
000602' 005067  004164                  CLR     TXTOR4+2
000606' 005067  004162                  CLR     FDBKR4
000612' 005067  004160                  CLR     FDBKR4+2
000616' 005067  004156                  CLR     CHKR4
000622' 005067  004154                  CLR     CHKR4+2
                                        ;
000626' 005067  004114                  CLR     PASTIM          ;CLR PASS TIME FLAG
000632' 005067  004106                  CLR     BRKFLG          ;CLR FLAG THAT SAYS WE'RE IN BREAK
000636' 012767  000212' 003734          MOV     #PHYTAB,NXTLIN  ;POINTS TO NEXT LINE TO TEST
000644' 005067  003766                  CLR     LINACT          ;SHOW NO LINES ACTIVE
000650' 026727  003724  000232' GETNXT: CMP     NXTLIN,#PHYTAB+16.;DONE ALL LINES?
000656' 001440                          BEQ     DONE            ;BR IF SO
000660' 016700  003714                  MOV     NXTLIN,R0       ;GET ADDR OF NEXT LINE
000664' 010064  004644'                 MOV     R0,CURLIN(R4)   ;SAVE IT
000670' 105060  000020                  CLRB    20(R0)          ;CLEAR RETRY COUNT FOR THIS LINE
000674' 005267  003700                  INC     NXTLIN          ;BUMP FOR NEXT TIME
000700' 111000                          MOVB    (R0),R0         ;GET PHYSICAL LINE BYTE
000702' 002362                          BGE     GETNXT          ;BR BACK IF BIT 7 NOT ON
000704' 016701  177076                  MOV     ADDR,R1         ;SET UP TO CALC ADDR
000710' 004767  004070                  JSR     PC,GETADR       ;GO CALC ADDR
000714' 010164  004602'                 MOV     R1,DVASAV(R4)   ;SAVE DVA FOR THIS LINE
000720' 016701  177064                  MOV     VECTOR,R1       ;SET UP TO CALC VECTOR ADDR
000724' 004767  004054                  JSR     PC,GETADR       ;GO CALC
000730' 010164  004606'                 MOV     R1,VCTSAV(R4)   ;SAVE IT
000734' 005267  003676                  INC     LINACT          ;SHOW A LINE'S RUNNING
000740' 005064  004660'                 CLR     LOOP(R4)        ;CLEAR ITERATION COUNT
000744' 005064  004740'                 CLR     RCVTO(R4)       ;CLR REC TIME OUT COUNT
000750' 032700  000020                  BIT     #20,R0          ;MULTI DROP OR NOT?
A 000754' 001377                        BNE     MD              ;BR IF IT IS
000756' 000427                          BR      PP              ;BR IF NOT TO P TO P
                                        ;
                                        ;
000760' 005767  003652          DONE:   TST     LINACT          ;ANY LINES RUNNING?
000764' 001402                          BEQ     1$              ;BR IF NO
000766' 000167  003306                  JMP     TIMCHK          ;IF YES GO CHECK FOR TIME OUT
000772' 012701  000212'         1$:     MOV     #PHYTAB,R1      ;MUST CHECK IF MAYBE ALL LINES DROPPED
000776' 105721                  2$:     TSTB    (R1)+           ;IS THIS LINE STILL SELECTED?
```



```
001000' 100405                          BMI     3$              ;BR OUT IF YES
001002' 022701  000232'                 CMP     #PHYTAB+16.,R1  ;CHECKED ALL 16?
001006' 001373                          BNE     2$              ;BR BACK IF NO
001010' 104403  000000'                 END$,BEGIN              ;
                                        ;
001014' 012767  000001  003724  3$:     MOV     #1,PASTIM       ;SHOW IT IS TIME FOR ENDPAS
                                                                ;FOR BREAK LOOP
001022' 005767  003716                  TST     BRKFLG          ;ARE WE IN A BREAK LOOP?
001026' 001002                          BNE     4$              ;BR IF WE ARE
001030' 000167  003360                  JMP     EPASS           ;GO DO END OF PASS
                                        ;
001034'                         4$:
001034' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                        ;POINT TO POINT
001036' 012764  000001  004734' PP:     MOV     #1,FDFLAG(R4)   ;SAY WERE FULL DUPLEX
001044' 032700  000100                  BIT     #100,R0         ;HALF OR FULL DUPLEX?
001050' 001402                          BEQ     1$              ;BR IF HALF
001052' 000167  000512                  JMP     PPFD            ;JMP TO FULL DUPLEX ROUTINE
001056' 005064  004734'         1$:     CLR     FDFLAG(R4)      ;CORRECT FLAG CAUSE WERE NOT FD
001062' 032700  000040                  BIT     #40,R0          ;SLAVE OR MASTER?
001066' 001141                          BNE     PPHDM           ;BR IF MASTER
                                        ;
                                        ;POINT TO POINT, HALF DUPLEX, SLAVE
001070'                         PPHDS:
001070' 016464  004612' 004616'         MOV     AMSGBF(R4),RECADR(R4)  ;SET UP INPUT AREA
001076' 016764  000000G 004622'         MOV     MSGSIZ,RECCNT(R4)  ;SET UP COUNT
001104' 062764  000012  004622'         ADD     #12,RECCNT(R4)  ;CORRECT FOR HEADER AND CRC
001112' 016764  177052  004626'         MOV     TIMELN,RECTIM(R4)  ;HOW LONG TO WAIT
001120' 066764  177044  004626'         ADD     TIMELN,RECTIM(R4)       ;DOUBLE IT
001126' 004767  000766                  JSR     PC,REC          ;GO RECEIVE DATA
001132' 000462                          BR      FERR            ;BR TO ERROR ROUTINE
001134' 000401          ONEREC: BR      1$                      ;MAKE THIS A NOP=240 TO
001136' 000432                          BR      PPHDS2          ;DO ONE-WAY-IN (REC. ONLY)
001140' 012764  000000G 004670' 1$:     MOV     #MSGHED,TXADR1(R4)  ;GET MESS ADR
001146' 016764  000000G 004700'         MOV     MSGSIZ,TXCNT1(R4)  ;GET MSG SIZE
001154' 062764  000012  004700'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC'S
001162' 012764  000004  004704'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
001170' 012764  011120' 004674'         MOV     #FILL,TXADR2(R4)
001176' 004767  002704                  JSR     PC,CTS          ;GO GET CLEAR TO SEND
001202' 016764  176762  004710'         MOV     TIMELN,TMTTIM(R4)       ;GIVE 'EM 1/2 MINUTE
001210' 004767  002162                  JSR     PC,TMT          ;GO TRANSMIT
001214' 000423                          BR      DERR            ;BR TO ERROR ROUTINE
001216' 042774  000004  004602'         BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
001224' 005264  004660'         PPHDS2: INC     LOOP(R4)        ;COUNT AN ITERATION
001230' 026764  176732  004660'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH!
001236' 003314                          BGT     PPHDS           ;NO, GO DO ANOTHER TIME
001240' 005367  003372          LNDONE: DEC     LINACT
001244' 016403  004602'                 MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
001250' 042713  177775                  BIC     #177775,(R3)    ;KILL ALL BUT DTR
001254' 005063  000004                  CLR     4(R3)           ;KILL TRANSMITTER TOO
001260' 000167  177364                  JMP     GETNXT          ;YES DO NEXT LINE
                                        ;
                                        ;
001264' 042774  000004  004602' DERR:   BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
001272' 012703  177777                  MOV     #-1,R3          ;SET FLAG FOR WHICH RETURN
001276' 000401                          BR      FERR1
```

```
001300' 005003                  FERR:   CLR     R3              ;SET FLAG FOR WHICH RETURN
001302' 016401  004644'         FERR1:  MOV     CURLIN(R4),R1   ;GET LINE # ADDR
001306' 105261  000020                  INCB    20(R1)          ;COUNT AN ERROR
001312' 126167  000020  176652          CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
001320' 002003                          BGE     DROP            ;BR IF YES
001322' 005703                          TST     R3              ;DO A TMT OR DO A REC?
001324' 001703                          BEQ     ONEREC          ;BR IF TMT
001326' 000736                          BR      PPHDS2          ;GO TRY AGAIN
001330' 142711  000200          DROP:   BICB    #200,(R1)       ;DROP THE LINE FROM TEST
001334' 005264  004750'                 INC     DROPR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
001340' 104411  011124' 000000'         MSGN$,ADROP,BEGIN       ;ASCII MESSAGE CALL WITH COMMON HEADER
001346' 005767  003376                  TST     DROPR4
001352' 001402                          BEQ     1$
001354' 005004                          CLR     R4
001356' 000402                          BR      2$
001360' 012704  000002          1$:     MOV     #2,R4
001364' 005364  004750'         2$:     DEC     DROPR4(R4)
001370' 000723                          BR      LNDONE

                                ;
                                ;POINT TO POINT HALF DUPLEX MASTER
                                ;
001372'                         PPHDM:
001372' 012764  000000G 004670'         MOV     #MSGHED,TXADR1(R4)      ;GET MSG ADDRESS
001400' 016764  000000G 004700'         MOV     MSGSIZ,TXCNT1(R4)       ;GET MSG SIZE
001406' 062764  000012  004700'         ADD     #12,TXCNT1(R4)          ;CORRECT FOR CRC AND HEADER
001414' 012764  000004  004704'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
001422' 012764  011120' 004674'         MOV     #FILL,TXADR2(R4)
001430' 004767  002452                  JSR     PC,CTS                  ;GO GET CLEAR TO SEND
001434' 016764  176530  004710'         MOV     TIMELN,TMTTIM(R4)               ;GIVE 'EM 1/2 MINUTE
001442' 004767  001730                  JSR     PC,TMT                  ;GO TRANSMIT
001446' 000434                          BR      GERR                    ;ERROR RETURN
001450' 042774  000004  004602'         BIC     #4,@DVASAV(R4)          ;KILL REQ TO SEND
001456' 000401                  ONETMT: BR      1$                      ;MAKE THIS A NOP=240 FOR ONE
                                                                        ;WAY OUT, TMT ONLY, OTHER SIDE
                                                                        ;OF LINE MUST DO ONE WAY IN.
001460' 000417                          BR      PPHDM2                  ;USED FOR ONE WAY OUT ONLY
001462' 016464  004612' 004616' 1$:     MOV     AMSGBF(R4),RECADR(R4)   ;SET UP INPUT AREA
001470' 016764  000000G 004622'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
001476' 062764  000012  004622'         ADD     #12,RECCNT(R4)          ;CORRECT FOR HEADER AND CRC
001504' 016764  176460  004626'         MOV     TIMELN,RECTIM(R4)       ;GIVE EM 1/2 MINUTES
001512' 004767  000402                  JSR     PC,REC                  ;GO RECEIVE
001516' 000413                          BR      HERR                    ;ERROR RETURN
001520' 005264  004660'         PPHDM2: INC     LOOP(R4)                ;COUNT AN ITERATION
001524' 026764  176436  004660'         CMP     LPCNT,LOOP(R4)          ;DONE ENOUGH?
001532' 003317                          BGT     PPHDM                   ;BR IF NOT
001534' 000167  177500                  JMP     LNDONE
                                ;
                                ;
001540' 042774  000004  004602' GERR:   BIC     #4,@DVASAV(R4)          ;KILL REQ TO SEND
001546' 016401  004644'         HERR:   MOV     CURLIN(R4),R1           ;GET LINE # ADDRESS
001552' 105261  000020                  INCB    20(R1)                  ;COUNT AN ERROR
001556' 126167  000020  176406          CMPB    20(R1),ERRLVL           ;TOO MANY ERRORS?
001564' 002261                          BGE     DROP                    ;BR IF YES
001566' 000754                          BR      PPHDM2                  ;GO TRY AGAIN
                                ;
```



