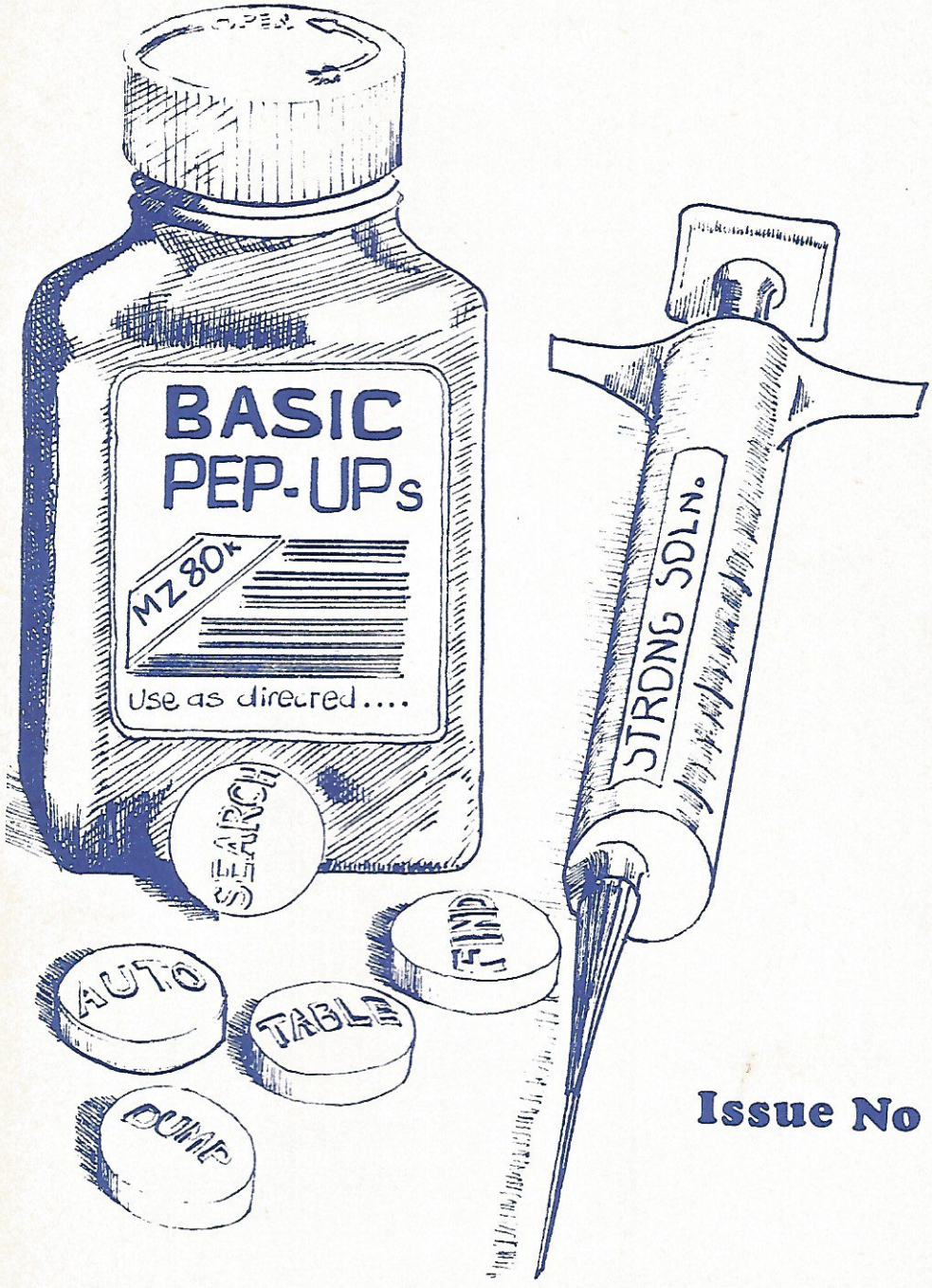
SHARPSOFT



Issue No 14

SHARPSOFT USER NOTES

Issue No. 14

C O N T E N T S

SHARPSOFT USER NOTES

ISSUE NO. 14

EDITORIAL

Our thanks to all readers who have sent articles, programs and letters. Please keep sending your material for publication especially on the MZ-80A. PASCAL, FORTH and machine code programs would make a welcome change from BASIC!

A number of our readers are keen amateur radio enthusiasts. The MZ-80K appears to be widely used by amateur's for translating morse and teletype radio transmissions or the calculation of range, bearing and /or location of other amateurs. Mr. K.E.J. Bowden and Mr. K.R Simmonds have contributed an elegant amateur radio program. In their article they describe how the daily position of the Oscar 10 satellite can be forecast using the MZ-80K. This program should also run on the MZ-80A without modification.

Also featured in this issue, for small business users, is an update to Larry Galliford's VAT Calculator in Issue 11, plus another very useful Small Business Program running on the MZ-80B from Mr. D. Barradine.

Various queries regarding the use and content of the CP/M Library disks has prompted us to publish an information/help sheet on page 29 to clarify matters.

Following the Fourth and final Part of the 'MZ-80K Gets Shugart Drives' is Peter Sydenham's 'Inside the MZ-80K Disc' an interesting and informative article on exactly what happens when discs are in use on the MZ-80K.

Once again we have amalgamated Members' Letters and Listings as it is becoming increasingly difficult to completely separate your questions, answers, tips and views, but keep them rolling in!

Issue 15 of the User Notes will be published late September 1984.

(This Issue is in fact our missing Issue 13 which the Post Office finally delivered 'back to us!' Should anything be out of date or repeated in the Listings Issue we (and the Post Office) apologise.

MIKE BRINSON

EDITOR

JODRELL BANK - EAT YOUR HEART OUT!

By K.E.J. Bowden and K.R. Simmonds

Nowadays, no radio shack is complete without the computer lying beside the transceivers, aerial switch boxes, power supply units and test gear.

Most amateurs and shortwave listeners are aware of the potential of the computer to translate morse and radio teletype and to calculate range, bearing and/or location or other amateurs. Programs for these applications are fairly easily available for most popular computers.

However, both of the authors own MZ-80K machines and found that a particular program which could forecast the daily positions of the current Oscar 10 amateur satellite did not seem to exist for Sharp machines and hence required some thought. Fortunately, a program devised by Dr. Thomas Clark from Maryland, U.S.A., and apparently written for the Sinclair ZX81 was located. This program was intended as a generalized routine for any orbital satellite and was thus easily adaptable.

