```
        REM $LINESIZE:132
        WIDTH "LPT1:",132
        DIM LLIMIT%(20)
        DATA 1, 1, 1, 0, 1, 1, 3, 1, 3, 1, 3, 10, 1, 0, 35, 15, 35, 35, 19, 4
        DIM IOBCMD$(4)
        DATA "read", "write", "execute", "init"
        DIM DIMCMD$(4)
        DATA "nop", "mount", "unmount", "info"
        DIM DIMACC$(3)
        DATA "ro", "rw", "xo"
        DIM DIMSHR$(4)
        DATA "shr", "TILT", "upd", "exc"
        REM PEEK A 68K STYLE INTEGER
        F256 = 256.0
        DEF FNRWORD(ADDRESS)=PEEK(ADDRESS)*F256+PEEK(ADDRESS+1)
        DEF FNWPEEK(ADDRESS)=PEEK(ADDRESS+1)*F256+PEEK(ADDRESS)
        DEF FNMIN(A%,B%)=-( ((A%<B%)*A%) + ((A%>=B%)*B%) )
        DEF FNNUM$(N) = RIGHT$(STR$(N), LEN(STR$(N))-1) + " "
        FRAMES%=0
        KEY OFF
        CLS
        VER$="v4.0"
        PRINT "                    SNIFFER "; VER$
        PRINT:PRINT"     Written by John H A Rowland  B.A. (Cantab)"
        PRINT:PRINT"                Zynar Ltd 1982."
        PRINT:PRINT"   Modified by Craig W Payne, Nothing (Lehigh U.)"
        PRINT:PRINT"                Compiled BASIC version."
        CALL INITIALIZE(BYTE%)
        PRINT:PRINT"           The sniffer's address is $";
        GOSUB 60700
        HEADER%=0
        FOR INDEX%=1 TO 20 : READ LLIMIT%(INDEX%) : NEXT INDEX%
        FOR INDEX%=1 TO 4 : READ IOBCMD$(INDEX%) : NEXT INDEX%
        FOR INDEX%=0 TO 3 : READ DIMCMD$(INDEX%) : NEXT INDEX%
        FOR INDEX%=1 TO 3 : READ DIMACC$(INDEX%) : NEXT INDEX%
        FOR INDEX%=1 TO 4 : READ DIMSHR$(INDEX%) : NEXT INDEX%
2700    CALL INITIALIZE(BYTE%)
2800    LOCATE 16,1
        PRINT" Please enter number of network frames to be recorded in RAM ";
        PRINT " (N.B. 0 fills all RAM available)
        INPUT FRAMES%
        PACKET%=FRAMES%
3300    LOCATE 20,1 : PRINT " Enter stations you are interested in (in hex), or RETURN to continue "
        PRINT "STATION 1 :";
        INPUT STN1$
        PRINT "STATION 2 :";
        INPUT STN2$
4000    IF VAL("&H"+STN1$) > 255 OR VAL("&H"+STN1$) < 0 OR VAL("&H"+STN2$) > 255 OR VAL("&H"+STN2$) < 0 THEN 3300
        STATIONS% = VAL("&H"+STN2$+"00")+VAL("&h"+STN1$)
        IF STN1$ = "" THEN STN1$ = "XX"
        IF STN2$ = "" THEN STN2$ = "XX"
        PREFLT = (STATIONS% <> 0)
        CLS
        PRINT "            packets seen"
        PRINT "            packets accepted"
        PRINT "            reconfigurations"
        PRINT
        PRINT "Accepting packets between ";STN1$;" and ";STN2$
```

```
4500    IF FRAMES%=0 THEN PRINT "Will receive frames until interrupted":GOTO 4800
        PRINT "Now waiting for "FRAMES%" frames to be received"
4800    PRINT "Press any key to interrupt receive operation."
        CALL MAINPROG(FRAMES%,STATIONS%)
        CLS
        OLDTIME=-1:OLDPACK%=0
        LOCATE 10,1
        PRINT "    Last frame received is ",FRAMES%
        IF FRAMES%=0 THEN 2800
        PRINT CHR$(7)
        PACKNO%=1
6000    LOCATE 25,2
6100    PRINT VER$; ": All, Cmds, Data, Frame, Head, Iobs, Next, Prev, Last, Restart, Quit ?";
6200    CMD$=INKEY$
        IF LEN(CMD$)=0 THEN 6200
        IF CMD$ > "Z" THEN CMD$ = CHR$(ASC(CMD$)-32)
        PRINT CMD$;
        IF CMD$="A" THEN CLS : GOTO 32000
        IF CMD$="C" THEN CLS : HEADER% = 0 : GOTO 32000
        IF CMD$="D" THEN 26500
        IF CMD$="F" THEN CLS : GOTO 7400
        IF CMD$="H" THEN HEADER%=-1 : GOTO 7600
        IF CMD$="I" THEN CLS : HEADER% = 0 : GOTO 32000
        IF CMD$="L" THEN PACKNO%=FRAMES% : CLS :LOCATE 15,1 : GOTO 7600
        IF CMD$="N" THEN PACKNO%=PACKNO%+1 : CLS :LOCATE 15,1 : GOTO 7600
        IF CMD$="P" THEN PACKNO%=PACKNO%-1 : CLS :LOCATE 15,1 : GOTO 7600
        IF CMD$="Q" THEN CLS : GOTO 10000
        IF CMD$="R" THEN CLS : GOTO 2700
        PRINT CHR$(7);:GOTO 6200
7400    LOCATE 20,5
        PRINT "ENTER NUMBER OF RECEIVED FRAME TO BE DISPLAYED ";: GOSUB 60300
7600    IF PACKNO%>FRAMES% THEN PRINT : PRINT "      ONLY "FRAMES%" FRAMES IN MEMORY.": GOTO 7400
        IF PACKNO%=0 THEN PRINT : PRINT "      ZERO IS AN INVALID NUMBER.": GOTO 7400
        IF PACKNO%<0 THEN PRINT : PRINT "      NO SUCH BEAST AS NEGATIVE FRAME NUMBER.": GOTO  7400
        CLS
        GOSUB 61600
        IF NOT HEADER% THEN 26500
        LOCATE 1,1
        PRINT "    FRAME NUMBER"PACKNO%
        LOCATE 1,25
        PRINT "TIMESTAMP =";
        TIME=FNWPEEK(256)
        PRINT TIME/1000;" sec."