```
                                ;
                                ;POINT -TO-POINT FULL DUPLEX
                                ;
001570' 012700  000040          PPFD:   BIT     #40,R0          ;MASTER OR SLAVE?
001574' 001003                          BNE     1$              ;BR IF MASTER
001576' 012764  177777  004734'         MOV     #177777,FDFLAG(R4)      ;SHOW ITS SLAVE
001604' 004767  002276          1$:     JSR     PC,CTS          ;GO GET CLEAR TO SEND
001610' 016464  004612' 004616' PPFD1:  MOV     AMSGBF(R4),RECADR(R4)   ;GET INPUT AREA
001616' 016764  000000G 004622'         MOV     MSGSIZ,RECCNT(R4)       ;SET UP COUNT
001624' 062764  000012  004622'         ADD     #12,RECCNT(R4)  ;CORRECT FOR CRC AND HEADER
001632' 004767  000262                  JSR     PC,REC          ;GO GET RECEIVING STARTED
001636' 000500                          BR      JERR            ;REC ERROR RETURN
001640' 000441                          BR      FDRCDN          ;REC DONE
001642' 012764  000000G 004670'         MOV     #MSGHED,TXADR1(R4)      ;FULL DUPLEX RETURN FROM REC
                                                                ;GET TMT ADDR
001650' 016764  000000G 004700'         MOV     MSGSIZ,TXCNT1(R4)       ;GET TMT COUNT
001656' 062764  000012  004700'         ADD     #12,TXCNT1(R4)  ;CORRECT FOR HEADER AND CRC
001664' 012764  000004  004704'         MOV     #4,TXCNT2(R4)           ;SEND 4 FILLS AFTER MSG
001672' 012764  011120' 004674'         MOV     #FILL,TXADR2(R4)
001700' 004767  001472                  JSR     PC,TMT          ;GO TMT
001704' 000455                          BR      JERR            ;TMT ERROR RETURN
001706' 000412                          BR      1$              ;GOOD TMT DONE RETURN
001710' 012700  002106'                 MOV     #FDTO,R0        ;FULL DUP RETURN, SET UP FOR TIME OUT
001714' 016701  176250                  MOV     TIMELN,R1               ;GIVE EM 1 MINUTE
001720' 066701  176244                  ADD     TIMELN,R1
001724' 012702  000000                  MOV     #0,R2           ;SHOW ITS A REC WAIT
001730' 000167  002270                  JMP     TIMKEP          ;GO WAIT
                                ;
001734' 012764  000001  004720' 1$:     MOV     #1,TMTDON(R4)   ;SHOW TMT IS DONE
001742' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
                                ;
                                ;
001744' 005764  004720'         FDRCDN: TST     TMTDON(R4)      ;IS TMT DONE YET?
001750' 001020                          BNE     4$              ;BR IF YES
001752' 005264  004774'                 INC     FDBKR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
001756' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
001762' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
001766' 005767  003002                  TST     FDBKR4
001772' 001402                          BEQ     2$
001774' 005004                          CLR     R4
001776' 000402                          BR      3$
002000' 012704  000002          2$:     MOV     #2,R4
002004' 005364  004774'         3$:     DEC     FDBKR4(R4)
002010' 000755                          BR      FDRCDN          ;BR AND WAIT SOME MORE
002012' 005264  004660'         4$:     INC     LOOP(R4)        ;COUNT AN ITERATION
002016' 026764  176144  004660'         CMP     LPCNT,LOOP(R4)  ;DONE ENOUGH
002024' 003271                          BGT     PPFD1           ;NO, GO DO MORE
002026' 042774  000004  004602'         BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
002034' 000167  177200                  JMP     LNDONE
                                ;
                                ;
                                ;FD TMT ERROR
002040' 012764  000001  004720' JERR:   MOV     #1,TMTDON(R4)   ;DON'T BOTHER WAITING FOR TMT
002046' 012764  177777  004664'         MOV     #177777,ABORT(R4)       ;TELL INT ROUTINES TO GIVE UP
002054' 016401  004644'                 MOV     CURLIN(R4),R1   ;GET CURRENT LINE INFO
002060' 105261  000020                  INCB    20(R1)          ;COUNT ANOTHER ERROR
```

```
 892  002064' 126167  000020  176140          CMPB    20(R1),ERRLVL   ;TOO MANY ERRORS?
 893  002072' 002724                          BLT     5DRCDN          ;NO GO TRY AGAIN
 894  002074' 042774  000004  004602'         BIC     #4,@DVASAV(R4)  ;KILL REQ TO SEND
 895  002102' 000167  177222                  JMP     DROP            ;GIVE UP ON THIS LINE
 896                                  ;
 897                                  ;
 898                                  ;FULL DUPLEX RECEIVER TIMEOUTS COME HERE
 899  002106' 012764  002040' 004664' FDTO:   MOV     #JERR,RECRTN(R4)        ;SET UP CORRECT RETURN
 900  002114' 000167  001110                  JMP     RDTO            ;GO REPORT REC TIME OUT
 901                                  ;
```



```
 902                                  ;
 903                                  ;SUBROUTINE TO RECEIVE DATA
 904                                  ;CALLED WITH A JSR PC,REC
 905                                  ;AFTER SETTING LOCATIONS RECADR(R4),RECCNT(R4),RECTIM(R4),VCTSAV(R4),
 906                                  ;AND DVASAV(R4). RETURNS ON ERROR OR RETURNS
 907                                  ;2 BYTES PAST CALL IF NO ERROR
 908                                  ;
 909  002120' 012064  004664'         REC:    MOV     (R6)+,RECRTN(R4);SAVE RTN ADDR
 910  002124' 016401  004606'                 MOV     VCTSAV(R4),R1   ;GET VECTOR ADR
 911  002130' 012711  002230'                 MOV     #RECINT,(R1)    ;MOVE IN INT. ROUTINE ADR
 912  002134' 060421                          ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
 913  002136' 116711  175650                  MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
 914  002142' 005064  004640'                 CLR     ABORT(R4)       ;CLEAR ABORT FLAG
 915  002146' 012764  000001  004632'         MOV     #1,SYNCNT(R4)   ;MUST GET 2 SYN CHARACTERS
 916  002154' 005764  004734'                 TST     FDFLAG(R4)              ;FULL DUPLEX?
 917  002160' 001410                          BEQ     1$              ;BR IF NOT
 918  002162' 052774  000100  004602'         BIS     #100,@DVASAV(R4)        ;KICK OFF REC
 919  002170' 016402  004664'                 MOV     RECRTN(R4),R2   ;GET RETURN ADDRESS
 920  002174' 062702  000004                  ADD     #4,R2           ;FD RETURN
 921  002200' 010207                          MOV     R2,R7           ;RETURN
 922  002202' 012700  003230'         1$:     MOV     #RDTO,R0        ;RETURN ADDR
 923  002206' 016401  004626'                 MOV     RECTIM(R4),R1   ;HOW LONG TO WAIT
 924  002212' 012702  000000                  MOV     #0,R2           ;SHOWS ITS A REC WAIT
 925  002216' 052774  000100  004602'         BIS     #100,@DVASAV(R4);ENABLE INT AND READ
 926  002224' 000167  001774                  JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
 927                                  ;
 928                                  ;RECEIVER INTERRUPT ROUTINE
 929                                  ;
 930  002230' 000405                  RECINT: BR      1$
 931  002232' 010446                          MOV     R4,-(R6)        ;SAVE R4
 932  002234' 010346                          MOV     R3,-(R6)        ;SAVE R3
 933  002236' 012704  000002                  MOV     #2,R4           ;SET UP FOR SECOND LINE
 934  002242' 000403                          BR      2$
 935  002244' 010446                  1$:     MOV     R4,-(R6)        ;SAVE R4
 936  002246' 010346                          MOV     R3,-(R6)        ;SAVE R3
 937  002250' 005004                          CLR     R4              ;SET UP FOR FIRST LINE
 938  002252' 016403  004602'         2$:     MOV     DVASAV(R4),R3   ;GET THE DEVICE ADDRESS
 939  002256' 005764  004640'                 TST     ABORT(R4)       ;GIVE UP?
 940  002262' 001403                          BEQ     3$              ;BR IF NO
 941  002264' 042713  000140                  BIC     #140,(R3)       ;DISABLE REC INT.S
 942  002270' 000452                          BR      7$              ;AND LEAVE
 943  002272' 011364  004654'         3$:     MOV     (R3),ERRWD(R4)  ;SAVE STATUS
 944  002276' 016346  000002                  MOV     2(R3),-(R6)     ;SAVE CHARACTER AND ERR BITS
 945  002302' 005764  004632'                 TST     SYNCNT(R4)      ;ARE WE IN SYNC
 946  002306' 001413                          BEQ     5$              ;BR IF YES
 947  002310' 122716  000220                  CMPB    #220,(R6)       ;IS IT A DLE?
 948  002314' 001401                          BEQ     4$              ;BR IF YES
 949  002316' 000434                          BR      6$              ;NOT IN SYN SO THROW AWAY
 950  002320' 005064  004632'         4$:     CLR     SYNCNT(R4)      ;SHOW WE'RE IN SYN
 951  002324' 005764  004734'                 TST     FDFLAG(R4)      ;FD SLAVE?
 952  002330' 100002                          BPL     5$              ;BR IF NOT
 953  002332' 005064  004724'                 CLR     TSYNCT(R4)      ;LET TMT GO IF FULL DUP
 954  002336' 005767  175622          5$:     TST     REPORT          ;REPORT ERR NOW OR LATER
 955  002342' 001013                          BNE     51$             ;BR IF LATER
 956  002344' 105764  004654'                 TSTB    ERRWD(R4)       ;IS REC DONE BIT SET?
 957  002350' 100040                          BPL     HERR            ;BR IF NOT
```

```
 958  002352' 005716                          TST     (R6)            ;DATA ERROR?
 959  002354' 100006                          BPL     51$             ;BR IF NO DATA ERR
 960  002356' 012664  004654'                 MOV     (R6)+,ERRWD(R4) ;SAVE INFORMATION
 961  002362' 042764  100000  004654'         BIC     #100000,ERRWD(R4)       ;USE BIT 15 FOR FLAG TO RERR
 962  002370' 000431                          BR      RERR+2
 963  002372' 111674  004616'         51$:   MOVB    (R6),@RECADR(R4)        ;STORE THE CHAR IN BUFFER
 964  002376' 005264  004616'                 INC     RECADR(R4)              ;FOR NEXT TIME
 965  002402' 005364  004622'                 DEC     RECCNT(R4)      ;COUNT IT
 966  002406' 003406                          BLE     8$              ;BR IF DONE
 967  002410' 005726                  6$:     TST     (R6)+           ;FIX STACK
 968  002412' 052713  000100                  BIS     #100,(R3)       ; READ ANOTHER
 969  002416' 012603                  7$:     MOV     (R6)+,R3        ;RESTORE R3
 970  002420' 012604                          MOV     (R6)+,R4        ;RESTORE R4
 971  002422' 000002                          RTI                     ;RETURN
 972  002424' 016467  004650' 000014  8$:     MOV     RECD(R4),RPIRQ+2
 973  002432' 005726                          TST     (R6)+           ;FIX STACK
 974  002434' 042713  000140                  BIC     #140,(R3)       ;DISABLE REC INT,S
 975  002440' 012603                          MOV     (R6)+,R3        ;RESTORE REGS
 976  002442' 012604                          MOV     (R6)+,R4
 977  002444'                         RPIRQ:
 978                                          ;---------------------------------------------------------------
 979  002444' 000004  002654' 000000'         PIRQ$,RECDN,BEGIN               ; RTI AND QUEUE UP TO CONTINUE AT RECDN
 980                                          ;---------------------------------------------------------------
 981                                  ;
 982  002452' 005726                  RERR:   TST     (R6)+           ;FIX STACK
 983  002454' 012603                          MOV     (R6)+,R3        ;RESTORE REG
 984  002456' 042774  000140  004602'         BIC     #140,@DVASAV(R4)        ;RESET THE RECEIVER
 985  002464' 012767  002512' 000014          MOV     #2$,1$+2                ;SET UP TO PIRQ TO
 986  002472' 060467  000010                  ADD     R4,1$+2         ;THE RIGHT PLACE
 987  002476' 060467  000004                  ADD     R4,1$+2
 988  002502' 012604                          MOV     (R6)+,R4        ;RESTORE REG
 989  002504'                         1$:
 990                                          ;---------------------------------------------------------------
 991  002504' 000004  002512' 000000'         PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
 992                                          ;---------------------------------------------------------------
 993  002512' 005004                  2$:     CLR     R4              ;SHOW ITS LINE 0
 994  002514' 000402                          BR      3$              ;
 995  002516' 012704  000002                  MOV     #2,R4           ;SHOW ITS LINE 1
 996  002522' 016467  004602' 175320  3$:     MOV     DVASAV(R4),CSRA ;SET UP FOR ERR CALL
 997  002530' 016467  004654' 175314          MOV     ERRWD(R4),ACSR  ;SET UP FOR ERR CALL
 998  002536' 016401  004644'                 MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE BYTE
 999  002542' 111102                          MOVB    (R1),R2         ;GET VALUE OF LINE #
1000  002544' 042702  177760                  BIC     #177760,R2      ;THROW OTHER BITS AWAY
1001  002550' 005764  004654'                 TST     ERRWD(R4)               ;IS IT A DATA OR LINE CHANGE ERR
1002  002554' 100010                          BPL     DATAER          ;BR IF DATA ERR
1003  002556' 010267  175272                  MOV     R2,ASTAT        ;SET UP ERR INDICATION
1004  002562' 062767  002000  175264          ADD     #002000,ASTAT   ;SHOW WHICH TYPE
1005                                          ;***************************************************************
1006  002570' 104404  000000'                 ERROR$,BEGIN            ;DATA SET CHANGE
1007                                          ;***************************************************************
1008  002574' 104403                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
1009                                  ;
1010                                  ;
1011  002576' 010267  175252          DATAER: MOV     R2,ASTAT        ;SET UP ERR LINE #
1012  002602' 116467  004655' 175245          MOVB    ERRWD+1(R4),ASTAT+1;GET ERR TYPE
1013  002610' 042767  107400  175236          BIC     #107400,ASTAT   ;LEAVE ONLY ERR BITS
```