The authors' version of the program which runs on SP-5025 BASIC and uses 9.5K bytes, has three main, menu fed options as follows:-

> Firstly a table giving azimuth, elevation, range etc. Secondly a table listing the principal world cities in range of Oscar 10 at a particularr instant and, thirdly the satellite's position is shown on a choice of two world maps - polar or cartesian.

These maps and routines used to make the satellite track across them accurately are the authors' own devising and add a great deal of interest as graphics always do.

In all three menu options, day, date, start time and step interval are user defined and control is returned to the user by pressing 'M' for menu.

This version of the program has been used for several months now and its predictions are quite accurate, certainly within the beam width of a standard multi-element beam aerial.

Some updating of the program is required occasionally as Oscar 10 is slowly moving away from its intended orbit. New parameters for Oscar 10 are published in various radio magazines and the program is well documented with 'REM' statements. Modification of satellite data is thus a simple matter.

If you wish to type in this program, remember to enter YOUR latitude, longitude, station height and data for your nearest town. Oscar 10's "footprint" varies quite considerably even over a few tens of miles.

In conclusion, therefore, it will be realised that the Sharp machines belonging to the authors are in constant use. The next project must be to let the computer directly control the beam antenna's phase, azimuth and elevation. Jodrell Bank - eat your heart out!

```
10 REM******** OSCARTEN **************
20 PRINT "[E"
30 PRINT TAB(3);"OSCARTEN PROGRAM FOR SHARP MZ80K"
40 PRINT TAB(5);"PREPARED BY G6UCV AND G6TWR"
50 PRINT TAB(2);"**********************************":PRINT
60 REM GO TO START.:GOTO510
70 REM SUBROUTINE "IN RANGE?"*********
80 CNG=ABS(W5-LN):IF CNG>180 THEN CNG = 360 - CNG
90 D1 =((SIN(LT*P0))*(SIN(L5*P0))+(COS(LT*P0))*(COS(L5*P0))*(COS(CNG*P0)))
100 D1 =(-ATN(D1 /SQR(-D1 *D1 +1))+π/2)/P0
110 IF(DZ>=D1)*(P$="***GUILDFORD***") THENPRINT TAB(6);P$:GOTO130
120 IF DZ>=D1 THEN PRINT TAB(9);P$
130 RETURN
140 REM SUBROUTINE "PLACES"**********
150 REM ENTER NAME,LATITUDE AND LONGITUDE OF NEAREST CITY
160 LT=51.28:LN=0.56:P$="***GUILDFORD***":GOSUB70
170 LT=57.5:LN=4.25:P$="INVERNESS":GOSUB70
180 LT=68.5:LN=-19.5:P$="TROMSO":GOSUB70
190 LT=42:LN=-12.5:P$="ROME":GOSUB70
200 LT=52.5:LN=-12.5:P$="BERLIN":GOSUB70
210 LT=41:LN=74:P$="NEW YORK":GOSUB70
220 LT=32.5:LN=96:P$="DALLAS":GOSUB70
230 LT=37.6:LN=122.5:P$="SAN FRANCISCO":GOSUB70
240 LT=-23:LN=-43:P$="RIO":GOSUB70
250 LT=-60:LN=-51:P$="PORT STANLEY":GOSUB70
260 LT=-33.9:LN=342:P$="CAPETOWN":GOSUB70
270 LT=1.3:LN=256.2:P$="SINGAPORE":GOSUB70
280 LT=32:LN=-35:P$="TEL AVIV":GOSUB70
290 LT=22.2:LN=245.75:P$="HONG KONG":GOSUB70
300 LT=35.75:LN=220.25:P$="TOKYO":GOSUB70
310 LT=-12.3:LN=229:P$="DARWIN":GOSUB70
320 LT=-32:LN=244:P$="PERTH":GOSUB70
330 LT=-34:LN=209:P$="SYDNEY":GOSUB70
340 LT=-41.3:LN=185:P$="WELLINGTON":GOSUB70
350 LT=21:LN=158:P$="HAWAII":GOSUB70
360 LT=61:LN=148:P$="ANCHORAGE":GOSUB70
370 LT=77:LN=68:P$="THULE":GOSUB70
380 LT=56:LN=-38:P$="MOSCOW":GOSUB70
390 LT=55:LN=-83:P$="NOVOSIBIRSK":GOSUB70
400 LT=5:LN=-80:P$="COLOMBO":GOSUB70
410 PRINT "Enter 'M' for menu,'Z' to step one hour"
420 RETURN
430 REM SUBROUTINE "TIME STRING"****
440 D$=STR$(T4):E$="0000":U=4-LEN(D$)
450 IF U=0 THEN T$=D$
460 IF U=1 THEN T$="0"+D$
470 IF U=2 THEN T$="00"+D$
480 IF U=3 THEN T$="000"+D$
490 IF U=4 THEN T$=E$
500 RETURN
510 PRINT:PRINT:PRINT
520 PRINT TAB(5);"OSCAR 10 INFORMATION BOARD": PRINT:PRINT