        IF OLDPACK% <=0 THEN 9600
        LOCATE 1,50
        PRINT (OLDTIME-TIME)/1000;" sec. from packet ";OLDPACK%
9600    OLDTIME=TIME:OLDPACK%=PACKNO%
        LOCATE 2,1
        PRINT "    ============="
        LOCATE 3,1
        PRINT "------------------------------** LEVEL 2 HEADERS **------------------------------"
        PRINT "      SOURCE ID. = $";
        BYTE%=PEEK(0):GOSUB 60700
        PRINT "      DESTINATION ID. = $";
        BYTE%=PEEK(1):GOSUB 60700
        IF LP%=0 THEN PRINT "      FRAME TYPE = LONG" ELSE PRINT "      FRAME TYPE = SHORT"
        PRINT "           COUNT = $";
        BYTE%=COUNT%:GOSUB 60700
```

```
        PRINT "          SYSTEM CODE = $";
        OFFSET%=4:GOSUB 60600
        PRINT "       GARBAGE COUNT = $";
        OFFSET%=5:GOSUB 60600
        LOCATE 8,1
        PRINT "-----------------------------** LEVEL 3 HEADERS **-----------------------------"
        PRINT "      PACKET ID. = $";
        OFFSET%=6:GOSUB 60600
        PRINT "            FRAME NUMBER = $";
        OFFSET%=7:GOSUB 60600
        PRINT "          CHECKSUM = $";
        OFFSET%=8:GOSUB 60800
        PRINT "          LENGTH = ";
        PRINT USING "###";FNRWORD(10);
        PRINT "      TRANSPORT CONTROL =";PEEK(12);
        PRINT "       PACKET TYPE =";PEEK(13)
        PRINT
        PRINT "                              >> SOURCE <<                    >> DESTINATION <<
        PRINT "      NETWORK:-"
        PRINT "         HOST:-"
        PRINT "       SOCKET:-"
        LOCATE 14,60
        PRINT "$";
        FOR OFFSET%=14 TO 17:GOSUB 60600:NEXT
        LOCATE 15,60
        PRINT "$";
        FOR OFFSET%=18 TO 23:GOSUB 60600:NEXT
        LOCATE 16,60
        PRINT FNRWORD(24);
        LOCATE 14,25
        PRINT "$";
        FOR OFFSET%=26 TO 29:GOSUB 60600:NEXT
        LOCATE 15,25
        PRINT "$";
        FOR OFFSET%=30 TO 35:GOSUB 60600:NEXT
        LOCATE 16,25
        PRINT FNRWORD(36);
        IF PEEK(7)>0 ANI PEEK(6)>128 THEN LOCATE 20,15 : GOTO 6000
        LOCATE 18,1
        PRINT "-----------------------------** LEVEL 4 HEADERS **-----------------------------"
        PRINT "    CONNECTION CONTROL :- ";
        CONROL%=PEEK(38)
        IF (CONROL% AND &H80) = &H80 THEN PRINT "SYS ";
        IF (CONROL% AND &H40) = &H40 THEN PRINT "ACK ";
        IF (CONROL% AND &H20) = &H20 THEN PRINT "ATT ";
        IF (CONROL% AND &H10) = &H10 THEN PRINT "EOM ";
        LOCATE 20,54
        PRINT "DATA STREAM TYPE =";PEEK(39);
        PRINT "    SOURCE CONNECTION ID. = $";
        OFFSET%=40:GOSUB 60800
        PRINT "              DESTINATION CONNECTION ID. = $";
        OFFSET%=42:GOSUB 60800
        PRINT
        PRINT "    SEQUENCE No. =";FNRWORI(44);
        PRINT "     ACKNOWLEDGE No. =";FNRWORD(46);
        PRINT "     ALLOCATION No. =";FNRWORD(48);
        GOTO 6000
```

```
10000   PRINT
        INPUT "Do you really wish to quit ? ", ANS$
        IF (LEFT$(ANS$, 1) = "Y") OR (LEFT$(ANS$, 1) = "y") THEN SYSTEM
        GOTO 6000

26500   CLS
        GOSUB 61600
        HEADER%=0
        LOCATE 3,20
        PRINT "DATA ASSOCIATED WITH RECEIVED FRAME NUMBER"PACKNO%
        LOCATE 4,20
        PRINT "============================================="
        DEND%=FNRWORD(10)-43
        IF LP%=0 THEN TOP%=511 ELSE TOP%=255
        IF COUNT%+45 > TOP% THEN LOCATE 15,1 : PRINT "  Count incorrect not leaving sufficient room for complete header in buffer."