```
1014  002616' 005264  004754'                 INC     DTERR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
1015                                          ;***************************************************************
1016  002622' 104404  000000'                 ERROR$,BEGIN            ;DATA ERR
1017                                          ;***************************************************************
1018  002626' 005767  002122                  TST     DTERR4
1019  002632' 001402                          BEQ     1$
1020  002634' 005004                          CLR     R4
1021  002636' 000402                          BR      2$
1022  002640' 012704  000002          1$:     MOV     #2,R4
1023  002644' 005364  004754'          2$:     DEC     DTERR4(R4)
1024                                  ;
1025                                  ;CHECK  ASTAT IN ERROR MESSAGE
1026                                  ;0100XX = REC DATA PARITY ERROR
1027                                  ;0200XX = FRAMING ERROR
1028                                  ;0400XX = OVERRUN
1029                                  ;XX = PHYSICAL LINE #
1030                                  ;
1031  002650' 000167  000230                  JMP     RRET            ;GO DO AN ERROR EXIT
1032                                  ;
1033                                  ;
1034                                  ;
1035  002654' 005004                  RECDN:  CLR     R4              ;SHOW WE'RE LINE 0
1036  002656' 000402                          BR      1$
1037  002660' 012704  000002                  MOV     #2,R4           ;SHOW WE'RE LINE 1
1038  002664' 042774  177771  004602'  1$:     BIC     #177771,@DVASAV(R4)     ;SHUT UP DEVICE
1039  002672' 005064  004740'                 CLR     RCVTO(R4)       ;RESET TIME OUT RETRY COUNTER
1040  002676' 005064  005040'                 CLR     RETTAB(R4)      ;KILL IMPENDING TIMEOUT
1041  002702' 005767  000000G         3$:     TST     CKMSG           ;IS CHECK MSG ROUTINE BUSY?
1042  002706' 001420                          BEQ     6$              ;BR IF NOT
1043  002710' 005264  005000'                 INC     CHKR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
1044  002714' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR,...
1045  002720' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
1046  002724' 005767  002050                  TST     CHKR4
1047  002730' 001402                          BEQ     4$
1048  002732' 005004                          CLR     R4
1049  002734' 000402                          BR      5$
1050  002736' 012704  000002          4$:     MOV     #2,R4
1051  002742' 005364  005000'          5$:     DEC     CHKR4(R4)
1052  002746' 000755                          BR      3$
1053  002750' 016402  004612'          6$:     MOV     AMSGBF(R4),R2   ;GET REC DATA
1054  002754' 026762  177776G 000006          CMP     MSG-2,6(R2)     ;IS HEADER CRC OK?
1055  002762' 001407                          BEQ     7$              ;BR IF OK
1056  002764' 005767  175174                  TST     REPORT          ;REPORT ERR NOW?
1057  002770' 001055                          BNE     X5              ;BR IF LATER
1058  002772' 012767  003000  175054          MOV     #003000,ASTAT   ;SHOW ERR TYPE
1059  003000' 000466                          BR      RECERR
1060  003002'                         7$:
1061  003002' 117467  004644' 000040          MOVB    @CURLIN(R4),9$  ;GET LINE #
1062  003010' 042767  177760  000032          BIC     #177760,9$      ;LEAVE ONLY LINE BITS
1063  003016' 005767  175142                  TST     REPORT          ;REPORT BAD DATA?
1064  003022' 001403                          BEQ     8$              ;BR IF YES
1065  003024' 052767  100000  000016          BIS     #100000,9$      ;SET FLAG TO CKMSG
1066  003032' 062702  000010          8$:     ADD     #10,R2          ;GET ADDR OF MSG
1067  003036' 010267  000010                  MOV     R2,9$+2
1068  003042' 004567  000002G                 JSR     R5,CKMSG+2      ;GO CHECK MSG
1069  003046' 000000'                         BEGIN
```

```
1070  003050' 000000                  9$:     0
1071  003052' 000000                          0
1072  003054' 000417                          BR      X4              ;BAD RETURN
1073  003056' 066702  000000G                 ADD     MSGSIZ,R2       ;POINT IT TO MSG CRC, DATA WAS OK
1074  003062' 126722  000000G                 CMPB    CRCMSG,(R2)+    ;CHECK 1ST CRC BYTE
1075  003066' 001025                          BNE     CER
1076  003070' 126722  000001G                 CMPB    CRCMSG+1,(R2)+  ;CHECK SECOND
1077  003074' 001022                          BNE     CER
1078  003076' 062764  000002  004664'         ADD     #2,RECRTN(R4)   ;GOOD RETURN
1079  003104' 005064  005040'         RRET:   CLR     RETTAB(R4)      ;KILL IMPENDING TIMEOUTS
1080  003110' 000174  004664'                 JMP     @RECRTN(R4)     ;AND GO BACK
1081                                  ;
1082                                  ;
1083  003114' 005767  175044          X4:     TST     REPORT          ;REPORT ERR NOW OR LATER?
1084  003120' 001001                          BNE     X5              ;BR IF LATER
1085  003122' 000770                          BR      RRET            ;OTHERWISE RETURN
1086                                  ;
1087  003124' 117402  004644'         X5:     MOVB    @CURLIN(R4),R2  ;GET CURRENT LINE #
1088  003130' 042702  177760                  BIC     #177760,R2      ;
1089  003134' 105262  000252'                 INCB    ECOUNT(R2)      ;COUNT AN ERROR FOR THAT LINE
1090  003140' 000761                          BR      RRET            ;AND RETURN
1091                                  ;
1092                                  ;
1093  003142' 005767  175016          CER:    TST     REPORT          ;REPORT NOW OR LATER?
1094  003146' 001366                          BNE     X5              ;BR IF LATER
1095  003150' 012767  004000  174676          MOV     #004000,ASTAT   ;INDICATE ERROR TYPE
1096  003156' 117402  004644'         RECERR: MOVB    @CURLIN(R4),R2  ;GENERATE LINE #
1097  003162' 042702  177760                  BIC     #177760,R2      ;
1098  003166' 050267  174662                  BIS     R2,ASTAT        ;ADD IN LINE #
1099  003172' 005264  004760'                 INC     CRCR4(R4)       ;SEE NOTE 1 IN FRONT OF LISTING
1100                                          ;*****************************************************************
1101  003176' 104404  000000'                 ERROR$,BEGIN            ;CRC ERROR
1102                                          ;*****************************************************************
1103  003202' 005767  001552                  TST     CRCR4
1104  003206' 001402                          BEQ     1$
1105  003210' 005004                          CLR     R4
1106  003212' 000402                          BR      2$
1107  003214' 012704  000002          1$:     MOV     #2,R4
1108  003220' 005364  004760'         2$:     DEC     CRCR4(R4)
1109  003224' 000167  177654                  JMP     RRET            ;BAD RETURN
1110                                  ;
1111                                  ;READ TIME OUT ROUTINE
1112                                  ;
1113  003230' 012764  177777  004640' RDTO:   MOV     #177777,ABORT(R4);TELL INT ROUTINES TO GIVE UP
1114  003236' 005264  004740'                 INC     RCVTO(R4)       ;COUNT ANOTHER RETRY
1115  003242' 026467  004740' 174722          CMP     RCVTO(R4),ERRLVL        ;ENOUGH TRYS?
1116  003250' 001043                          BNE     4$              ;BR IF NO
1117  003252' 017467  004602' 174572          MOV     @DVASAV(R4),ACSR;GET REC STATUS
1118  003260' 042774  177771  004602'         BIC     #177771,@DVASAV(R4)     ;KILL ALL BUT DTR
1119  003266' 016467  004602' 174554          MOV     DVASAV(R4),CSRA ;PUT ADDR IN FOR ERR CALL
1120  003274' 016401  004644'                 MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE BYTE
1121  003300' 111102                          MOVB    (R1),R2         ;GET VALUE OF PHY LINE #
1122  003302' 042702  177760                  BIC     #177760,R2      ;THROW OTHER BITS AWAY
1123  003306' 062702  001000                  ADD     #1000,R2        ;INDICATE REC TIMEOUT ERR
1124  003312' 010267  174536                  MOV     R2,ASTAT        ;SET UP FOR ER CALL
1125  003316' 005264  004764'                 INC     RCTOR4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
```