530 PRINT "Enter reauired day,month,year.eg 21,1,84"
540 INPUT DD,MM,YY
550 PRINT"[E"
```

```
560 REM FETCH PARAMETERS:GOSUB1860:P0=π/180:R0=6378.16:F=1/298.25
570 PRINT TAB(9);"MENU SELECTION":PRINT:PRINT
580 PRINT TAB(3);"Enter menu number required":PRINT:PRINT
590 PRINT TAB(3);" 1. SATELLITE POSITION MAP":PRINT
600 PRINT TAB(3);" 2. AZ/EL TRACKING TABLE":PRINT
610 PRINT TAB(3);" 3. DX BOARD":PRINT:PRINT
620 INPUT "Choose now ";H2
630 IF ( H2<1)+( H2>3) THEN580
640 PRINT:PRINT:PRINT
650 INPUT "Enter start time gmt hhmm  ";A$:PRINT:PRINT:PRINT:I$="M"
660 IF H2=3 THEN690
670 INPUT "Enter step interval required.(mins)  ";DT
680 ST=DT/(24*60)
690 LR=0:T1=VAL(A$)
700 D6=INT(YY/4)-INT((YY-1)/4)+212+DD
710 D6=D6+(SGN(MM-8)*(INT(.5+(ABS((MM-8)*30.5)))))
720 IF MM<3 THEN D6=DD+(MM-1)*31
730 D6=D6+365*(YY-YS)+INT((YY-1)/4)-INT((YS-1)/4)
740 T=(INT(T1/100))/24+(T1-(INT(T1/100))*100)/(60*24)+D6
750 IF N0>.1 THEN A0=((G0/(N0↑2))↑(1/3))
760 IF N0<=.1 THEN N0=SQR(G0/(A0↑3))
770 E2=1-E0↑2:E1=SQR(E2):Q0=M0/360 +K0
780 IF (H2=1)+(H2=3) THEN840
790 PRINT "E"
800 PRINT "OSCAR 10 ON ";DD;"";MM;"";YY;" COMMENCING ";A$;" hrs"
810 PRINT "*****************************************":PRINT
820 PRINT SPC(1);"GMT";SPC(3);"AZ";SPC(2);"EL";SPC(2);"RANGE(km)";SPC(2);
830 PRINT "PHA";SPC(4);"DOP(hz)":PRINT
840 K2=9.95*((R0/A0)↑3.5)/(E2↑2)
850 S1=SIN(I0*P0):C1=COS(I0*P0)
860 O=O0-(T-T0)*K2*C1
870 S0=SIN(O*P0):C0=COS(O*P0)
880 W=W0+(T-T0)*K2*(2.5*(C1↑2)-.5)
890 S2=SIN(W*P0):C2=COS(W*P0)
900 DIM C(2,2)
910 C(0,0)=(C2*C0)-(S2*S0*C1)
920 C(0,1)=-(S2*C0)-(C2*S0*C1)
930 C(1,0)=(C2*S0)+(S2*C0*C1)
940 C(1,1)=-(S2*S0)+(C2*C0*C1)
950 C(2,0)=(S2*S1):C(2,1)=(C2*S1)
960 Q=N0*(T-T0)+Q0:K=INT(Q):P=INT((Q-K)*256):M=(Q-K)*2*π
970 E=M+E0*SIN(M)+.5*(E0↑2)*SIN(2*M)
980 S3=SIN(E):C3=COS(E):R3=1-E0*C3
990 M1=E-E0*S3 :M5=M1-M
1000 IF ABS(M5)<1E-6 THEN1020
1010 E=E-M5/R3:GOTO980
1020 X0=A0*(C3-E0):Y0=A0*E1*S3:R=A0*R3
1030 X1=X0*C(0,0)+Y0*C(0,1)
1040 Y1=X0*C(1,0)+Y0*C(1,1)
1050 Z1=X0*C(2,0)+Y0*C(2,1)
1060 G7=T*G1+G2:G7=(G7-(INT(G7)))*2*π
1070 S7=-SIN(G7):C7=COS(G7)
1080 X=(X1*C7)-(Y1*S7):Y=(X1*S7)+(Y1*C7)
1090 Z=Z1:L8=L9*P0
1100 S9=SIN(L8):C9=COS(L8)
1110 S8=SIN(-W9*P0):C8=COS(W9*P0)
1120 R9=R0*(1-(F/2)+(F/2)*COS(2*L8))+H9/1000
1130 L8=ATN((1-F)↑2*S9/C9)
1140 Z9=R9*SIN(L8):X9=R9*COS(L8)*C8:Y9=R9*COS(L8)*S8
1150 X5=(X-X9):Y5=(Y-Y9):Z5=(Z-Z9)
1160 RA=SQR(X5*X5+Y5*Y5+Z5*Z5)
1170 DZ=6378/R:DZ=(-ATN(DZ/(SQR(-DZ*DZ+1)))+π/2)/P0
1180 Z8=(X5*C8*C9)+(Y5*S8*C9)+(Z5*S9)
1190 X8=-(X5*C8*S9)-(Y5*S8*S9)+(Z5*C9):Y8=(Y5*C8)-(X5*S8)
1200 S5=Z8/RA:C5=SQR(1-S5*S5)
1210 EL=(ATN(S5/C5))/P0:AZ=(ATN(Y8/X8))/P0
```

```
1220 B5=Z/R:L5=(ATN(B5/(SQR(1-B5*B5))))/P0:W5=(ATN(Y/X))/P0
1230 IF (X<0) THEN W5=-(180+W5)
1240 IF (X>0) THEN W5=-W5
1250 IF (X<0)*(Y<0) THEN W5=360+W5
1260 IF (X=0)*(Y>0) THEN W5=270
1270 IF (X=0)*(Y<0) THEN W5=90
1280 IF (X8<0) THEN AZ=AZ+180
1290 IF (X8>0)*(Y8<0) THEN AZ=360+AZ
1300 IF (X8=0)*(Y8>0) THEN AZ=90
1310 IF (X8=0)*(Y8=0) THEN AZ=90
1320 IF (X8=0)*(Y8<0) THEN AZ=270
1330 T4=(INT((T-INT(T))*2400+.5))/100
1340 T4=INT((100*((T4-INT(T4))*.6+INT(T4)))+.5)
1350 IF T4-INT(T4)<.5 THEN T4=INT(T4)
1360 IF T4-INT(T4)=.5 THEN T4=INT(T4+.5)
1370 IF T4-INT(T4)>.5 THEN T4=INT(T4+.5)
1380 IF T4=2400 THEN T4=0
1390 IF T4=0 THEN DD=DD+1
1400 REM GET TIME STRING:GOSUB430
1410 T=T+ST:IF  H2=1 THEN1730
1420 IF H2=3 THEN1530:REM FOR A TITLE
1430 SV=(LR-RA)/(DT*60):DP=SV*486.33333:REM DOPPLER SHIFT AT 145.9Mhz
1440 PRINT T$;TAB(5);INT(AZ);TAB(10);INT(EL);TAB(15);INT(RA);TAB(25);P;
1450 IF LR=0 THENPRINT TAB(33);" ---":GOTO1470
1460 PRINT TAB(33);INT(DP)
1470 IF(EL<=1) THENPRINT:PRINT "OSCAR10 BELOW YOUR HORIZON":GOTO1490
1480 REM GET MORE VALUES:LR=RA:GOTO860
1490 PRINT:PRINT "Enter 'T' for new time,'M' for menu"