: GOTO 6000
        PRINT
        IF DEND%<0 THEN PRINT "  No data in packet" : goto 6000
        FOR ROW%=0 TO DEND% \ 16
        BYTE%=ROW% * 16 : GOSUB 60700:PRINT ": ";
        FOR INDEX%=ROW%*16 TO FNMIN(ROW%*16+15, DEND%)
        BYTE% = PEEK(INDEX%+50)
        GOSUB 60700
        PRINT " ";
        NEXT INDEX%
        LOCATE (ROW% + 6),60
        PRINT "*";
        FOR INDEX%=ROW%*16 TO FNMIN(ROW%*16+15, DEND%)
        BYTE% = PEEK(INDEX%+50)
        IF BYTE%<&H20 OR BYTE%>&H7E THEN PRINT " "; ELSE PRINT CHR$(BYTE%);
        NEXT INDEX%
        PRINT "*"
        NEXT ROW%
        GOTO 6000

32000   PRINT "starting frame ";:GOSUB 60300
        START%=PACKNO%
        PRINT "ending frame ";:GOSUB 60300
        ENDP%=PACKNO%
        IF (START%<1) OR (FRAMES%<START%) THEN 32000
        IF (ENDP%<1) OR (FRAMES%<ENDP%) THEN 32000
        PRINT "  Enter stations you are interested in (in hex), or RETURN to continue "
        PRINT "STATION 1 :";
        INPUT STN1$
        STN1 = VAL("&H" + STN1$)
        PRINT "STATION 2 :";
        INPUT STN2$
        STN2 = VAL("&H" + STN2$)
        POSTFLT = (STN1 <> 0) OR (STN2 <> 0)
33400   INPUT "enter list device (RETURN for screen, 'prn' for printer) ";D$
        IF D$="" THEN D$="SCRN:" : GOTO 33410
        IF RIGHT$(I$, 1) = ":" THEN PRINT "Bad file name" : GOTO 33400

33410   OPEN D$ FOR OUTPUT AS #1
        OLDTIME=0
        PRINT "press the space bar to abort listing"
        FOR PACKNO%=START% TO ENDP%
        IF INKEY$=" " THEN 40000
```

```
        GOSUB 61600
        IF (STN1 <> 0) AND (STN1 <> PEEK(0)) AND (STN1 <> PEEK(1)) THEN 39900
        IF (STN2 <> 0) AND (STN2 <> PEEK(0)) AND (STN2 <> PEEK(1)) THEN 39900

        IF HEADER% THEN GOSUB 61300 : GOTO 39900

        IF (PEEK(38) AND 128) = 128 THEN 39900
        IF CMD$ = "A" THEN 33490
        IF CMD$ = "C" THEN 33420
        IF CMD$ = "I" THEN 33430
        PRINT "WHAT IS COMMAND "; CMD$; " DOING AT 32000 ?" : STOP

33420   IF (TYPE% = 3) OR (TYPE% = 16) THEN 33490
        IF (TYPE% = 5) AND (PEEK(50) = 1) AND (PREFLT OR POSTFLT) THEN PRINT #1, " "
        IF ((TYPE% = 5) OR (TYPE% = 6)) AND (PEEK(50) <> 2) AND (PEEK(50) <> 3) THEN 33490
        GOTO 39900

33430   IF (TYPE% = 15) OR (TYPE% = 17) THEN 33490
        IF (TYPE% = 5) AND (PEEK(50) = 2) OR (PEEK(50) = 3) AND (PREFLT OR POSTFLT) THEN PRINT #1, " "
        IF ((TYPE% = 5) OR (TYPE% = 6)) AND ((PEEK(50) = 2) OR (PEEK(50) = 3)) THEN 33490
        GOTO 39900

33490   IF (FNRWORD(42)=0) AND (PREFLT OR POSTFLT) THEN PRINT #1, " "
        GOSUB 61500
        IF (TYPE% < 1) OR (20 < TYPE%) THEN GOSUB 62100 : GOTO 39900
        IF LENGTH% >= LLIMIT%(TYPE%) THEN 35210
        PRINT #1, "<TILT - packet too small for type"; TYPE%; ">"
        GOTO 39900

35210   REM                1      2      3      4      5      6      7      8      9      10
        ON TYPE% GOSUB 61700, 61800, 62000, 62100, 62500, 62500, 63300, 63800, 63400, 63800

        IF TYPE% < 11 THEN 39900

        REM                   11     12     13     14     15     16     17     18     19     20
        ON TYPE% - 10 GOSUB 63500, 63900, 62100, 64000, 64100, 64200, 64300, 64400, 64500, 64600

39900   NEXT PACKNO%
40000   IF (D$="PRN") OR (D$="prn") THEN PRINT #1, CHR$(27);"v.c"
        CLOSE #1
        PACKNO%=START%
        GOTO 6100

60000   IF PEEK(OFFSET%)<16 THEN PRINT #1,"0";
        PRINT #1,HEX$(PEEK(OFFSET%));" ";
        RETURN

60100   TEMP=FNRWORD(OFFSET%)
60110   IF TEMP <4096 THEN PRINT #1,"0";
        IF TEMP <256 THEN PRINT #1,"0";
        IF TEMP <16 THEN PRINT #1,"0";
        PRINT #1,HEX$(TEMP);" ";
        RETURN

60200   IF (PEEK(38) AND MASK%) = MASK% THEN PRINT #1,"1 "; ELSE PRINT #1,"0 ";