```
1126                                    ;*****************************************************************
1127  003322' 104404  000000'           ERROR$,BEGIN            ;REC TIMED OUT
1128                                    ;*****************************************************************
1129  003326' 005767  001432            TST     RCTOR4
1130  003332' 001402                    BEQ     1$
1131  003334' 005004                    CLR     R4
1132  003336' 000402                    BR      2$
1133  003340' 012704  000002    1$:     MOV     #2,R4
1134  003344' 005364  004764'   2$:     DEC     RCTOR4(R4)
1135  003350' 005064  004740'           CLR     RCVTO(R4)       ;RESET RETRY COUNT
1136  003354' 000174  004664'   3$:     JMP     @RECRTN(R4)     ;ERROR RTN
1137                                    ;
1138  003360' 016401  004644'   4$:     MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
1139  003364' 105361  000020            DECB    20(R1)          ;DEC ERROR COUNT TO CORRECT FOR
1140                                                            ;INSUING INC
1141  003370' 005364  004660'           DEC     LOOP(R4)        ;CORRECT FOR LATER CALLING THIS A GOOD RUN
1142  003374' 000767                    BR      3$              ;RETURN
1143                                    ;
1144                                    ;
1145                                    ;SUBROUTINE TO TRANSMIT DATA
1146                                    ;CALLED WITH A JSR PC,TMT
1147                                    ;AFTER SETTING LOCATIONS TXADR1(R1), TXADR2(R4), TXCNT1(R4),TXCNT2(R4), TMTIM(R4),
1148                                    ;VCTSAV(R4), DVASAV(R4), SUBROUTINE CTS MUST BE CALLED FIRST.
1149                                    ;RETURNS ON ERROR, RETURNS 2 BYTES PAST NORMAL RETURN
1150                                    ;IF NO ERROR
1151                                    ;
1152  003376' 012664  004714'   TMT:    MOV     (SP)+,TMTRET(R4)  ;SAVE RTN ADDRESS
1153  003402' 005064  004720'           CLR     TMTDON(R4)      ;INITIALIZE FLAG
1154  003406' 016401  004606'           MOV     VCTSAV(R4),R1   ;GET REC VECTOR ADDRESS
1155  003412' 062701  000004            ADD     #4,R1           ;MAKE IT TMT VECTOR
1156  003416' 012711  003546'           MOV     #TMTINT,(R1)    ;MOVE IN INT ROUTINE ADDRESS
1157  003422' 060421                    ADD     R4,(R1)+        ;CORRECT FOR WHICH LINE
1158  003424' 116711  174362            MOVB    BR1,(R1)        ;MOV IN PRIORITY LEVEL
1159  003430' 005064  004640'           CLR     ABORT(R4)       ;CLEAR TIME OUT ABORT FLAG
1160  003434' 012764  000005  004724'   MOV     #5,TSYNCT(R4)   ;SEND FIVE SYN CHARACTERS
1161  003442' 016403  004602'           MOV     DVASAV(R4),R3   ;GET DEVICE ADDRESS
1162  003446' 005764  004734'           TST     FDFLAG(R4)      ;ARE WE FULL DUPLEX?
1163  003452' 001417                    BEQ     2$              ;BR IF NOT, ELSE
1164  003454' 100003                    BPL     1$              ;SKIP IF MASTER
1165  003456' 012764  177777  004724'   MOV     #177777,TSYNCT(R4)      ;MAKE TMT WAIT FOR A REC'D SYNC
1166  003464' 012763  000100  000004    MOV     #100,4(R3)      ;SET TMT INT ENABLE
1167  003472' 012763  000177  000006    MOV     #177,6(R3)      ;SEND A FILL CHAR
1168  003500' 016402  004714'           MOV     TMTRET(R4),R2   ;GET BAD RET ADDR
1169  003504' 062702  000004            ADD     #4,R2           ;MAKE IT THE FULL DUPLEX RETURN
1170  003510' 010207                    MOV     R2,R7           ;AND RETURN
1171                                    ;
1172                                    ;
1173  003512' 012700  003772'   2$:     MOV     #TXTO,R0        ;TIME OUT RETURN ADDR
1174  003516' 016401  004710'           MOV     TMTTIM(R4),R1   ;HOW LONG TO WAIT
1175  003522' 012702  000004            MOV     #4,R2           ;SHOW ITS TRANSMITTER
1176  003526' 012763  000100  000004    MOV     #100,4(R3)      ;SET TMT INT ENABLE
1177  003534' 012763  000177  000006    MOV     #177,6(R3)      ;SEND A FILL CHAR
1178  003542' 000167  000456            JMP     TIMKEP          ;EXIT TO TIMKEP AND WAIT
1179                                  ;TMT INTEPRUPT ROUTINE
1180                                  ;
1181  003546' 000405                  TMTINT: BR      1$
```

```
1182  003550' 010446                          MOV     R4,-(SP)        ;SAVE R4
1183  003552' 010346                          MOV     R3,-(SP)        ;SAVE R3
1184  003554' 012704  000002                  MOV     #2,R4           ;SET TO LINE 1
1185  003560' 000403                          BR      2$
1186  003562' 010446                  1$:     MOV     R4,-(SP)        ;SAVE R4
1187  003564' 010346                          MOV     R3,-(SP)        ;SAVE R3
1188  003566' 005004                          CLR     R4              ;SET TO LINE 0
1189  003570' 005764  004644'          2$:    TST     ABORT(R4)       ;GIVE UP?
1190  003574' 100417                          BMI     TIEXIT          ;BR IF YES
1191  003576' 016403  004602'                 MOV     DVASAV(R4),R3   ;GET DVA
1192  003602' 005764  004700'                 TST     TXCNT1(R4)      ;FIRST BUFFER TMT'ED?
1193  003606' 001015                          BNE     FTBUF           ;BR IF NOT
1194  003610' 005764  004704'                 TST     TXCNT2(R4)      ;SECOND BUFFER TMT'ED?
1195  003614' 001434                          BEQ     TIDONE          ;BR IF YES
1196  003616' 117463  004674' 000006          MOVB    @TXADR2(R4),6(R3)  ;SEND NEXT BYTE
1197  003624' 005264  004674'                 INC     TXADR2(R4)      ;MOV ADDRESS POINTER
1198  003630' 005364  004704'                 DEC     TXCNT2(R4)      ;COUNT IT
1199  003634' 012603                  TIEXIT: MOV     (R6)+,R3        ;RESTORE R3
1200  003636' 012604                          MOV     (R6)+,R4        ;RESTORE R4
1201  003640' 000002                          RTI                     ;RETURN
1202  003642' 005764  004724'          FTBUF: TST     TSYNCT(R4)      ;SENT ENOUGH FILL CHARS?
1203  003646' 001407                          BEQ     2$              ;BR IF YES
1204                                                                  ;IF NEGITIVE
1205  003650' 100402                          BMI     1$              ;SEND TILL RECINT CLEARS TSYNCT
1206  003652' 005364  004724'                 DEC     TSYNCT(R4)      ;ELSE COUNT IT
1207  003656' 112763  000177  000006  1$:     MOVB    #177,6(R3)      ;SEND IF YES
1208  003664' 000763                          BR      TIEXIT          ;AND LEAVE
1209  003666' 117463  004670' 000006  2$:     MOVB    @TXADR1(R4),6(R3)  ;SEND NEXT BYTE
1210  003674' 005264  004670'                 INC     TXADR1(R4)      ;MOVE ADDRESS POINTER
1211  003700' 005364  004700'                 DEC     TXCNT1(R4)      ;COUNT IT
1212  003704' 000753                          BR      TIEXIT          ;AND LEAVE
1213                                  ;
1214                                  ;ALL CHAR'S HAVE BEEN TMT'ED
1215  003706' 042763  000100  000004  TIDONE: BIC     #100,4(R3)      ;KILL TMT'ER
1216  003714' 012603                          MOV     (R6)+,R3        ;RESTORE R3
1217  003716' 005064  005044'                 CLR     RETTAB+4(R4)    ;KILL IMPENDING TIMEOUT
1218  003722' 062764  000002  004714'         ADD     #2,TMTRET(R4)   ;MAKE IT A GOOD RETURN
1219  003730' 005704                          TST     R4              ;LINE 0 OR 1
1220  003732' 001004                          BNE     1$              ;BR IF LINE 1
1221  003734' 012604                          MOV     (R6)+,R4        ;RESTORE R4
1222                                          ;---------------------------------------------------------------
1223  003736' 000004  003754' 000000'         PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
1224                                          ;---------------------------------------------------------------
1225  003744' 012604                  1$:     MOV     (R6)+,R4        ;RESTORE R4
1226                                          ;---------------------------------------------------------------
1227  003746' 000004  003762' 000000'         PIRQ$,3$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 3$
1228                                          ;---------------------------------------------------------------
1229  003754' 005004                  2$:     CLR     R4              ;SET TO LINE 0
1230  003756' 000174  004714'                 JMP     @TMTRET(R4)     ;RETURN
1231  003762' 012704  000002          3$:     MOV     #2,R4           ;SET TO LINE 1
1232  003766' 000174  004714'                 JMP     @TMTRET(R4)     ;RETURN
1233                                  ;
1234                                  ;
1235                                  ;ROUTINE IF TRANSMIT TIMES OUT
1236                                  ;
1237  003772' 012764  177777  004640' TXTO:   MOV     #177777,ABORT(R4)       ;TELL INT ROUTINES TO GIVE UP
```



```
1238  004000' 016467  004602' 174042          MOV     DVASAV(R4),CSRA ;GET ADDR OF STATUS
1239  004006' 062767  000004  174034          ADD     #4,CSRA         ;MAKE IT TMT CSR
1240  004014' 017767  174030  174030          MOV     @CSRA,ACSR      ;SAVE STATUS
1241  004022' 005077  174022                  CLR     @CSRA           ;KILL TRANSMISSION AND INT
1242  004026' 016401  004644'                 MOV     CURLIN(R4),R1   ;GET ADDR OF CUR LINE
1243  004032' 111102                          MOVB    (R1),R2         ;GET VALUE
1244  004034' 042702  177760                  BIC     #177760,R2      ;LEAVE ONLY LINE #
1245  004040' 062702  005000                  ADD     #5000,R2        ;INDICATE TMT TIMEOUT
1246  004044' 010267  174004                  MOV     R2,ASTAT        ;SET UP FOR ERROR CALL
1247  004050' 005264  004770'                 INC     TXTOP4(R4)      ;SEE NOTE 1 IN FRONT OF LISTING
1248                                          ;***************************************************************
1249  004054' 104404  000000'                 ERROR$,BEGIN            ;TMT TIMEOUT
1250                                          ;***************************************************************
1251  004060' 005767  000704                  TST     TXTOP4
1252  004064' 001402                          BEQ     1$
1253  004066' 005004                          CLR     R4
1254  004070' 000402                          BR      2$
1255  004072' 012704  000002          1$:     MOV     #2,R4
1256  004076' 005364  004770'          2$:    DEC     TXTOR4(R4)
1257  004102' 000174  004714'                 JMP     @TMTRET(R4)     ;BAD RETURN
1258                                  ;
1259                                  ;
1260                                  ;
1261                                  ;
1262                                  ;
1263                                  ;SUB GETS CLEAR TO SEND FOR LINE IN DVASAV(R4) AND VCTSAV(R4)
1264
1265  004106' 012664  004730'          CTS:   MOV     (R6)+,CTSRET(R4)        ;SAVE RET ADDRESS
1266  004112' 016401  004606'                 MOV     VCTSAV(R4),R1   ;GET VECTOR ADDR
1267  004116' 012711  004154'                 MOV     #CTSINT,(R1)    ;MOV IN INT ROUTINE ADDR
1268  004122' 062421                          ADD     R4,(R1)+        ;SHOWS WHICH LINE
1269  004124' 116711  173662                  MOVB    BR1,(R1)        ;GET BR1 LEVEL
1270  004130' 032774  020000  004602'         BIT     #20000,@DVASAV(R4)  ;IS CLEAR TO SEND UP?
1271  004136' 001402                          BEQ     RTSSET          ;BR IF NOT
1272  004140' 016407  004730'                 MOV     CTSRET(R4),R7   ;RETURN
1273  004144' 012774  000006  004602' RTSSET: MOV     #6,@DVASAV(R4)  ;SET REQ TO SEND
1274  004152'                          NULL:
1275  004152' 104400                          EXIT$                   ;EXIT TO MONITOR, MODULE WAIT FOR INTERUPT.
1276                                  ;INT ROUTINE
1277  004154' 000403                  CTSINT: BR      1$              ;DIFFERENT PIRQ'S
1278                                          ;---------------------------------------------------------------
1279  004156' 000004  004172' 000000'         PIRQ$,2$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 2$
1280                                          ;---------------------------------------------------------------
1281  004164'                          1$:
1282                                          ;---------------------------------------------------------------
1283  004164' 000004  004200' 000000'         PIRQ$,3$,BEGIN          ; RTI AND QUEUE UP TO CONTINUE AT 3$
1284                                          ;---------------------------------------------------------------
1285  004172' 012704  000002          2$:    MOV     #2,R4           ;SET UP FOR LINE 1
1286  004176' 000401                          BR      4$
1287  004200' 005004                  3$:     CLR     R4              ;SET UP FOR LINE 0
1288  004202' 032774  020000  004602' 4$:     BIT     #20000,@DVASAV(R4)  ;IS CLEAR TO SEND SET?
1289  004210' 001755                          BEQ     RTSSET          ;GO TRY AGAIN IF NOT
1290  004212' 042774  000040  004602'         BIC     #40,@DVASAV(R4) ;DISABLE DATA SET CHG INT'S
1291  004220' 016407  004730'                 MOV     CTSRET(R4),R7   ;RETURN
1292                                  ;
1293                                  ;
```