1500 GET CH$:IF CH$="" THEN1500
1510 IF CH$="T" THENPRINT "Ø":GOTO640
1520 IF CH$="M" THENPRINT "Ø":GOTO1610
1530 PRINT"Ø":PRINT "DX STATE AT ";T$;" hrs on ";DD;"";MM;"";YY
1540 PRINT "**********************************":PRINT:PRINT
1550 PRINT TAB(1);"OSCAR 10 CAN BE HEARD IN ":PRINT
1560 REM GET "PLACES":GOSUB140:GOSUB430
1570 GET Z$:IF Z$="" THEN1570
1580 IF Z$="M" THEN1610
1590 IF Z$="Z" THENH2=3:A$=STR$(VAL(A$)+100):GOTO690
1600 GOTO1570
1610 PRINT "Ø":M$=""
1620 PRINT:PRINT:PRINT "Enter"
1630 PRINT:PRINT:PRINT "1 For new orbit"
1640 PRINT:PRINT:PRINT "2 for satellite position on map"
1650 PRINT:PRINT:PRINT "3 for dx board"
1660 PRINT:PRINT:PRINT "4 for tracking table"
1670 GET B$:IF B$="" THEN1670
1680 IF B$="1" THENPRINT "Ø":H2=1:GOTO510
1690 IF B$="2" THENPRINT "Ø":H2=1:GOTO640
1700 IF B$="3" THENPRINT "Ø":H2=3:GOTO640
1710 IF B$="4" THENPRINT "Ø":H2=2:GOTO640
1720 PRINT "Ø":GOTO1620
1730 IF (M$="C")+(M$="P") THEN1770
1740 PRINT:PRINT:PRINT:PRINT "Key 'C' for Cartesian or 'P' for Polar map"
1750 GET M$:IF M$="" THEN1750
1760 IF I$="M" THEN PRINT "Ø"
1770 PRINT"Ø";
1780 IF EL>=0 THENPRINT" IN RANGE AZ=";INT(AZ);" AT ";T$;" hrs ";DD;MM;YY;"   "
1790 IF EL<0 THENPRINT "OUT RANGE AZ=";INT(AZ);" AT ";T$;" hrs ";DD;MM;YY;"   "
1800 IF M$="C" THENGOSUB2040
1810 IF M$="P" THENGOSUB2300
1820 GET I$:IF I$="" THENSETSX,SY:RESETSX,SY:GOTO1820
1830 IF I$="M" THEN1610:REM FOR MENU
1840 IF I$="Z" THENH2=1:SETSX,SY:GOTO860
1850 GOTO1820
1860 REM SUBROUTINE "STATION AND SATELLITE DETAILS**************************
1870 REM ENTER YEAR NUMBER OF ORBIT DATA EPOCH:YS=83
```

```
1880 REM ENTER DAY OF ORBIT DATA EPOCH:T0=285.5
1890 REM ENTER ORBIT INCLINATION ANGLE IN DEGREES:I0=25.876
1900 REM ENTER ORBIT REVOLUTION COUNT AT EPOCH TIME.K0=249
1910 REM ENTER MEAN ANOMALY AT EPOCH IN DEG:M0=17.222
1920 REM ENTER MEAN MOTION IN REV PER DAY:N0=2.0584769
1930 REM ENTER ECCENTRICITY:E0=0.608
1940 REM ENTER ARGUMENT OF PERIGEE IN DEG:W0=225.462
1950 REM ENTER RIGHT ASCENSION OF ASCENDING MODE:O0=227.616
1960 REM ENTER GRAVITATIONAL CONSTANT:  G0=7.536979E+13
1970 REM ENTER SUN/SIDERIAL TIME CONVERSION:G1=1.0027379093
1980 REM ENTER ANNUAL SIDERIAL TIME CORRECTION
1990 IF YY=84 THEN G2=.2746066342
2000 REM ENTER STATION LATITUDE IN DEG:L9=51.265
2010 REM ENTER STATION LONGITUDE IN DEG WEST:W9=0.5667
2020 REM ENTER STATION HEIGHT ABOVE SEA LEVEL IN METRES:H9=35
2030 RETURN
2040 REM SUBROUTINE "CARTESIAN MAP"****
2050 SX=37-INT(W5/5.29):SY=26-INT(L5/3.91)
2060 IF I$="Z" THEN RETURN
2070 PRINTTAB(2);"
2080 PRINTTAB(2);"
2090 PRINTTAB(2);"
2100 PRINTTAB(2);"
2110 PRINTTAB(2);"
2120 PRINTTAB(2);"
2130 PRINTTAB(2);"
2140 PRINTTAB(2);"
2150 PRINTTAB(2);"
2160 PRINTTAB(2);"
2170 PRINTTAB(2);"
2180 PRINTTAB(2);"
2190 PRINTTAB(2);"
2200 PRINTTAB(2);"
2210 PRINTTAB(2);"
2220 PRINTTAB(2);"
2230 PRINTTAB(2);"
2240 PRINTTAB(2);"
2250 PRINTTAB(2);"      Key 'Z' to step
2260 PRINTTAB(2);"
2270 PRINTTAB(2);"      Key 'M' for menu
2280 PRINTTAB(2);"
2290 RETURN
2300 REM SUBROUTINE "POLAR MAP"*******
2310 SX=((51-L5)*SIN(AZ*P0))/3.72:SX=37+INT(SX)
2320 SY=((51-L5)*COS(AZ*P0))/3.95:SY=29-INT(SY)
2330 IF I$="Z" THEN RETURN
2340 PRINTTAB(3);"
2350 PRINTTAB(3);"
2360 PRINTTAB(3);"
2370 PRINTTAB(3);" Key 'Z' to step
2380 PRINTTAB(3);" Key 'M' for menu
2390 PRINTTAB(3);"
2400 PRINTTAB(3);"
2410 PRINTTAB(3);"
2420 PRINTTAB(3);"
2430 PRINTTAB(3);"
2440 PRINTTAB(3);"
2450 PRINTTAB(3);"
2460 PRINTTAB(3);"
2470 PRINTTAB(3);"
2480 PRINTTAB(3);"
2490 PRINTTAB(3);"
2500 PRINTTAB(3);"
2510 PRINTTAB(3);"
2520 PRINTTAB(3);"
2530 PRINTTAB(3);"
```

```
2540 PRINTTAB(3);   "                                    "
2550 PRINTTAB(3);   "                                    "
2560 RETURN
2570 END
```