        RETURN

60300   INPUT IN$
```

```
        PACKNO%=VAL(IN$)
        IF IN$="l" OR IN$="L" THEN PACKNO%=FRAMES%
        IF IN$="f" OR IN$="F" THEN PACKNO%=1
        RETURN

60600   BYTE%=PEEK(OFFSET%)
60700   IF BYTE%<16 THEN PRINT "0";
        PRINT HEX$(BYTE%);
        RETURN

60800   WORD=FNRWORD(OFFSET%)
        IF WORD<16 THEN PRINT "000";:GOTO 61000
        IF WORD<256 THEN PRINT "00";:GOTO 61000
        IF WORD<4096 THEN PRINT "0";
61000   PRINT HEX$(WORD);
        RETURN

61100   LZERO = -1
        PRINT #1, "$";
        FOR INDEX%=OFFSET% TO OFFSET% + 3
        BYTE% = PEEK(INDEX%)
        LZERO = LZERO AND (BYTE% = 0)
        IF LZERO THEN 61110
        IF BYTE% < 16 THEN PRINT #1, "0";
        PRINT #1, HEX$(BYTE%);
61110   NEXT INDEX%
        IF LZERO THEN PRINT #1, "0";
        RETURN

61300   IF PREFLT OR POSTFLT THEN 61310
        IF (PACKNO% - START%) MOD 10 = 0 THEN 61320 ELSE 61400

61310   IF FNRWORD(42) <> 0 THEN 61400
61320   PRINT #1," "
        PRINT #1,"                    s    d      t            s a a e t"
        PRINT #1,"                    r    s      y            y c t o y"
        PRINT #1,"  #   time   delta c    t  len p  ssck dsck s k n m p  sid  did  seq  ack"
61400   GOSUB 61500
        PRINT #1,USING "### ";FNRWORD(10);
        OFFSET%=13:GOSUB 60000
        OFFSET%=36:GOSUB 60100
        OFFSET%=24:GOSUB 60100
        MASK%=128:GOSUB 60200
        MASK%=64:GOSUB 60200
        MASK%=32:GOSUB 60200
        MASK%=16:GOSUB 60200
        OFFSET%=39:GOSUB 60000
        OFFSET%=40:GOSUB 60100
        OFFSET%=42:GOSUB 60100
        OFFSET%=44:GOSUB 60100
        OFFSET%=46:GOSUB 60100
        PRINT #1," "
        RETURN

61500   PRINT #1,USING "#### ";PACKNO%;
        TIME=FNWPEEK(256)
        PRINT #1,USING "##.### ";TIME/1000;
        DELTA=OLDTIME-TIME
```

```
        IF DELTA <0 THEN DELTA = DELTA+65536!
        PRINT #1,USING "##.### ";DELTA/1000;
        OLDTIME=TIME
        OFFSET%=0:GOSUB 60000
        IF PEEK(258) = 0 THEN PRINT #1, "> "; : ELSE PRINT #1, ">>";
        OFFSET%=1:GOSUB 60000
        RETURN

61600   CALL PACKLOC(PACKNO%, PACKBASE%)
        DEF SEG=PACKBASE%
        LP%=PEEK(2)
        IF LP%<>0 THEN COUNT%=LP% ELSE COUNT%=PEEK(3)
        TYPE%=PEEK(39)
        LENGTH%=FNRWORD(10)
        RETURN

61700   PRINT #1, "<connect>"
        RETURN

61800   PRINT #1, "<disconnect>"
        RETURN

61900   IF SOFF% > EOFF% THEN 61920
        FOR OFFSET%=SOFF% TO EOFF%
        BYTE%=PEEK(OFFSET%)
        IF (32 <= BYTE%) AND (BYTE% < 127) THEN PRINT #1, CHR$(BYTE%); : GOTO 61910
        PRINT #1, "<";
        IF BYTE% < 16 THEN PRINT #1, "0";
        PRINT #1, HEX$(BYTE%); ">";
61910   NEXT OFFSET%
61920   PRINT #1, ""
        RETURN

62000   SOFF%=51
        IF PEEK(50) = 1 THEN EOFF%=LENGTH%+7 ELSE EOFF%=50 + PEEK(50)
        GOSUB 61900
        RETURN

62100   PRINT #1, "<unknown type";TYPE%;"> ";
        SOFF%=50
        EOFF%=LENGTH%+7
        LIMIT%=10
        GOSUB 62200
        RETURN


62200   IF SOFF% > EOFF% THEN 62400
        FOR OFFSET%=SOFF% TO EOFF%
        BYTE%=PEEK(OFFSET%)
        IF (32 <= BYTE%) AND (BYTE% < 127) THEN PRINT #1, CHR$(BYTE%); " "; : GOTO 62310
        GOSUB 60000
62310   IF OFFSET% = EOFF% THEN 62300
        IF OFFSET% - SOFF% + 1 >= LIMIT% THEN PRINT #1, "..."; : GOTO 62400
62300   NEXT OFFSET%
62400   PRINT #1, ""
        RETURN

62500   IF (PEEK(50) < 1) OR (PEEK(50) > 3) THEN 62600
```

```
        ON PEEK(50) GOSUB 62800, 62900, 63000
        GOTO 62700

62600   IF TYPE%=6 THEN 62610
        PRINT #1, "<unknown subtype of type";TYPE%;"> ";
        SOFF%=50
        EOFF%=LENGTH%+7
        LIMIT%=6
        GOSUB 62200
        GOTO 62700

62610   GOSUB 62000