```
                                ;
                                ;ALL ROUTINES COME HERE WHILE WAITING FOR I/O
                                ;ROUTINE EITHER KICKS OFF SECOND LINE, NOTIFYS
                                ;ROUTINES OF TIMEOUTS, OR JUST DOES BREAKS.
                                ;I/O ROUTINES THAT FINISH UP BEFORE TIMEOUT VALUE
                                ;MUST CLEAR LOCATION RETTAB(R4) +0 IF RECEIVER OR +4
                                ;IF TMT.
                                ;CALLED WITH A JMP TO TIMKEP AFTER LOADING
                                ; R0 = WHERE TO RETURN IF TIME OUT
                                ; R1 = HOW LONG TO WAIT (IN SECONDS)
                                ; R2 = 0 IF ITS A REC TIMEING OR 4 IF ITS A TMT WAIT
                                ; R4 = 0 FOR ACTIVE LINE 0, 2 FOR ACTIVE LINE 1
                                ;
                                ;
004224' 060402                  TIMKEP: ADD     R4,R2           ;FIGURE OUT WHICH TABLE ENTRY
004226' 010162  005030'                 MOV     R1,TIMTAB(R2)   ;STORE HOW LONG TO WAIT
004232' 067762  000000G 005030'         ADD     @TIME,TIMTAB(R2)        ;ADD PRESENT TIME
004240' 010462  005050'                 MOV     R4,R4TAB(R2)    ;SAVE WHICH ACTIVE LINE
004244' 010062  005040'                 MOV     R0,RETTAB(R2)   ;WHERE TO RET TO, PLUS
                                                                ;IF THIS ENTRY IN TABLE IS
                                                                ;0 THEN NO TIMEOUT WILL
                                                                ;OCCUR FOR THIS ENTRY SET
004250' 022767  000002  000360          CMP     #2,LINACT       ;ARE 2 LINES GOING?
004256' 001410                          BEQ     TIMCHK          ;BR IF YES
004260' 005704                          TST     R4              ;ARE WE LINE 0?
004262' 001402                          BEQ     1$              ; BR IF YES
004264' 005004                          CLR     R4              ; IF NO MAKE IT 0
004266' 000402                          BR      2$
004270' 012704  000002          1$:     MOV     #2,R4           ;WE WERE 0 SO NOW DO 1
004274' 000167  174350          2$:     JMP     GETNXT          ;GO DO OTHER LINE
                                ;
                                ;
004300' 012702  005040'         TIMCHK: MOV     #RETTAB,R2      ;GET ADDR OF TIME STATUSES
004304' 005722                  1$:     TST     (R2)+           ;ANY WAIT GOING ON?
004306' 001026                          BNE     4$              ;BR IF YES
004310' 020227  005050'         11$:    CMP     R2,#RETTAB+10   ;ARE WE DONE LOOKING
004314' 001373                          BNE     1$              ;BR BACK IF NO
004316' 005767  000424                  TST     PASTIM          ;TIME FOR END OF PASS
004322' 001402                          BEQ     2$              ;BR IF NO
004324' 000167  000064                  JMP     EPASS           ;GO REPORT END OF PASS
                                ;
004330' 005767  000410          2$:     TST     BRKFLG          ;ARE WE ALREADY IN A BREAK?
004334' 001401                          BEQ     3$              ;BR IF NO
004336' 104400                          EXIT$                   ;EXIT TO MONITOR. MODULE WAIT FOR INTERUPT.
                                ;
004340' 012767  000001  000376  3$:     MOV     #1,BRKFLG               ;SHOW WERE DOING BREAK
004346' 104407  000000'                 BREAK$,BEGIN            ;TEMPORARY RETURN TO MONITOR....
004352' 104407  000000'                 BREAK$,BEGIN            ;THEN CONTINUE AT NEXT INSTRUCTION.
004356' 005067  000362                  CLR     BRKFLG          ;SHOW WERE BACK
004362' 000746                          BR      TIMCHK
                                ;
004364' 005742                  4$:     TST     -(R2)           ;MOVE R2 BACK
004366' 077762  000000G 177770          CMP     @TIME,-10(R2)   ;IS TIME UP?
004374' 003002                          BGT     5$              ;BR IF YES
004376' 005722                          TST     (R2)+           ;BUMP R2 BACK AGAIN
004400' 000743                          BR      11$             ;GO CHECK OTHERS
```



```
                                ;
004402' 016204  000010          5$:     MOV     10(R2),R4       ;SET R4 CORRECTLY FOR WHICH LINE
004406' 011203                          MOV     (R2),R3         ;SAVE RETURN
004410' 005012                          CLR     (R2)            ;CLEAR THIS ENTRY
004412' 010307                          MOV     R3,R7           ;JMP TO TIME OUT ROUTINE
                                ;
                                ;
                                ;
                                ;EPASS DOES END OF PASS CALL AND PRINTS ERR SUMMARY
                                ;IF LOCATION REPORT IS NOT 0
                                ;
004414' 005767  173544          EPASS:  TST     REPORT          ;DO ERR SUM OR NOT
004420' 001461                          BEQ     5$              ;BR IF NO
004422' 005267  000146                  INC     MSGCNT          ;COUNT A PASS
004426' 026767  000142  173530          CMP     MSGCNT,REPORT   ;TIME FOR A MSG?
004434' 001053                          BNE     5$              ;IF NOT EVEN 8, SKIP MSG
004436' 005067  000132                  CLR     MSGCNT
004442' 012700  000212'                 MOV     #PHYTAB,R0      ;FIND WHICH LINES ARE RUNNING
004446' 012701  011220'                 MOV     #SUM1+4,R1      ;POINTS TO MSG
004452' 105710                  1$:     TSTB    (R0)            ;IS LINE IN USE?
004454' 100403                          BMI     2$              ;BR IF YES
004456' 012711  054130                  MOV     #54130,(R1)     ;PUT XX IN # OF ERRS
004462' 000427                          BR      4$
004464' 116002  000040          2$:     MOVB    32.(R0),R2      ;GET # OF ERRS
004470' 105060  000040                  CLRB    32.(R0)         ;RESET ERR COUNT
004474' 022702  000077                  CMP     #77,R2          ;ANY GOOD MSGS
004500' 002416                          BLT     3$              ;BR IF NO
004502' 010203                          MOV     R2,R3           ;PUT IN MSG
004504' 042703  177770                  BIC     #177770,R3      ;LEAVE ONLY LOW DIGIT
004510' 062703  000060                  ADD     #60,R3  ;MAKE IT ASCII
004514' 110361  000001                  MOVB    R3,1(R1)        ;STORE IT AWAY
004520' 006202                          ASR     R2              ;GET NEXT DIGIT
004522' 006202                          ASR     R2
004524' 006202                          ASR     R2
004526' 062702  000060                  ADD     #60,R2          ;MAKE IT ASCII
004532' 110211                          MOVB    R2,(R1) ;PUT IT IN MSG
004534' 000402                          BR      4$
004536' 012711  042102          3$:     MOV     #42102,(R1)     ;PUT IN AN 'BD' FOR BAD
004542' 005200                  4$:     INC     R0              ;BUMP LINE POINTER
004544' 062701  000010                  ADD     #10,R1          ;POINT TO NXT MSG
004550' 020027  000232'                 CMP     R0,#PHYTAB+20   ;DONE
004554' 001336                          BNE     1$              ;GO DO MORE
004556' 104411  004570' 000000'         MSGN$,MSUM,BEGIN        ;ASCII MESSAGE CALL WITH COMMON HEADER
                                ;
004564'                         ;
004564' 104402  000000'         5$:
                                        ENDPS$,REGIN            ;SIGNAL END OF PASS. CONTINUE AT RESTRT
                                ;
004570' 011130'                 MSUM:   SUM
004572' 177777                          -1
                                ;
004574' 000000                  MSGCNT: 0                       ;COUNTS END PASSES FOR MESSAGE CALL
004576' 000000                  MASK:   0                       ;USED FOR BIC'ING IN LOOPS
004600' 000000                  NXTLIN: 0                       ;ADR OF NEXT LIN
004602' 000000                  DVASAV: 0                       ;HOLDS DVA FOR ACTIVE L#0
004604' 000000                          0                       ;HOLDS DVA FOR ACTIVE L#1
```

```
1406  004606' 000000                VCTSAV: 0                       ;HOLDS VCT FOR ACTIVE L#0
1407  004610' 000000                        0                       ;HOLDS VECTOR FOR ACTIVE L#1
1408  004612' 005060'               AMSGBF: MSGBF0                  ;ADDRESS OF MSG BUFFER 0
1409  004614' 007100'                       MSGBF1                  ;ADDRESS OF MSG BUFFER 1
1410  004616' 000000                RECADR: 0                       ;CURRENT REC WORD POINTER 0
1411  004620' 000000                        0                       ;CURRENT REC WORD POINTER 1
1412  004622' 000000                RECCNT: 0                       ;REC BYTE COUNT FOR 0
1413  004624' 000000                        0                       ;REC BYTE COUNT FOR 1
1414  004626' 000000                RECTIM: 0                       ;TEMP HOLDS TIME OUT VALUE FOR A REC
1415  004630' 000000                        0                       ;SAME BUT FOR LINE 1
1416  004632' 000000                SYNCNT: 0                       ;COUNTS HOW MANY SYN CHARACTERS REC
1417  004634' 000000                        0                       ;SHOULD WAIT FOR.
1418  004636' 000000                LINACT: 0                       ;HOLD THE NUMBER OF LINES RUNNING,0-2
1419  004640' 000000                ABORT:  0                       ;ABORT FLAG FOR L#0
1420  004642' 000000                        0                       ;ABORT FLAG FOR L#1
1421  004644' 000000                CURLIN: 0                       ;POINTS TO STATUS BYTE FOR CURRENT L#0
1422  004646' 000000                        0                       ;POINTS TO STATUS BYTE FOR CURRENT L#1
1423  004650' 002654'               RECD:   RECDN                   ;WHERE TO PIRQ AFTER REC DONE
1424  004652' 002660'                       RECDN+4                 ;
1425  004654' 000000                EPRWD:  0                       ;HOLDS ERR WORDS TEMPORARILY
1426  004656' 000000                        0                       ;HOLDS ERR WORDS TEMPORARILY
1427  004660' 000000                LOOP:   0                       ;HOLDS LOOP COUNT FOR LINE # 0
1428  004662' 000000                        0                       ;HOLDS LOOP COUNT FOR LINE # 1
1429  004664' 000000                RECRTN: 0                       ;HOLDS RETURN ADDR FOR LINE #0
1430  004666' 000000                        0                       ;HOLDS RETURN ADDR FOR LINE #1
1431  004670' 000000                TXADR1: 0                       ;HOLDS 1ST ADDR TO TMT FROM, LINE 0
1432  004672' 000000                        0                       ;SAME FOR LINE 1
1433  004674' 000000                TXADR2: 0                       ;HOLDS 2ND ADDR TO TMT FROM, LINE 0
1434  004676' 000000                        0                       ;SAME FOR LINE 1
1435  004700' 000000                TXCNT1: 0                       ;HOLDS 1ST TMT COUNT FOR LINE 0
1436  004702' 000000                        0                       ;SAME FOR LINE 1
1437  004704' 000000                TXCNT2: 0                       ;HOLDS 2ND TMT COUNT FOR LINE 0
1438  004706' 000000                        0                       ;SAME FOR LINE 1
1439  004710' 000000                TMTTIM: 0                       ;HOLDS TIME OUT TIME FOR LINE 0
1440  004712' 000000                        0                       ;SAME FOR LINE 1
1441  004714' 000000                TMTRET: 0                       ;HOLDS RTN ADDR FOR TMT, LINE 0
1442  004716' 000000                        0                       ;SAME FOR LINE 1
1443  004720' 000000                TMTDON: 0                       ;FLAG FOR WHEN A TMT IS DONE
1444  004722' 000000                        0                       ;SAME FOR OTHER LINE
1445  004724' 000000                TSYNCT: 0                       ;HOLDS # OF SYN CHAR TO SEND, LINE 0
1446  004726' 000000                        0                       ;SAME FOR LINE 1
1447  004730' 000000                CTSRET: 0                       ;HOLDS RETURN ADDR FOR CTS
1448  004732' 000000                        0                       ;SAME FOR LINE 1
1449  004734' 000000                FDFLAG: 0                       ;0 IF HALF DUPLEX, 1 IF FULL
1450  004736' 000000                        0
1451  004740' 000000                RCVTO:  0                       ;COUNTS REC TIME OUT RETRYS
1452  004742' 000000                        0
1453  004744' 000000                BRKFLG: 0                       ;IS A ONE DURING BREAKS
1454  004746' 000000                PASTIM: 0                       ;HAS A 1 WHEN ITS TIME FOR END OF PASS
1455  004750' 000000                DROPR4: 0       ;EXIT FLAGS
1456  004752' 000000                        0
1457  004754' 000000                DTERR4: 0
1458  004756' 000000                        0
1459  004760' 000000                CRCR4:  0
1460  004762' 000000                        0
1461  004764' 000000                RCTOR4: 0
```