MAKING A SECURITY COPY OF DISC BASIC SP-6015

WITH THE ARDEN SYSTEMS TOOLPAK ON THE MZ-80K

from M. Wiechowski in Sweden

The Toolpak diskette is uncopyable which presents a serious security problem. I never use the original diskettes as this could mean, if things go wrong, an irreversible damage of purchased programs.

I could not find a way of making a security copy of the Toolpak diskette (is there anybody who can?) However, it is quite easy to save the Disc Basic together with the Toolpak on tape and then transfer the tape contents to a disc.

1. Boot up the Disc BASIC and load the Toolpak as described in the Manual (p. 2).

2. Enter the following program:

```
100 F$="DISC BASIC":A=4336
110 FT=1:B=4608:L=21291:E=8698
120 FOR I=1 TO 16
130 POKE A+I,13:NEXT I
140 FOR I=1 TO LEN(F$)
150 POKE A+I,ASC(MID$(F$,I,1))
160 NEXT I
170 POKE A,FT
180 POKE A+18,L-256*INT(L/256)
190 POKE A+19,INT(L/256)
200 POKE A+20,B-256*INT(B/256)
210 POKE A+21,INT(B/256)
220 POKE A+22,E-256*INT(E/256)
230 POKE A+23,INT(E/256)
240 USR(33):USR(36)
250 END
```

3. Fit a blank cassette into the recorder.

4. Run the above program.

5. Rewind the cassette and copy its contents onto a (formatted!) disc by means of the stndard Sharp utility "FILING - CMT" as described in the Sharp manual (p. 177).

The Disc Basic including the Toolpak is now safely saved and may be run by RUN "DISC BASIC" like every machine code (OBJ) file. You may change the name "DISC BASIC" on line 100 to any valid file name, if you like.

BEGINNER'S TUTORIAL GUIDE TO PASCAL

PART 4

DATA TYPES

28. A data TYPE is an attribute possessed by an item that can have a value - variables, constants, literals, functions and expressions all have a type. Each of these items is completely determined by its type, that is

    1. The exact range of values the item may have.
    2. How these values are represented.
    3. Which operators may be used with these values.

29. In PASCAL data types are classified according to their representation:

    (a) SIMPLE (Scalor) (b) STRUCTURED (c) POINTER

    Standard
    - Integer
    - Real
    - Character
    - Boolean

    User-defined
    - Subrange
    - Enumeration

    Array
    Record
    Set
    File*

* The FILE type is not implemented in HISOFT PASCAL 4T.

Structured data types can be complex, but are in all cases formed from the simple types.

PASCAL "data abstractions" allow the programmer to create logical data structures that closely match the data's physical/conceptional form.

30. Types and Variables.

It is important that the PASCAL programmer distinguishes between a data type and a variable. A variable represents a value. The value is hehd in a group of RAM locations in the computer's memory. In contrast, a data type represents a set of characteristics possessed by a variable. The type of a variable determines exactly what values that variable may have, what the representation of that variable is, and which operators may be used with that variable.

PASCAL is called a STRONGLY-TYPED language. This implies that data types play a very important role in determining the correctness of a program.

31. User-defined Types.

PASCAL allows programmers to define their own data types. These types are either enumerated, subrange or user defined structured types. User-defined types make programs much more readable.

32. TYPE declarations.

TYPEs are defined in the type declaration part, which is headed by the PASCAL reserved word TYPE:

```
TYPE
    < declaration list >
```

where the < declaration list > is one or more type declarations, separated by semicolons. Each member of the ist has the form:

< identifier > = < type definition >

33. Enumerated types.

An enumerated type has a name and a list of values that a variable of the enumerated type can have, for example:

```
TYPE
    FRUIT  = (APPLE, CHERRY, ORANGE, BANANA, GRAPE);
    ANIMAL = (CAT, DOG, BEAR, TIGER, LION);
    DAYS   = (SUN, MON, TUE, WED, THUR, FRI, SAT);
```

Note the predefined typa Boolean is an enumerated type, where:

```
BOOLEAN  =  (FALSE, TRUE);
```

An enumerated type defines a sequence for its values. The ordinal position of a value within an enumerated type is given by the predefined PASCAL function ORD.

In the above TYPE definitions

DOG < BEAR

because ORD(DOG) < ORD(BEAR)

Examples:

```
TYPE
   DAYS = (SUN, MON, TUE, WED, THUR, FRI, SAT);

VAR
   TODAY,YESTERDAY : DAYS;

                BEGIN
            (     TODAY:=TUE;
            (     YESTERDAY:=MON;
            (     IF TODAY <> THUR THEN YESTERDAY:=THUR;
  Examples  (     CASE TODAY OF
     of     (
   PASCAL   (        MON  :  STARTWEEK;
 enumerated (      FRI  :  ENDWEEK
constructions(     ELSE     OVERTIME
            (    END;
                END.
```

34. Enumerated types - operators.

The following operators apply to enumerated types:

Assignment	:=
Relational	< <= = <> >= > IN
Predefined functions	ORD PRED SUCC

PRED is the predecessor function
SUCC is the successor function.

In the examples given in Section 33.

```
PRED (BEAR) = DOG
PRED (SAT)  = FRI

SUCC (TIGER) = LION
SUCC (MON)   = TUE
```

35. Subrange Types

A subrange type is defined by the user a a contiguous subset of a previously defined enumerated type, or the standard integer or Char types. The TYPE from which a SUBRANGE type is formed is called the BASE type.

Examples:

```
TYPE
   DAYS = (SUN, MON, TUE, WED, THUR, FRI, SAT);
                          (* BASE TYPE *)

   WORKDAYS = MON.. FRI; (* SUSBRANGE TYPE *)


   POSINT   = 0..32767;  (* SUBRANGE OF INTEGER *)
```

SUBRANGE TYPES have a representation and use operators similar to their BASE type.

36. Given

```
TYPE
   PCOLOUR = (RED, YELLOW, BLUE);

VAR
   COLOUR : PCOLOUR;

BEGIN
   COLOUR:=YELLOW;
   .
   .
   .
```

which of the following are True?

```
RED < YELLOW
YELLOW > BLUE
COLOUR = RED
ORD (RED) = 0
ORD (BLUE) = 3
```

```
SUCC (YELLOW) = BLUE
PRED (BLUE) = RED
SUCC (PRED(COLOUR)) = ?
```

37. Un-named types.

When a new TYPE is defined it is given a type identifier. It is however, possible to define a new type without naming it. This is done by defining the type when declaring variables of that type - for example:

```
VAR
    TODAY, YESTERDAY : (SUN, MON, TUE, WED,
                        THUR, FRI, SAT);
    POSINT           : 0..32767;
```

38. TYPE CHECKING

A PASCAL compiler, because the language is a strongly-typed language, checks that all TYPES are consistent and that the correct operators are being used on operands - for example if:

A is REAL, then ORD (A) yields an error.