62700   RETURN

62800   SOFF%=51
        EOFF%=LENGTH%+7
        GOSUB 61900
        RETURN

62900   IF TYPE% = 6 THEN 62910
        PRINT #1, "<ADOS iob - "; IOBCMD$(PEEK(63)); ",drive:"; FNNUM$(PEEK(53));
        PRINT #1, "vol:"; FNNUM$(PEEK(54)); "track:"; FNNUM$(PEEK(55)); "sector:"; FNNUM$(PEEK(56)); ">"
        GOTO 62920

62910   PRINT #1, "<ADOS result iob - status:"; FNNUM$(PEEK(64)); ">"
62920   RETURN

63000   IF TYPE%=6 THEN 63100
        PRINT #1, "<AUCSD iob - "; IOBCMD$(PEEK(63)); " drive:"; FNNUM$(PEEK(53));
        PRINT #1, "block:"; FNNUM$(FNWPEEK(55)); "size:"; FNNUM$(FNWPEEK(68)); ">"
        GOTO 63200

63100   PRINT #1, "<AUCSD result iob - status:"; FNNUM$(PEEK(64)); ">"
63200   RETURN

63300   PRINT #1, "<ctrl - read from memory at $";
        GOTO 63600

63400   PRINT #1, "<ctrl - write to memory at $";
        GOTO 63600

63500   PRINT #1, "<ctrl - jump to $";
63600   TEMP = FNWPEEK(50) : GOSUB 60110
        IF TYPE% = 11 THEN 63700
        PRINT #1, "for $"; HEX$(PEEK(52) + 1); " bytes";
63700   PRINT #1, ">"
        RETURN

63800   PRINT #1, "<type"; TYPE%; "data> ";
        SOFF%=50 : EOFF% = LENGTH% + 8 : LIMIT% = 10
        GOSUB 62200
        RETURN

63900   PRINT #1, "<rfnm - type:"; peek(50); " sub:"; peek(51);
        PRINT #1, "size:";
        OFFSET% = 52 : GOSUB 61100
        PRINT #1, " offset:";
        OFFSET% = 56 : GOSUB 61100
```

```
        PRINT #1, ">"
        RETURN

64000   PRINT #1, "<abort>"
        RETURN

64100   PRINT #1, "<etna iob - "; IOBCMD$(PEEK(64)); " drive:"; FNNUM$(FNRWORD(66));
       .PRINT #1, "size:";
        OFFSET% = 72 : GOSUB 61100
        PRINT #1, " offset:";
        OFFSET% = 68 : GOSUB 61100
        PRINT #1, ">"
        RETURN

64200   SOFF% = 65 : EOFF% = 64 + PEEK(64) : GOSUB 61900
        RETURN

64300   PRINT #1, "<etna iob result - status1:"; FNNUM$(PEEK(78)); "status2:"; FNNUM$(PEEK(79)); ">"
        RETURN

64400   PRINT #1, "<biob - ";
        IF PEEK(64) = 2 THEN PRINT #1, "bsave at "; : GOTO 64410
        IF PEEK(65) = 1 THEN PRINT #1, "bload at "; : GOTO 64410
        PRINT #1, "brun at ";
64410   OFFSET% = 66 : GOSUB 61100
        PRINT #1, " for ";
        OFFSET% = 70 : GOSUB 61100
        IF (PEEK(64) <> 1) OR (PEEK(65) <> 2) THEN 64420
        PRINT #1, ", jump to ";
        OFFSET% = 74 : GOSUB 61100
64420   PRINT #1, ">"
        RETURN

64500   PRINT #1, "<dim - "; DIMCMD$(PEEK(50)); " drive:"; FNNUM$(FNRWORD(52));
        PRINT #1, DIMACC$(PEEK(56)); ; DIMSHR$(PEEK(57));
        PRINT #1, " t="; CHR$(PEEK(54)); " size: ";
        OFFSET%=58 : GOSUB 61100
        PRINT #1, ">"
        RETURN

64600   PRINT #1, "<boot request - machine type:"; FNNUM$(FNRWORD(50));
        PRINT #1, "subtype:"; FNNUM$(FNRWORD(52)); ">"
        RETURN
```

```
        page 60,132
        title   ETNA SNIFFER SOFTWARE (or hark I hear a message)

;*****************************************************************************
;
; COPYRIGHT ZYNAR LTD 1982.
; =========================
;
; ZYNAR CONIDENTIAL
; =================
;
;-----------------------------------------------------------------------------
;
;  NET 82 (ETNA) LISTENING DEVICE SOFTWARE
;  =======================================
;  TART (Transmit And Receive Totaliser)
;  =====================================
;
;   WRITTEN UNDER IBM DOS 2.x
;   =========================
;
;   VERSION : 0.0               DATE : OCTOBER 1982
;   --------                    -----
;
;   AUTHOR : JOHN H. A. ROWLAND  (B.A. (Cantab))       23/10/82
;   --------
;            Craig W. Payne      (Nothing (Lehigh U.)) 20/12/82
;
;-----------------------------------------------------------------------------
;
;   CHANGES TO PROGRAM LOG.