```
1462  004766' 000000                        0
1463  004770' 000000                TXTOR4: 0
1464  004772' 000000                        0
1465  004774' 000000                FDBKR4: 0
1466  004776' 000000                        0
1467  005000' 000000                CHKR4:  0
1468  005002' 000000                        0
1469
1470                                ;
1471                                ;
1472                                ;R1 HAS ADDR OR VECTOR
1473                                ;R0 HAS LINE # IN LOW BITS, RETURNS WITH ADDR OR VECTOR IN R1
1474  005004'                       GETADR:
1475  005004' 010046                        MOV     R0,-(R6)        ;SAVE R0
1476  005006' 042700  177760                BIC     #177760,R0      ;SAVE ONLY LINE #
1477  005012' 001404                1$:     BEQ     2$              ;DONE
1478  005014' 062701  000010                ADD     #10,R1          ;BUMP TO NEXT LINE
1479  005020' 005300                        DEC     R0              ;ADDED ENOUGH TIMES?
1480  005022' 000773                        BR      1$              ;GO BACK
1481  005024' 012600                2$:     MOV     (R6)+,R0        ;RESTORE R0
1482  005026' 000207                        RTS     PC              ;RETURN WITH DVA IN R1
1483                                ;
1484                                ;
1485  005030' 000000                TIMTAB: 0                       ;WHEN TO TIME OUT FOR REC,LINE 0
1486  005032' 000000                        0                       ;WHEN TO TIME OUT FOR REC,LINE 1
1487  005034' 000000                        0                       ;WHEN TO TIME OUT FOR TXT,LINE 0
1488  005036' 000000                        0                       ;WHEN TO TIME OUT FO TXT,LINE 1
1489  005040' 000000                RETTAB: 0                       ;WHERE TO RETURN FOR REC LINE 0
1490  005042' 000000                        0                       ;WHERE TO RETURN FOR REC LINE 1
1491  005044' 000000                        0                       ;WHERE TO RETURN FOR TXT LINE 0
1492  005046' 000000                        0                       ;WHERE TO RETURN FOR TXT LINE 1
1493
1494  005050' 000000                R4TAB:  0                       ;HOLD R4 FOR RETURNS
1495  005052' 000000                        0
1496  005054' 000000                        0
1497  005056' 000000                        0
1498                                ;
1499                                ;
1500  005060' 001010                MSGBF0: .BLKW   520.            ;REC BUFFER FOR LINE 0
1501  007100' 001010                MSGBF1: .BLKW   520.            ;REC BUFFER FOR LINE 1
1502  011120' 177777                FILL:   177777                  ;4 CHARACTERS OF FILLER BYTES
1503  011122' 177777                        177777
1504  011124' 011416'               ADROP:  MDROP
1505  011126' 177777                        -1
1506                                .EVEN
1507  011130' 005015  044523  041516 SUM:   .ASCII  <15><12>'SINCE LAST MSG PER LINE, IN OCTAL '<15><12>
1508  011136' 020105  040514  052123
1509  011144' 046440  043523  050040
1510  011152' 051105  046040  047111
1511  011160' 026105  044440  020116
1512  011166' 041517  040524  020114
1513  011174' 005015
1514  011176' 044514  042516  021440         .ASCII  'LINF #  FRRS'<15><12>
1515  011204' 020040  051105  051522
1516  011212' 005015
1517  011214' 030060  026455  030060 SUM1:   .ASCII  '00--00'<15><12>
```

```
 1518  011222' 005015
 1519  011224' 030460  026455  030060          .ASCII  '01--00'<15><12>
 1520  011232' 005015
 1521  011234' 031060  026455  030060          .ASCII  '02--00'<15><12>
 1522  011242' 005015
 1523  011244' 031460  026455  030060          .ASCII  '03--00'<15><12>
 1524  011252' 005015
 1525  011254' 032060  026455  030060          .ASCII  '04--00'<15><12>
 1526  011262' 005015
 1527  011264' 032460  026455  030060          .ASCII  '05--00'<15><12>
 1528  011272' 005015
 1529  011274' 033060  026455  030060          .ASCII  '06--00'<15><12>
 1530  011302' 005015
 1531  011304' 033460  026455  030060          .ASCII  '07--00'<15><12>
 1532  011312' 005015
 1533  011314' 030061  026455  030060          .ASCII  '10--00'<15><12>
 1534  011322' 005015
 1535  011324' 030461  026455  030060          .ASCII  '11--00'<15><12>
 1536  011332' 005015
 1537  011334' 031061  026455  030060          .ASCII  '12--00'<15><12>
 1538  011342' 005015
 1539  011341' 031461  026455  030060          .ASCII  '13--00'<15><12>
 1540  011352' 005015
 1541  011354' 032061  026455  030060          .ASCII  '14--00'<15><12>
 1542  011362' 005015
 1543  011364' 032461  026455  030060          .ASCII  '15--00'<15><12>
 1544  011372' 005015
 1545  011374' 033061  026455  030060          .ASCII  '16--00'<15><12>
 1546  011402' 005015
 1547  011404' 033461  026455  030060          .ASCII  '17--00'<15><12>
 1548  011412' 005015
 1549
 1550  011414'    000                          .BYTE   0
 1551          011416'                         .EVEN
 1552                                  ;
 1553  011416' 044514  042516  053440  MDROP:  .ASCIZ  'LINE WAS DROPPED'
 1554  011424' 051501  042040  047522
 1555  011432' 050120  042105     000
 1556          000001                  .END
```



```
ABORT   004640R          889*   914*   939   1113*  1159*  1189   1237*  1419#
ACSR    000052R          477#   997*  1117*  1240*
ADDR    000006R          456#   651    705
ADROP   011124R          790   1504#
AMSGBF  004612R          748    817    843   1053   1408#
ASB     000056R          480#
ASTAT   000054R          479#  1003*  1004*  1011*  1012*  1013*  1058*  1095*  1098*  1124*  1246*
AWAS    000060R          481#
BDCNV = ******  G          1#
BEGIN   000000R          453#   632    633    661    662    675    727    790    871    872    979    991   1006
                        1016   1044   1045   1069   1101   1127   1223   1227   1249   1279   1283   1340   1341
                        1392   1396
BIT0  = 000001           490#
BIT1  = 000002           490#
BIT10 = 002000           490#
BIT11 = 004000           490#
BIT12 = 010000           490#
BIT13 = 020000           490#
BIT14 = 040000           490#
BIT15 = 100000           490#
BIT2  = 000004           490#
BIT3  = 000010           490#
BIT4  = 000020           490#
BIT5  = 000040           490#
BIT6  = 000100           490#
BIT7  = 000200           490#
BIT8  = 000400           490#
BIT9  = 001000           490#
BREAK$= 104407           490#   661    662    871    872   1044   1045   1340   1341
BRKFLG  004744R          694*   731   1335  1339*  1342*  1453#
BR1     000012R          458#   913   1158   1269
BR2     000013R          459#
CDATA$= 104414           490#
CER     003142R         1075   1077   1093#
CHKR4   005000R          690*   691*  1043*  1046   1051*  1467#
CKMSG = ******  G        490#  1041   1068
CRCMSG= ******  G        490#  1074   1076
CRCR4   004760R          682*   683*  1099*  1103   1108*  1459#
CSRA    000050R          475#   996*  1119*  1238*  1239*  1240   1241*
CTS     004106R          762    808    842   1265#
CTSINT  004154R         1267   1277#
CTSRET  004730R         1265*  1272   1291   1447#
CURLIN  004644R          700*   781    830    890    998   1061   1087   1096   1120   1138   1242   1421#
DATAER  002576R         1002   1011#
DATCK$= 104417           490#
DATER$= 104405           490#
DERR    001264R          765    777#
DEVSET= ******  G        491#   634
DONE    000760R          698    719#
DROP    001330R          784    788#   833    895
DROPR4  004750R          678*   679*   789*   791    796*  1455#
DTERR4  004754R          680*   681*  1014*  1018   1023*  1457#
DVASAV  004602R          707*   766*   771    777*   812*   829*   883*   894*   918*   925*   938   984*   996
                        1038*  1117   1118*  1119   1161   1191   1238   1270   1273*  1288   1290*  1404#
DVIDI   000014R          460#   630
ECOUNT  000252R          560#  1089*
```

```
ENDPS$= 104402          490#  1396
END$  = 104403          490#   632    675    727
EPASS   004414R         733   1333   1361#
ERRCNT  000030R         467#
ERRLVL  000172R         499#   783    832    892   1115
ERRN$ = 104410          490#
ERRORS= 104404          490#  1006   1016   1101   1127   1249
ERRTAB  000232R         541#
ERRWD   004651R         943*   956    960*   961*   997   1001   1012   1425#
EXIT$ = 104400          490#   736    865   1008   1275   1337
FDBKR4  004774R         688*   689*   870*   873    878*  1465#
FDFLAG  004734R         738*   742*   841*   916    951   1162   1449#
FDRCDN  001744R         848    868#   879    893
FDTO    002106R         858    899#
FERR    001300R         754    780#
FERR1   001302R         779    781#
FILL    011120R         761    807    854   1502#
FTBUF   003642R        1193   1202#
GERR    001540R         811    829#
GETADR  005004R         706    709   1474#
GETNXT  000650R         697#   704    774   1323
GETPA$= 104413          490#
GWBUF$= 104412          490#
HERR    001546R         822    830#
INIT    000022R         464#
JERR    002040R         847    856    888#   899
LINACT  004636R         696*   711*   719    770*  1316   1410#
LNDONE  001240R         770#   797    826    884
LOOP    004660R         712*   767*   768    823*   824    880*   881   1141*  1427#
LPCNT   000166R         497#   768    824    881
MAP22$= 104415          490#
MASK    004576R         635*   639    641    645    647*  1402#
MD    = ******  GX        715
MDATAB  000272R         584#
MDMTAB  000312R         611#
MDROP   011416R        1504   1553#
MODNAM  000000R         454#
MODSP   000164R         465    488#
MSG   = ******  G       490#  1054
MSGBF0  005060R        1408   1500#
MSGBF1  007100R        1409   1501#
MSGCNT  004574R         658*  1363*  1364   1366*  1401#
MSGHED= ******  G       490#   757    803    849
MSGN$ = 104411          490#   790   1392
MSGSIZ= ******  G       490#   749    758    804    818    844    851   1073
MSGS$ = 104416          490#
MSGS  = 104406          490#
MSUM    004570R        1392   1398#
NULL    004152R        1274#
NXTLIN  004600R         695*   697    699    702*  1403#
OACNV = ******  G         1#   755#   786
ONEREC  001134R         501   813#
ONETMT  001456R         502
OPEN  = 000000          455    461    468    469    470    471    472    473    474    475    477    479    480
                        481    490#
PASCNT  000026R         466#
```