39. RANGE CHECKING

Logical errors in programs often result in variables having incorrect values - for example floating point values too small or too large. By carefully defining subrange types, the programmer can have the PASCAL compiler check for many of these errors at run time. A range error at run time will abort the program. Range errors are also detected at compile time.

MORE PASCAL NEXT ISSUE

THE LIGHTCYCLE GAME FROM TRON

in Hisoft PASCAL by Alan Stevens

I enclose a game written in Hisoft PASCAL for the MZ-80K. It is a two player game based on the lightcycle chase game from the film TRON. Each player controls a lightcycle and must avoid crashing into any walls or trails for as long as possible.

The game is trivial (though a favourite with my children) but the program contains a couple of features which may be of interest to SUN readers.

Firstly, since the game requires continuous action from both players at once, a way must be found of testing for two, simultaneous key presses. This is done in TRON by examining each player's keys in turn to see if they have been pressed. The function 'KEY' (lines 11 to 15) sends a specific signal to the keyboard via address E000, and by examining address E001 can tell if a particular key has been pressed. The 'strobe' signal must be a number from 0 to 9, each one of which refers to a different group of keys. Each key in a group then returns a different number in E001. Function 'KEY' is used in procedure 'Nextposn' (lines 72 to 82).

The major problem with fast graphics in PASCAL on the MZ-80K though, is that the standard PASCAL WRITE procedure is much too slow (it calls the video blanking routine before each character output). This can be overcome of course by POKEing directly to the screen as in BASIC. The problem is that when several screen positions have to be changed the POKE procedure has to be called several times and the resulting picture is inevitably covered with that irritating 'snow'. In an attempt to overcome this problem I have used the fact that the Hisoft PASCAL POKE is much more powerful than any BASIC POKE. In PASCAL one may POKE a whole screenful of characters onto the screen in one go. Hence, in TRON, the 'action' takes place in an array called 'Screen'. This 'Screen' is then POKEd onto the real screen after a call to the video blanking routine.

Unfortunately this doesn't work!!