;      WHO:          WHEN:          WHY:
;      cwp          12-20-82        circular buffer logic
;      cwp          12-27-82        changed filter logic
;      cwp            1-4-83        added indicators in upper corner of screen
;      cwp            7-12-83       added counters in upper corner
;                                   moved board to d0000 to miss "real" board
;      cwp            9-28-83       changed to work with compiled BASIC
;      cwp            10-4-83       fixes to max_pop logic
;                                   removed memory zero code
;                                   compressed packet storage
;      cwp            10-10-83      allow one "don't care" and one care in
;                                     filter
;                                   pass back our stn from initialize
;                                   tag recons
;
;*****************************************************************************

        page


;-----------------------------------------------------------------------------
;
;               MEMORY DEFINITIONS
;
```

```
;-----------------------------------------------------------------------------

        .radix  16      ;HEX default for constants

DATASEG SEGMENT


page0   dw      100  dup (?)     ;Start of buffer RAM page 0
page1   db      200  dup (?)     ;Start of buffer RAM page 1
page2   db      200  dup (?)     ;Start of buffer RAM page 2
page3   db      200  dup (?)     ;Start of buffer RAM page 3
rimmask db      ?                ;Memory address offset of RIM mask register
rimstat equ     rimmask
rimcmd  db      ?                ;Memory address offset of RIM command register
niccon  db      ?                ;Memory address offset of I/F control register
nicstat equ     niccon           ;Memory address offset of I/F status register
irqind  db      ?                ;Memory address offset of irq level indicator
pitct0  db      ?                ;Memory address offset of PIT CT0 control
pitct1  db      ?                ;Memory address offset of PIT CT1 control
pitct2  db      ?                ;Memory address offset of PIT CT2 control
pitcon  db      ?                ;Memory address offset of PIT control register
userram db      7F8  dup (?)     ;User RAM area
rom     db      1000 dup (?)     ;NIC ROM area

DATASEG ENDS

        page


screen  segment

line_one db                      ;upper left hand corner of screen
        org     160d
line_two db                      ;on line below
        org     320d
line_three db                    ;third line

screen  ends

        page

;-----------------------------------------------------------------------------
;
;               EQUATES
;
;-----------------------------------------------------------------------------


baseadd equ     0d000            ;Base address for sniffer board

riset   equ     80               ;RI in RIM mask set
```

```
recbuf  equ     84              ;Common part of receive command for RIM (broadcast)
shortp  equ     05              ;Configure to short packets
longpa  equ     0DH             ;Configure to long packets
setpor  equ     02              ;Set soft POR command to 'niccon'
reset   equ     00              ;Reset command to 'niccon' (reset POR)
porfclr equ     1E              ;Clear RECON & POR flag command to RIM
rec_clr equ     16              ;clear RECON flag command
rec_flag equ    4               ;mask for RECON flag in status reg.
snifen  equ     04              ;Enable sniffer interrupt in NIC control
frrec   equ     08              ;Frame received by sniffer

ct0mod3 equ     36              ;PIT counter 0 mode 3 set up
ct1mod2 equ     74              ;PIT counter 1 mode 2 set up
ct1lch  equ     40              ;Latch counter 1 current count
count0  equ     2000D           ;1 ms prescale for counter 0
count1  equ     0               ;Maximal count value for counter 1
irqmask equ     21              ;8259 mask register address
keyb    equ     0FDH            ;IRQ level 2 mask byte (keyboard enabled)

digits          equ     10d     ;num. of digits in screen counts
sw2_port        equ     62h     ;io port of sense switch two
out_port        equ     61h     ;output port for main board
m_size_en       equ     2       ;raise this bit on out_port to
                                ;  read expansion memory size
read_key        equ     0       ;function code to rom keyboard routine
check_key       equ     1       ;check for pressed key
key_func        equ     16h     ;interrupt for keyboard functions
```

```
        page
CSEG    SEGMENT PUBLIC 'CODE'

                PUBLIC INITIALIZE, MAINPROG, PACKLOC


;-----------------------------------------------------------------------------
;
;               MEMORY LOCATIONS USED
;
;-----------------------------------------------------------------------------

intmask         db              ;Status of interrupt controller mask
resind          db      0       ;Reset / reboot indicator
stns            dw              ;Stations to be filtered out of the frames
count           dw              ;Number of packets left to receive
oldest          dw              ;segment of oldest packet in ram
ram_start       dw              ;segment of free ram begining
ram_top         dw              ;Memory segment address after last memory location
cur_pop         dw              ;current number of packets in buffer
max_pop         dw              ;mamimum    "      "     "    "    "
spacing         dw      11h     ;distance between packets in ram,
                                ;  used on seg. regs., actually 110h bytes


        page
```

```
;-------------------------------------------------------------------------
;
;       INITIALISATION
;       This section initialises the variables to be used in the main program
;       call format CALL INITIALIZE(stn%)
;
;-------------------------------------------------------------------------


        ASSUME  CS:CSEG,DS:NOTHING,SS:NOTHING,ES:NOTHING

initialize proc far

        push    bp              ;Save bp register
        mov     bp,sp
        push    es              ;Store es for return
        push    ds              ;Store ds for return

        mov     ax,baseadd      ;Set ds to point to sniffer board