```
PASTIM  004746R         693*   729*  1331   1454#
PHYTAB  000212R         519#   636    643    656    695    697    722    725   1367   1390
PIRQ$ = 000004          490#   979    991   1223   1227   1279   1283
POPSP = 005726          490#
POPSP2= 022626          490#
PP      001036R         716    738#
PPFD    001570R         741    839#
PPFD1   001610R         843#   882
PPHDM   001372R         744    802#   825
PPHDM2  001520R         816    823#   834
PPHDS   001070R         747#   769
PPHDS2  001224R         756    767#   787
PRTY  = 000000          490#
PRTY0 = 000000          490#
PRTY1 = 000040          490#
PRTY2 = 000100          490#
PRTY3 = 000140          490#
PRTY4 = 000200          458    459    490#
PRTY5 = 000240          490#
PRTY6 = 000300          490#
PRTY7 = 000340          490#
PS    = 177776          490#
PSW   = 177776          490#
PUSH  = 005746          490#
PUSH2 = 024646          490#
PWRCNT= ******  G       490#   673
QUE$  = 104401          490#
RCTOR4  004764R         684*   685*  1125*  1129   1134*  1461#
RCVTO   004740R         713*  1039*  1114*  1115   1135*  1451#
RDTO    003230R         900    922   1113#
REC     002120R         753    821    846    909#
RECADR  004616R         748*   817*   843*   963*   964*  1410#
RECCNT  004622R         749*   750*   818*   819*   844*   845*   965*  1412#
RECD    004650R         972   1423#
RECDN   002654R         979   1035#  1423   1424
RECERR  003156R        1059   1096#
RECINT  002230R         911    930#
RECPTN  004664R         899*   909*   919   1078*  1080   1136   1429#
RECTIM  004626R         751*   752*   820*   923   1414#
RFPORT  000164R         494#   954   1056   1063   1083   1093   1361   1364
RERR    002452R         957    962    982#
RESTRT  000516R         482    660    673#
RETTAB  005040R        1040*  1079*  1217*  1312*  1326   1329   1489#
RPIRQ   002444R         972*   977#
RRET    003104R        1031   1079*  1085   1090   1109
RSTRT   000362R         482#
RTSSET  004144R        1271   1273#  1289
R4TAB   005050R        1311*  1494#
SRADR   000052R         476#
SPARE1  000202R         504#
SPARE2  000204R         505#
SPARE3  000206R         506#
SPARE4  000210R         507#
SPOINT  000024R         465#   677
SPSIZ = 000040            1#   483
SR1     000016R         461#   618
```

```
START   000332R          464     630#
STAT    000020R          463#
SUM     011130R         1398    1507#
SUM1    011214R         1368    1517#
SVR0    000032R          468#
SVR1    000034R          469#
SVR2    000036R          470#
SVR3    000040R          471#
SVR4    000042R          472#
SVR5    000044R          473#
SVR6    000046R          474#
SYNC    000174R          500#
SYNCNT  000632R          915*    945     950*   1416#
TIDONE  003706R         1195    1215#
TIEXIT  003634R         1190    1199#   1208    1212
TIMCHK  004300R          721    1317    1326#   1343
TIME  = ******  G        490#   1310    1346
TIMELN  000170R          498#    751     752     763     809     820     859     860
TIMKEP  004224R          862     926    1178    1308#
TIMTAB  005030R         1309*   1310*   1405#
TMT     003376R          764     810     855    1152#
TMTDON  004720R          864*    868     888*   1153*   1443#
TMTINT  003546R         1156    1181#
TMTRET  004714R         1152*   1168    1218*   1230    1232    1257    1441#
TMTTIM  004710R          763*    809*   1174    1439#
TPX   = 000000           490#
TRAPX = 000020           490#
TSYNCT  004724R          953*   1160*   1165*   1202    1206*   1445#
TXADR1  004670R          757*    803*    849*   1209    1210*   1431#
TXADR2  004674R          761*    807*    854*   1196    1197*   1433#
TXCNT1  004700R          758*    759*    804*    805*    851*    852*   1192    1211*   1435#
TXCNT2  004704R          760*    806*    853*   1194    1198*   1437#
TXTO    003772R         1173    1237#
TXTOR4  004770R          686*    687*   1247*   1251    1256*   1463#
VCTSAV  004606R          710*    910    1154    1266    1406#
VECTOR  000010R          457#    708
WAIT  = ******  G        490#    659
WASADR  000054R          478#
XFLAG   000005R          455#
X4      003114R         1072    1083#
X5      003124R         1057    1084    1087#   1094
.     = 011437R         1500#   1501#   1551#

. ABS.  000000     000
        011437     001

% ERRORS DETECTED:  1
% DEFAULT GLOBALS GENERATED:  1

 *NLAA,NLAA/SOL/CRF:SYM=DDXCOM,NLAA
 RUN-TIME: 7 12 1 SECONDS
 RUN-TIME RATIO: 87/21=4.1
 CORE USED:  8K  (15 PAGES)
```

```
                        .REM %
```

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DXN1A-A-D

PRODUCT NAME: DECNET DEC/X11 SERVICE EXERCISER MODULE #2

DATE: 21-JUN-76

MAINTAINER: DEC/X11 DIAGNOSTIC GROUP

38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

IDENTIFICATION

PRODUCT CODE:  MAINDEC-11-DXN2A-A-D

PRODUCT NAME:  DECNET DEC/X11 EXERCISER SERVICE MODULE

DATE:  21-JUN-76

MAINTAINER:  DEC/X11 DIAGNOSTIC GROUP



1.  ABSTRACT

N2A? MODULE IS A SERVICE MODULE WHICH MUST BE USED WHEN ANY DECNET DEC/X11 MODULE IS USED. IT MUST BE CONFIGURED AFTER THE LAST DECNET DEC/X11 MODULE AND IN FRONT OF THE CLOCK MODULE AND ALL OTHER NORMAL DEC/X11 MODULES. ITS PURPOSE IS TO PROVIDE A PAUSE POINT IN THE RUN TIME EXERCISER TO ALLOW ALL NODES TO GET ALL PHONE CONNECTIONS MADE BEFORE EXERCISING BEGINS. IT ALSO PROVIDES A DN11 HANDLER FOR AUTO CALL UNITS. (NOT YET SUPPORTED)

2.  HARDWARE REQUIREMENTS

NONE

3.  SOFTWARE REQIREMENTS

N2A? SHOULD ONLY BE USED IN A DEC/X11 EXERCISER THAT ALSO CONTAINS THE N1A? MODULE, A CLOCK MODULE, AND ONE OR MORE DECNET DEC/X11 MODULES.

4.  OPERATING PROCEDURES

```
                        IF SR1 HAS BIT 0 SET TO 0, THE MODULE WILL TYPE OUT A MESSAGE
REQUESTING THAT ALL PHONE CONNECTIONS SHOULD BE MADE.  WHEN THIS  MESSAGE
IS TYPED OUT ON ALL NODES, THE OPERATORS MAY BEGIN TO DIAL ALL
MANUAL CONNECTIONS.  WHEN DONE, THE OPERATOR AT EACH NODE SHOULD
TYPE A CARRIAGE RETURN, ALLOWING THE EXERCISING TO BEGIN.  AT THIS
POINT THE N2A? MODULE WILL DESELECT ITSELF.
        IF SR1 HAS BIT 0 SET TO 1, THE MODULE WILL ATTEMPT TO DIAL
PHONE CONNECTIONS USING DN11'S.  DVA,VCT, AND BR1 SHOULD
BE SET TO THE LOWEST ADDRESSED DN11.  THE MODULE WILL TYPE A P#
PROMT REQUESTING THE FIRST PHONE NUMBER. IT WILL ACCEPT ONLY DIGITS,
DO NOT TYPE -'S AS PART OF THE NUMBER.  TYPE A CARRIAGE RETURN AT THE
END OF THE PHONE NUMBER AND THEN THE MODULE WILL ATTEMPT THE PHONE CALL.
THE MODULE WILL TYPE ANOTHER P# WHEN IT HAS SUCCESSFULLY DIALED
THE NUMBER, ALLOWING FOR ANY NUMBER OF DN11'S TO BE USED.
A CARRIAGE RETURN WITH NO NUMBER INDICATES YOU ARE DONE USING DN11'S
AND THE MODULE SHOULD CONTINUE ON.  IT WILL THEN GO ON TO THE MANUAL
CONNECTION PHASE AS DESCRIBED ABOVE.
        THERE ARE 4 ERROR NUMBERS TYPED OUT  FOR DN11 PROBLEMS;
        THEY ARE:
                0 = PND (PRESENT NEXT DIGIT) BIT NOT SET
                1 = POWER IN ACU FAILED OR ACR (ABANDONED CALL AND
                    RETRY) BIT SET
                2 = SOFTWARE TIMED OUT  AFTER WAITING 3 MINUTES FOR 1 CALL
                3 = POWER NOT ON IN ACU (AUTO-CALL UNIT)

**********************************************************************
                IMPORTANT
EVERY NODE TO BE EXERCISED MUST REACH THE MESSAGE P# IF SR1 HAS BIT 0 SET,
 OR "DIAL ALL MANUAL CONNECTIONS" IF SR1 BIT 0 IS A
ZERO BEFORE ANY OTHER ACTION, INCLUDING ENTERING PHONE NUMBERS
OR MANUALLY DIALING CONNECTIONS, IS ATTEMPTED.  OTHERWISE
LINES THAT APPEAR TO BE SET UP MAY HANG UP BY THEMSELVES BECAUSE
THE OTHER SIDE OF THE LINE HAS NOT CONDITIONED THE MODEM TO ACCEPT A CALL.
**********************************************************************

AFTER ALL MANUAL CONNECTIONS ARE COMPLETE AND THE OPERATOR TYPES
A CARRIAGE RETURN, THE N2A? MODULE WILL DESELECT ITSELF BECAUSE IT
IS NO LONGER NEEDED.

5.      OPERATOR OPTIONS

        THE ONLY OPTION IS TO SET SR1 BIT 0 TO A ONE TO ALLOW THE
MODULE TO DIAL AUTO DIAL UNITS.  (THIS OPTION IS NOT YET SUPPORTED)
%
```

```
                                        IOMOD   <N2AA >, 175200,350,4,0,0
                                        MODULE  140000,N2AA ,175200,350,4,0,0
                                        .TITLE  N2AA DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  4    9/6/75
                                        .LIST   BIN
                                ;*************************************************************************
000000'                         BEGIN:
000000' 031116  040501      040 MODNAM: .ASCII  /N2AA / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 175200                  ADDR:   175200+0                ;1ST DEVICE ADDR.
000010' 000350                  VECTOR: 350+0                   ;1ST DEVICE VECTOR.
000012'     200                 BR1:    .BYTE   PRTY4+0         ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
                                ;*************************************************************************
000020' 140000                  STAT:   140000                  ;STATUS WORD.
000022' 000164'                 INIT:   START                   ;MODULE START ADDR.
000024' 000164'                 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.
000062' 000164'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
                                        .NLIST
                                        .WORD   0
                                        .LIST
                                        .ENDR
000164'                         MODSP:
                                ;*************************************************************************
                                        .GLOBL  TIME, TKS, TKB, TYPE, WAIT
000164'                         START:
000164' 000240                  RESTRT: NOP
000166' 012700  000020                  MOV     #20,R0  ;SET UP TIME LOOP TO GIVE OTHER SYSTEMS
                                                        ;TIME TO SET DATA TERM. READY ON THEIR LINES
000172' 005001                  1$:     CLR     R1
000174' 005301                  2$:     DEC     R1              ;COUNT A TIC
000176' 001376                          BNE     2$              ;
000200' 005300                          DEC     R0              ;COUNTS TRIPS THROUGH LITTLE LOOP
000202' 001373                          BNE     1$              ;GO COUNT SOME MORE
000204' 017767  000000G 000500          MOV     @TKS,TKSSAV     ;SAVE TTY STATUS
000212' 005077  000000G                 CLR     @TKS            ;DISABLE INTERRUPTS
000216' 005067  000432                  CLR     ERRNUM          ;CLEAR ERR #
```