That is, this technique does not eliminate the 'snow'. However, the result is not too bad so I've left it as it is rather than attempt a machine-code solution. Although I have not tried it I have a sneaking suspicion that POKEing the 'action' directly onto the real screen would be just as good in this program!!

```
6EEC    2 PROGRAM TRON;
6EEC    3 (* The lightcycle game from TRON *)
6EEC    4 CONST null=CHR(0); cr=CHR(13);
6EEC    5 VAR Screen, IS : ARRAY[0..24,0..39] OF CHAR;
6EF5    6       s :ARRAY[1..2,1..2] OF CHAR;
6EF5    7      h : ARRAY[1..2,-2..2] OF CHAR;
6EF5    8      UB,LB,UH,LH,r : CHAR;
6EF5    9      I,J,K,L,DI,DJ,DK,DL,N,US,LS : INTEGER;
6EF5   10      end : BOOLEAN;
6EF5   11 FUNCTION KEY(strobe:CHAR):CHAR;
6EF8   12 CONST out=#E000; in=#E001;
6EF8   13 BEGIN
6F10   14 POKE(out,strobe); KEY:=PEEK(in,CHAR)
6F1D   15 END;
```

```
6F27    16 FUNCTION OK(row,col:INTEGER):BOOLEAN;
6F2A    17 BEGIN
6F42    18 OK:=(Screen[row,col]=null)
6F86    19 END;
6F92    20 FUNCTION SPEED:INTEGER;
6F95    21 VAR k:INTEGER;
6F95    22 BEGIN
6FAD    23 REPEAT PAGE;
6FB5    24        WRITELN('What speed? (1 - 5)',cr);
6FDB    25        WRITELN('1 = fast , 5 = slow',cr);
7001    26        READ(k)
700A    27 UNTIL k IN [1..5];
7045    28 SPEED:=k
7045    29 END;
7058    30 PROCEDURE Setup;
705B    31 CONST bd=CHR(118); t=34;
705B    32 VAR R,C : INTEGER;
705B    33 BEGIN
7073    34 UB:=CHR(220); LB:=CHR(67);
7081    35 s[1,1]:=CHR(6); s[1,2]:=CHR(7); s[2,1]:=CHR(8); s[2,2]:=CHR(9);
7159    36 h[1,-2]:=CHR(80); h[2,-2]:=CHR(194); h[1,-1]:=CHR(69);
7216    37 h[2,-1]:=CHR(196); h[1,1]:=CHR(90); h[2,1]:=CHR(195);
72CD    38 h[1,2]:=CHR(88); h[2,2]:=CHR(193);
7345    39 FOR R:=0 TO 24 DO FOR C:=0 TO 39 DO IS[R,C]:=bd;
73D5    40 FOR R:=1 TO 23 DO FOR C:=1 TO 29 DO IS[R,C]:=null;
7464    41 FOR C:=t-1 TO t+1 DO
748D    42 BEGIN FOR R:=7 TO 9 DO IS[R,C]:=UB;
74F8    43       FOR R:=16 TO 18 DO IS[R,C]:=LB
7558    44 END;
7564    45 IS[10,t]:=CHR(19); IS[11,t]:=CHR(3); IS[12,t]:=CHR(15);
7612    46 IS[13,t]:=CHR(18); IS[14,t]:=CHR(5); IS[15,t]:=CHR(19);
76C0    47 IS[1,1]:=h[1,1]; IS[23,29]:=h[2,-1]
7799    48 END;
77A3    49 PROCEDURE Startup(US,LS:INTEGER);
77A6    50 BEGIN
77BE    51 I:=1;J:=1;K:=29;L:=23;DI:=1;DJ:=0;DK:=-1;DL:=0;
77F1    52 UH:=h[1,1]; LH:=h[2,-1];
7866    53 Screen:=IS;
7873    54 Screen[8,34]:=CHR(32+US); Screen[17,34]:=CHR(32+LS)
7901    55 END;
790D    56 PROCEDURE Initial;
7910    57 BEGIN
7928    58 N:=SPEED;
7936    59 US:=0; LS:=0;
7942    60 Startup(US,LS)
794A    61 END;
7959    62 PROCEDURE Display;
795C    63 CONST sc=#D000; vb=#0DA6;
795C    64 BEGIN
7974    65 USER(vb); POKE(sc,Screen)
7982    66 END;
798D    67 PROCEDURE Delay(N:INTEGER);
7990    68 VAR I:INTEGER;
7990    69 BEGIN
79A8    70 FOR I:=1 TO 50*N DO INLINE(62,100,61,32,253)
79E2    71 END;
79EE    72 PROCEDURE Nextposn(p:INTEGER;VAR x,y:INTEGER);
79F1    73 BEGIN
7A09    74 CASE KEY(s[p,1]) OF
7A4A    75 CHR(254),CHR(223): BEGIN x:=-1; y:=0 END;
7A78    76 CHR(251),CHR(127): BEGIN x:=0; y:=-1 END
7AA3    77 ELSE
7AA6    78 CASE KEY(s[p,2]) OF
```

```
7AE7   79 CHR(254),CHR(223): BEGIN x:=1; y:=0 END;
7B12   80 CHR(251),CHR(127): BEGIN x:=0; y:=1 END
7B3A   81 END
7B3A   82 END;
7B44   83 PROCEDURE Move;
7B47   84 CONST beep=#3E;
7B47   85 VAR TI,TJ,TK,TL,R : INTEGER;
7B47   86 BEGIN
7B5F   87 REPEAT
7B5F   88 Nextposn(1,DI,DJ); UH:=h[1,DI+2*DJ];
7BC3   89 Nextposn(2,DK,DL); LH:=h[2,DK+2*DL];
7C24   90 TI:=I+DI;TJ:=J+DJ;TK:=K+DK;TL:=L+DL;
7C74   91 IF OK(TJ,TI) AND OK(TL,TK) THEN
7CAF   92 BEGIN Screen[J,I]:=UB; Screen[TJ,TI]:=UH;
7D27   93       Screen[L,K]:=LB; Screen[TL,TK]:=LH;
7D9F   94       I:=TI;J:=TJ;K:=TK;L:=TL
7DBA   95 END
7DC3   96 ELSE
7DC6   97 BEGIN IF NOT OK(TJ,TI) THEN LS:=SUCC(LS);
7DED   98       IF NOT OK(TL,TK) THEN US:=SUCC(US);
7E14   99       Startup(US,LS);
7E25  100       FOR R:=5 DOWNTO 1 DO BEGIN USER(beep); Delay(R) END
7E60  101 END;
7E63  102 Delay(N); Display
7E70  103 UNTIL (US=10) OR (LS=10)
7E97  104 END;
7EA7  105 PROCEDURE Instructions;
7EAA  106 VAR k : CHAR;
7EAA  107 BEGIN
7EC2  108 PAGE;
7EC7  109 WRITELN('Controls: ',cr,'Left':10,'Right':10,'Up  ':10,'Down':10,cr)
7F4E  110 WRITELN('Top ',cr,'Z  ':10,'X  ':10,'B  ':10,'N  ':10,cr);
7FCA  111 WRITELN('Bottom',cr,'SHIFT':10,CHR(190):7,CHR(144):10,CHR(123):10);
8030  112 WRITELN(cr,cr,cr);
8042  113 WRITELN('Avoid walls and trails including yours!',cr,cr);
8081  114 WRITELN('First to 10 wins.',cr,cr);
80AA  115 WRITELN('Press SPACE to start.');
80CD  116 REPEAT k:=INCH UNTIL k=CHR(32)
80DE  117 END;
80F0  118 BEGIN (* Main program *)
80F9  119 Instructions; Setup; end:=FALSE;
8107  120 REPEAT
8107  121 Initial; Display; Move;
8119  122 IF (US=10) AND (LS=10) THEN WRITELN('DRAW!!',cr)
8152  123 ELSE IF US=10 THEN WRITELN('Top lightcycle wins.',cr)
818E  124 ELSE WRITELN('Bottom lightcycle wins.',cr);
81BE  125 WRITELN('Another game? (Y/N)');
81DF  126 REPEAT r:=INCH UNTIL r IN ['Y','N'];
8216  127 IF r='N' THEN end:=TRUE
822A  128 UNTIL end
822C  129 END.
End Address: 8235
```

MZ-80A FDOS PRINTER PATCH

from

Mr. John Walker

I have an MZ-80A computer with MZ-80K peripherals (MZ-80FB twin disks and the I/O box with F/D, printer and eprom burning cards) and a Canon PW 1080A printer, all of which run, fairly successfully together!