        mov     ds,ax           ;

        ASSUME DS:DATASEG

        mov     niccon,setpor   ;Cause soft POR to sniffer
        mov     niccon,reset    ;Reset sniffer and clear interrupts

        mov     pitcon,ct0mod3  ;Set up counter 0 to 1 ms period clock
        mov     ax,count0       ;
        mov     pitct0,al       ;
        mov     pitct0,ah       ;

        mov     pitcon,ct1mod2  ;Set up counter 1 as a timestamp clock
        mov     ax,count1       ;
        mov     pitct1,al       ;
        mov     pitct1,ah       ;

        in      al,out_port     ;save current state
        push    ax
        or      al,m_size_en    ;enable mem. size switches
        out     out_port,al
        in      al,sw2_port     ;read expansion mem. size in 32k steps
        and     al,0fh          ;  in lower four bits
        mov     cl,3            ;convert to paragraphs
        sal     al,cl
        mov     ah,al
        sub     al,al
        add     ax,1000h        ;add in base 64k
        mov     ram_start,ax    ;save for later
        pop     ax              ;recover old state of out_port
        out     out_port,al

        mov     ds,ram_start    ;Set up start of RAM card pointer

        ASSUME  DS:NOTHING

        cmp     resind,0        ;Test for first time through
        jnz     cont            ;Flag is set so skip RAM tests
        mov     resind,0FF      ;Set flag to show been through once
```

```
        mov     ax,55aa         ;Set up test patterns
        mov     cx,0aa55
        mov     si,0            ;test word zero of each 64k
retry:
        mov     [si],ax         ;Write test pattern to memory
        cmp     [si],ax         ;Test that it was written
        jnz     gotsize         ;Not written so --->
        mov     [si],cx         ;Try a different test pattern just in case
        cmp     [si],cx         ;Test that it was written
        jnz     gotsize         ;Not written so --->
        mov     bx,ds
        add     bx,1000         ;Set ds to next 64k chunk
        mov     ds,bx           ;Set up new base segment for RAM test
        jmp     retry           ;Is this the memory size ?

gotsize:
        sub     bx,spacing      ;move back to last complete entry
        mov     ram_top,bx      ;Set up RAM size limit
        mov     ax,bx           ;calculate max_pop
        sub     ax,ram_start    ;size of buffer in paragraphs
        mov     dx,0            ;convert to packets
        div     spacing
        inc     ax
        mov     max_pop,ax

cont:   mov     ax,baseadd      ;Now set ds to point to sniffer board again
        mov     ds,ax           ;
        ASSUME DS:DATASEG

        mov     ax,page0        ;get selftest and our stn
        mov     al,ah
        sub     ah,ah
        mov     rimcmd,porfclr  ;Clear recon and POR flags in RIM
        mov     rimcmd,shortp   ;Set RIM for short packets (at present)

        pop     ds              ;Restore ds for host program
        pop     es              ;Restore es for host program
        mov     bx,[bp+6]       ;get param. pointer
        mov     [bx],ax         ;pass back our stn
        pop     bp              ;Restore bp register
        ret     2               ;Return
initialize endp

page


;-----------------------------------------------------------------------------
;
;       MAIN TRANSFER
;       This program transfers the incomming data to IBM RAM for retrieval by
;       the host program.
;       call format CALL MAINPROG(PACKETS%, FILTER%)
;       where
;       PACKETS% is passed as the number of packets to receive and
```

```
;         will be set to the number of packets received on return,
;       FILTER% is the two station filter packed into one word,
;
;-------------------------------------------------------------------------


        ASSUME  CS:CSEG,DS:NOTHING,SS:NOTHING,ES:NOTHING

mainprog proc   far

        push    bp              ;Save bp register
        mov     bp,sp           ;Prepare for parameter
        mov     si,[bp]+8       ;Get first parameter (Count value)
        mov     bx,[si]         ;
        mov     count,bx

        mov     si,[bp]+6       ;Get second parameter (Station filter)
        mov     bx,[si]         ;
        mov     stns,bx         ;

        push    ds              ;Store ds for return
        push    es              ;Store es for return

        mov     bx,baseadd      ;Set up data segment to sniffer board
        mov     ds,bx           ;

        mov     es,ram_start    ;Set up extra segment to IBM RAM

        ASSUME  DS:DATASEG

        mov     cur_pop,Ø       ;no packets in buffer

        in      al,irqmask      ;Get current 8259 mask
        mov     intmask,al      ;Store 8259 mask away for return
        mov     al,keyb         ;Set up keyboard mask for 8259
        out     irqmask,al      ;Output to 8259 mask

        cld                     ;set direction flag
        mov     rimcmd,recbuf   ;Set RIM to receive to buffer one command
        mov     niccon,snifen   ;Let it rip man !

main:
        mov     bx,baseadd      ;Set up data segment to sniffer board
        mov     ds,bx           ;
        mov     al,rimstat      ;get flags
        and     al,rec_flag     ;isolate recon flag
        je      no_recon        ;don't clear if not on
        mov     rimcmd,rec_clr  ;  else clear the recon flag
        mov     byte ptr es:258d,1      ;tag the packet
;tweak third line of screen
        mov     si,offset line_three+digits*2-2
        call    bump_count
no_recon:
        test    nicstat,frrec   ;Test for received packet
        jnz     contin          ;If received then do transfer

        mov     ah,check_key    ;check keyboard
```

```
        int     key_func
        jz      main            ;Nothing there so get on with program
        mov     ah,read_key     ;consume the pressed key
        int     key_func
        jmp     end             ;Key pressed so leave routine

contin: mov     niccon,reset    ;Stop sniffer until done transfer
        test    rimmask,riset   ;See if RIM is still set up to receive
        jz      nowgo           ;If RI not set then still receiving
        mov     rimcmd,recbuf   ;Set RIM to receive to buffer one command
nowgo:  mov     niccon,snifen   ;Let it rip man !