```
000222' 032767  000001  177566        BIT     #1,SR1          ;ANY DN11'S?
000230' 001002                        BNE     3$              ;BR IF
000232' 000167  000346                JMP     DONE            ;NO SKIP
000236' 216704  177544          3$:   MOV     ADDR,R4         ;GET DEVICE ADDRESS
000242' 005014                  GETNUM: CLR     (R4)            ;RESET DN11
000244' 012700  000656'               MOV     #NUMBER, R0     ;R0 POINTS TO PHONE #
000250' 005001                        CLR     R1              ;COUNTS DIGIT
000252' 005067  000376                CLR     ERRNUM          ;SHOWS WHICH TYPE OF DN ERROR
000256' 004567  000000G               JSR     R5,TYPE         ;TYPE A "P# "
000262' 000714'                       PNM
000264' 105777  000000G         3$:   TSTB    @TKS            ;WAIT FOR A CHARACTER
000270' 100375                        BPL     3$
000272' 117702  000000G               MOVB    @TKB,R2         ;BRING CHAR IN
000276' 142702  000200                BICB    #200,R2         ;GET RID OF PARITY
000302' 120227  000015                CMPB    R2,#15          ;IS IT A CR?
000306' 001430                        BEQ     6$              ;BR IF DONE INPUTTING
000310' 162702  000060                SUB     #60,R2          ;MAKE IT A DIGIT
000314' 103363                        BCT     3$              ;GO BACK AND IGNORE IT
000316' 120227  000117                CMPB    R2,#117         ;IS IT A RUBOUT?
000322' 001406                        BEQ     4$              ;BR IF YES
000324' 120227  000012                CMPB    R2,#12          ;IS IT A LEGAL DIGIT
000330' 103355                        BGE     3$              ;BR BACK AND IGNORE IT
000332' 110220                        MOVB    R2,(R0)+        ;SAVE DIGIT
000334' 005201                        INC     R1              ;COUNT A DIGIT
000336' 000404                        BR      5$
000340'                         4$:
000340' 005701                        TST     R1              ;ANY DIGITS TO RUBOUT?
000342' 001750                        BEQ     3$              ;BR IF NOT
000344' 114002                        MOVB    -(R0),R2        ;GET OLD CHAR
000346' 005301                        DEC     R1              ;1 LESS DIGIT
000350' 062702  000060          5$:   ADD     #60,R2          ;MAKE IT ASCII
000354' 110267  000342                MOVB    R2,TBUF         ;TYPE IT
000360' 004567  000000G               JSR     R5,TYPE
000364' 000722'                       TBUF
000366' 000736                        BR      3$              ;GET NEXT DIGIT
000370' 004567  000000G         6$:   JSR     R5,TYPE         ;TYPE A CR,LF
000374' 000724'                       CRLF
000376' 005701                        TST     R1              ;ANY DIGITS?
000400' 001002                        BNE     7$              ;BR IF YES
000402' 000167  000176                JMP     DONE            ;YES,DONE DIALING
000406' 000240                  7$:   NOP
000410' 012703  000656'               MOV     #NUMBER,R3      ;GET ADDR OF NUMBER
000414' 012714  000001                MOV     #1, (R4)        ;SET CALL REQ
000420' 005714                        TST     (R4)            ;IS PWR ON IN ACU
000422' 100445                        BMI     ERR             ;BR IF NOT
000424' 000415                        BR      DIAL            ;GO START CALLING
000426' 032714  000200          DNL1: BIT     #200,(R4)       ;IS DOING FLAG SET
000432' 001001                        BNE     1$              ;BR IF YES
000434' 000774                        BR      DNL1            ;BR BACK
000436' 042714  000200          1$:   BIC     #200, (R4)      ;CLEAR DONE FLAG
000442' 032714  140000                BIT     #140000,(R4)    ;ANY ERRORS?
000446' 001035                        BNE     ERR+4           ;BR IF YES
000450' 032714  000020                BIT     #20, (R4)       ;IS PRESENT NEXT DIGIT SET?
000454' 001001                        BNE     DIAL            ;BR IF YES
000456' 000432                        BR      ERR+6           ;ERR, PND NOT SET
000460' 112364  000001          DIAL: MOVB    (R3)+,1(R4)     ;LOAD NEXT DIGIT
```

```
  251  000464' 052714  000002                BIS     #2, (R4)        ;SET DIGIT PRESENT
  252  000470' 005301                        DEC     R1              ;COUNT A DIGIT
  253  000472' 001355                        BNE     DNL1            ;BR IF MORE DIGITS
  254  000474' 012703  002000        1$:     MOV     #2000,R3        ;SET UP TIMING LOOP
  255  000500' 032714  000200        2$:     BIT     #200,(R4)       ;WAIT FOR DONE
  256  000504' 001006                        BNE     4$              ;BR IF YES
  257  000506' 005005                        CLR     R5
  258  000510' 005305                3$:     DEC     R5              ;JUST KILLING TIME
  259  000512' 001376                        BNE     3$
  260  000514' 005303                        DEC     R3              ;COUNT LITTLE LOOPS
  261  000516' 001370                        BNE     2$              ;BR BACK IF NOT TIME OUT
  262  000520' 000407                        BR      ERR+2           ;BR IF SO TO ERR
  263  000522' 032714  040000        4$:     BIT     #40000, (R4)    ;WAS CALL ABANDONED
  264  000526' 001005                        BNE     ERR+4           ;BR IF YES
  265  000530' 005724                        TST     (R4)+           ;BUMP TO NEXT DN
  266  000532' 000167  177504                JMP     GETNUM          ;GET NEXT PHONE NUMBER
  267  000536' 005267  000112        ERR:    INC     ERRNUM          ;SHOW WHICH TYPE ERR
  268  000542' 005267  000106                INC     ERRNUM          ;
  269  000546' 005267  000102                INC     ERRNUM          ;
  270  000552' 062767  000060  000074        ADD     #60,ERRNUM      ;
  271  000560' 116767  000070  000154        MOVB    ERRNUM,EX       ;TYPE ERR MSG
  272  000566' 005067  000062                CLR     ERRNUM          ;CLEAR FOR NEXT TIME
  273  000572' 004567  000000G               JSR     R5,TYPE
  274  000576' 000730'                        EPRMSG
  275  000600' 000167  177436                JMP     GETNUM          ;TRY AGAIN
  276  000604' 004567  000000G       DONE:   JSR     R5,TYPE         ;TELL HIM TO MAKE MANUAL CALLS
  277  000610' 000761'                        MCALL
  278  000612' 105777  000000G       1$:     TSTB    @TKS            ;WAIT FOR CR
  279  000616' 100375                        BPL     1$
  280  000620' 117702  000000G               MOVB    @TKB,R2         ;BRING CHAR IN
  281  000624' 142702  000200                BICB    #200,R2         ;GET RID OF PARITY
  282  000630' 122702  000015                CMPB    #15,R2          ;IS IT A CR?
  283  000634' 001366                        BNE     1$              ;NO, IGNORE IT
  284  000636' 005067  000000G               CLR     WAIT            ;CLEAR GLOBAL FLAG
  285
  286  000642' 016777  000044  000000G       MOV     TKSSAV,@TKS     ;RESTORE TTY
  287  000650' 104403  000000'               END$,BEGIN              ;
  288  000654' 000000                ERRNUM: 0                       ;HOLDS ERR NUMBER
  289  000656' 000016                NUMBER: .BLKW   16              ;PHONE # GOES HERE
  290  000712' 000000                TKSSAV: 0
  291  000714' 021520  037440  000040 PNM:    .ASCIZ  'P# ? '         ;PHONE # PROMPT MESSAGE
  292  000722' 000000                TBUF:   0                       ;1 CHAR TYPE BUFFER
  293  000724'     015               CRLF:   .BYTE   15              ;CR AND LF MESSAGE
  294  000725'     012                       .BYTE   12
  295  000726' 000000                        .WORD   0
  296  000730' 047104  042440  051122 ERRMSG: .ASCII  'DN ERROR #'
  297  000736' 051117  021440
  298  000742'     040               EX:     .BYTE   40
  299  000743'     054  051124  020131        .ASCIZ  ',TRY AGAIN.'<15><12>
  300  000750' 043501  044501  027116
  301  000756' 005015      000
  302
  303  000761'     015  042012  040511 MCALL:  .ASCII  <15><12>'DIAL ALL MANUAL PHONE CONNECTIONS,'<15><12>
  304  000766' 020114  046101  020114
  305  000774' 040515  052516  046101
  306  001002' 050040  047510  042516
```



```
  307  001010' 041440  047117  042516
  308  001016' 052103  047511  051516
  309  001024' 006454      012
  310  001027'     124  050131  020105        .ASCIZ  'TYPE CR WHEN DONE.'<15><12>
  311  001034' 051103  053440  042510
  312  001042' 020116  047504  042516
  313  001050' 006456  000012
  314          000001                        .END
```

```
ACSR    000052R          169#
ADDR    000006R          148#    198
ASB     000056R          172#
ASTAT   000054R          171#
AWAS    000060R          173#
BDCNV = ******  G          1#
BEGIN   000000R          145#    287
BIT0  = 000001           182#
BIT1  = 000002           182#
BIT10 = 002000           182#
BIT11 = 004000           182#
BIT12 = 010000           182#
BIT13 = 020000           182#
BIT14 = 040000           182#
BIT15 = 100000           182#
BIT2  = 000004           182#
BIT3  = 000010           182#
BIT4  = 000020           182#
BIT5  = 000040           182#
BIT6  = 000100           182#
BIT7  = 000200           182#
BIT8  = 000400           182#
BIT9  = 001000           182#
BREAK$= 104407           182#
BR1     000012R          150#
BR2     000013R          151#
CDATA$= 104414           182#
CRLF    000724R          231     293#
CSRA    000050R          167#
DATCK$= 104417           182#
DATER$= 104405           182#
DIAL    000460R          240     248     250#
DNL1    000426R          241#    243     253
DONE    000604R          197     234     276#
DVID1   000014R          152#
ENDPS$= 104402           182#
END$  = 104403           182#    287
ERR     000536R          239     246     249     262     264     267#
ERRCNT  000030R          159#
ERRMSG  000730R          274     296#
ERRNUM  000654R          194*    202*    267*    268*    269*    270*    271     272*    288#
ERRN$ = 104410           182#
ERROR$= 104404           182#
EX      000742R          271*    298#
EXIT$ = 104400           182#
GETNUM  000242R          199#    266     275
GETPA$= 104413           182#
GWBUF$= 104412           182#
INIT    000022R          156#
MAP22$= 104415           182#
MCALL   000761R          277     303#
MODNAM  000000R          146#
MODSP   000164R          157     180#
MSGN$ = 104411           182#
MSGS$ = 104416           182#
MSG$  = 104406           182#
```



```
NUMBER  000656R          200     236     289#
OACNV = ******  G          1#
OPEN  = 000000           147     153     160     161     162     163     164     165     166     167     169     171     172
                         173     182#
PASCNT  000026R          158#
PIRQ$ = 000004           182#
PNM     000714R          204     291#
POPSP = 005726           182#
POPSP2= 022626           182#
PRTY  = 000700           182#
PRTY0 = 000000           151     182#
PRTY1 = 000040           182#
PRTY2 = 000100           182#
PRTY3 = 000140           182#
PRTY4 = 000200           150     182#
PRTY5 = 000240           182#
PRTY6 = 000300           182#
PRTY7 = 000340           182#
PS    = 177776           182#
PSW   = 177776           182#
PUSH  = 005746           182#
PUSH2 = 024646           182#
QUES  = 104401           182#
RESTRT  000164R          174     184#
RSTRT   000062R          174#
SBADR   000052R          168#
SPOINT  000024R          157#
SPSIZ = 000040             1#    175
SR1     000016R          153#    195
START   000164R          156     183#
STAT    000020R          155#
SVR0    000032R          160#
SVR1    000034R          161#
SVR2    000036R          162#
SVR3    000040R          163#
SVR4    000042R          164#
SVR5    000044R          165#
SVR6    000046R          166#
TBUF    000722R          226*    228     292#
TIME  = ******  G          182#
TKB   = ******  G          182#    207     280
TKS   = ******  G          182#    192     193*    205     278     286*
TKSSAV  000712R          192*    206     290#
TPX   = 000000           182#
TRAPX = 000020           182#
TYPE  = ******  G          182#    203     227     230     273     276
VECTOR  000010R          149#
WAIT  = ******  G          182#    284*
WASADR  000054R          170#
XFLAG   000005R          147#
.     = 001054R          289#

. ABS.  000000     000
        001054     001
```

```
ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

*N2AA,N2AA/SOL/CRF:SYM=DDXCOM,N2AA
RUN-TIME:  2  3  .4 SECONDS
RUN-TIME RATIO: 48/7=5.4
CORE USED:  8K  (15 PAGES)
```