;tweak first line of screen
        mov     si,offset line_one+digits*2-2
        call    bump_count

        mov     bx,stns         ;Get station filter information
        mov     ax,page0        ;Get SID and DID
        cmp     ah,0            ;broadcast ?
        je      accept          ;accept if yes
        cmp     bl,0            ;don't care ?
        je      low_ok
        cmp     bl,al           ;match source ?
        je      low_ok
        cmp     bl,ah           ;match dest. ?
        jne     main            ; no, ignore
low_ok:
        cmp     bh,0            ;don't care ?
        je      accept
        cmp     bh,al           ;match source ?
        je      accept
        cmp     bh,ah           ;match dest. ?
        jne     main            ; no, ignore
accept:

;tweak second line of screen
        mov     si,offset line_two+digits*2-2
        call    bump_count

        mov     pitcon,ct1lch   ;Latch current timestamp from counter 1
        mov     al,pitct1       ;(lsb)   Read timestamp
        mov     ah,pitct1       ;(msb)

        mov     es:256D,ax      ;Store timestamp in RAM
        mov     di,0            ;Set up RAM offset
        mov     si,0            ;Set up sniffer pointer

        movsw                   ;Transfer SID & DID
        xor     bx,bx           ;Zero bx
        mov     bl,[si]         ;Obtain 3rd byte of transfered packet
        movsw                   ;Transfer the counts
        mov     cx,256D         ;set up loop count for transfer
        sub     cx,bx           ;

        shr     cx,1            ;Divide by two for word move
                                ;packet length is always even for xns
        mov     si,bx           ;Create pointer to data
        mov     di,4            ;Create pointer to IBM RAM
```

```
        rep     movsw           ;Do data transfer

        inc     cur_pop         ;bump population
        mov     ax,cur_pop
        cmp     ax,1            ;first time ?
        jnz     not_first       ;  skip this if not
        mov     oldest,es       ;set oldest packet pointer
not_first:
        cmp     ax,max_pop      ;check for full buffer
        jbe     not_full
        dec     cur_pop         ;  else decrease buffer pop
        mov     ax,oldest       ;advance the oldest packet pointer
        call    bump_pointer
        mov     oldest,ax
not_full:
        mov     ax,es           ;get data pointer
        call    bump_pointer    ;Point to next bit of RAM
        mov     es,ax           ;move pointer back to es register
        mov     byte ptr es:258d,0      ;clear recon tag
        dec     count           ;Decrement number of received packets
        jz      end
        jmp     main            ;Now go and do the next packet

end:
        mov     bx,cur_pop      ;get number of packets in buffer
        mov     niccon,reset    ;Prevent any impending interrupts
        mov     al,intmask      ;Recall 8259 mask from start of routine
        out     irqmask,al      ;Restore updated 8259 mask

        pop     es              ;Restore es for host program
        pop     ds              ;Restore ds for host program
        mov     di,[bp]+8       ;Save return packet count for host program
        mov     [di],bx         ;
        pop     bp              ;Restore bp register
        ret     4               ;Return (pop 2 parameters)


mainprog endp

;
; locate the segment base of a packet
; call packloc(packno%, packbase%)
;
packloc proc    far
        assume cs:cseg, ds:nothing
        push    bp              ;Save bp register
        mov     bp,sp           ;Prepare for parameter
        mov     si,[bp+8]       ;Get packet number
        mov     ax,[si]
        dec     ax              ;index array based at zero, not one
        mul     spacing         ;scale by element size
        add     ax,oldest       ;index from oldest packet (number one)
        mov     cx,ax           ;save index
        cmp     ax,ram_top      ;need to wrap around ?
        jc      inbounds        ;jump if no
        mov     ax,max_pop      ;calc size of correction
        mul     spacing
        sub     cx,ax           ;wrap around
```

```
inbounds:
        mov     si,[bp+6]       ;pass back packet's segment
        mov     [si],cx
        pop     bp              ;Restore bp register
        ret     4               ;Return (pop 2 parameters)
packloc endp
;
inc_table:
                org     inc_table+' '
                db      '1'
                org     inc_table+'0'
                db      '1234567890'
;
;
; si : start of ascii sequence to inc
;
bump_count      proc    near
                push    bx
                push    cx
                push    ds              ;save normal data seg.
                mov     ax,0b000h
                mov     ds,ax           ;set seg. to screen's seg.

                assume  ds:screen

                mov     bh,0
                mov     cx,digits
inc_loop:
                mov     bl,[si]
                mov     al,byte ptr inc_table[bx]
                mov     [si],al
                sub     si,2
                cmp     al,'0'
                loopz   inc_loop
                pop     ds
                assume  ds:dataseg
                pop     cx
                pop     bx
                ret
bump_count      endp

;-----------------------------------------------------------------------------
;
;       BUMP_POINTER
;       bump a segment pointer by packet spacing and check for wrap-around
;
;-----------------------------------------------------------------------------
bump_pointer proc near
        add     ax,spacing      ;move to next packet
        cmp     ax,ram_top      ;Check whether this location exists
        jc      not_end         ;Its all right so -->
        mov     ax,ram_start    ;wrap to start of buffer
not_end:
        ret
bump_pointer endp

CSEG    ends
```

```
        end
```