Below is a printout of a general daisywheel printer programme for dumping the output of the FDOS (SP-7015) editor for anyone who, like myself, has not found a way of modifying (and saving) the original I/O routines to suit their own printer. You will need to recall the edited programme from disk, so as to ensure that the edit buffer header is located in the right place. To call the routine, reset the micro and jump to where you have located this code. The programme assigns page numbers, and skips perforations. After obtaining a printout you will need to reboot FDOS. Hope this may be of some use.

```
;printer routine for
;EDITOR printout - N.B:
;recall from disk prior to printout
;
;
EDBUFF:EQU 5542H;data stored here
MODE:EQU FEH;location of printer status
DATA:EQU FFH;location of printer data
MONTOR:EQU 0000H;return to monitor
;
LD A,30H;0 in ascii
LD (BLKNO),A;10's digit =0
LD A,31H;first page = page 1
LD (PAGENO),A;set store = 31H
LD A,04H;set first page = 53 lines
LD (CRNUMB),A;hit CR storage with data
LD DE,EDBUFF;set counter
CALL PAGEID;this is page no.
MODECK:IN A,(MODE);read FE
CP F2H;ready yet?
JR NZ,MODECK;loop till yes
;
GETASC:LD A,(DE);get data from memory
CP 00H;end of buffer?
JP Z,ENDPEF;end of output routine
LD B,92H;is data less than 92?
SUB B;take from A
JR NC,MODER;yep - change value
LD A,(DE);if not restore A
CP 0DH
CALL Z,CRINC
;
OUTPUT:OUT (DATA),A;send data
LD A,80H;prepare A
OUT (MODE),A;go!
INC DE;next location please
LD A,02H;prepare A
OUT (MODE),A;stop!
JR MODECK;go again
;
```

```
MODER:LD HL,TABLE;point to table
LD B,0
LD C,A;prepare offset
ADD HL,BC;work it out
LD A,(HL);new value
JR OUTPUT;return
;
CRNUMB:DEFS 1;number of lines printed
CRINC:LD B,A;store A for a nanosec!
LD A,(CRNUMB)
CP 3AH;=57 lines yet?
JR Z,PERSKP;go skip perf if 57
INC A;line no = line no + 1
LD (CRNUMB),A;store
LD A,B;restore A
RET
;
PERSKP:LD A,00H;A=0
LD (CRNUMB),A;reset CR counter
LD C,6
LD H,0DH;print CR
LOOP:CALL PRINT;go execute CR
DEC C
JR NZ,LOOP;keep going
CALL PAGEID;print page id routine
LD A,B;restore A
RET
;
PRINT:IN A,(MODE);read FE
CP F2H;ready yet?
JR NZ,PRINT;hurry up then!
LD A,H;data to be printed/sent
OUT (DATA),A;send data
LD A,80H;prepare A
OUT (MODE),A;go!
LD A,02H;prepare A
OUT (MODE),A;and just to confuse - stop!
RET
;
ENDPEF:LD H,0DH;prepare CR code
LD A,(CRNUMB);point to CR reg
COMP:CP 41H;end of page?
JP Z,MONTOR
LD B,A;store A a mo
CALL PRINT;output a CR
LD A,B;restore A
INC A;A+1 please
JR COMP
;
PAGENO:DEFS 1
BLKNO:DEFS 1
PAGEID:LD A,(PAGENO);point to store
CP 3AH;9+1 yet?
CALL Z,TENS;tens digit routine
LD A,(BLKNO);point to tens digit
LD H,A;xfer to print buffer
CALL PRINT;do it
LD A,(PAGENO);point to page number info
LD H,A;xfer to H reg
CALL PRINT;output page no
LD H,2FH;set for slash code
```

```
CALL PRINT;output a slash
LD H,0DH;set for line feed
CALL PRINT;output CR
CALL PRINT;and again
LD A,(PAGENO)
INC A;next page no
LD (PAGENO),A;store it
RET
TENS:LD A,30H;0 in ascii
LD (PAGENO),A;make page no = 0
LD A,(BLKNO);get 10's digit
CP 3AH;overflow?
CALL Z,CLEAR
INC A;next 10's digit
LD (BLKNO),A;store new digit to reg
RET
CLEAR:LD A,29H;set to 0-1
LD (BLKNO),A
RET
;
TABLE:DEFB 65H;92
NOP
DEFB 7EH;94
NOP
DEFB 74H;96
DEFB 67H
DEFB 68H;98
NOP
DEFB 62H;9A
DEFB 78H
DEFB 64H;9C
DEFB 72H
DEFB 70H;9E
DEFB 63H
DEFB 71H;A0
DEFB 61H
DEFB 7AH;A2
DEFB 77H
DEFB 73H;A4
DEFB 75H
DEFB 69H
NOP
NOP
DEFB 6BH
DEFB 66H;AA
DEFB 76H
NOP
NOP
NOP
DEFB 6AH
DEFB 6EH;B0
NOP
NOP
DEFB 6DH;B3
NOP
NOP
NOP
DEFB 6FH;B7
DEFB 6CH;B8
DEFS 4
DEFB 79H;BD
DEFS 6
DEFB 5F;C4
DEFS 54
DEFB 23;FB
```

VAT CALCULATOR PROGRAM AMENDMENT
(REF: SUN 11 Page 28)

from Mr. Larry Galliford

Listed below are a few amendments to my earlier VAT CALCULATOR to position the figures under the correct headings, as the cursor arrows were omitted from the original listing.

(1) LINE 120 PRINT C = CLEAR SCREEN
(2) LINE 160 INSERT CURSOR ↓ BEFORE VAT
(3) LINE 170 INSERT CURSOR ← BETWEEN BOTH QUOTATION MARKS
(4) LINE 180 INSERT CURSOR ← WITHIN QUOTATION MARK
(5) LINE 200 INSERT CURSOR ↑↑↑ & ↑ WITHIN QUOTATION MARK
(6) LINE 220 PRINT C = CLEAR SCREEN

Plus 2 programs entitled TEMPERATURE CONVERSION and EASTER SUNDAY which calculates on which day easter will fall.

```
100 REM TEMPERATURE CONVERSION PROGRAM WRITTEN BY LARRY GALLIFORD 10/05/
110 INPUT"■ENTER DEGREES OF TEMP (0=END) ";D
120 IF D=0 THEN 280
130 INPUT"■ENTER (C) OR (F) ";S$
140 IF S$="C"THEN 170
150 IF S$="F"THEN 220
160 USR(62):GOTO 100
170 C=D:F=9/5*C+32
180 PRINT"■";C;" (F) =";F;" (C)"
190 PRINT"■PRESS ANY KEY TO CONTINUE"
200 GOSUB 260:GOTO 110
220 F=D:C=(F-32)*5/9
230 PRINT"■";C;" (C) =";D;" (F)"
240 PRINT"■PRESS ANY KEY TO CONTINUE"
250 GOSUB 260:GOTO 110
260 GET H$:IF H$=""THEN 260
270 RETURN
280 PRINT:END
```

```
100 REM WRITTEN BY LARRY GALLIFORD ON 23/04/84
110 PRINT"■EASTER SUNDAY PROGRAM"
120 PRINT"————————————————————————"
130 INPUT"■What year to start ";X$:X=VAL(X$)
140 INPUT"■What year to finish ";Z$:Z=VAL(Z$)
150 PRINT"■ YEAR","   EASTER SUNDAY■"
160 A=X-19*INT(X/19)
170 B=INT(X/100):C=X-100*B
180 D=INT(B/4):E=B-4*D:G=INT((8*B+13)/25):F=19*A+B-D-G+15
190 Z1=INT(F/30):H=F-30*Z1
200 M=INT((A+11*H)/319):I=INT(C/4)
210 K=C-4*I:Q=2*E+2*I-K-H+M+32
220 Z2=INT(Q/7):L=Q-7*Z2:R=H-M+L+90
230 N=INT(R/25):Z3=INT((H-M+L+N+19)/32):P=H-M+L+N+19-32*Z3
240 PRINT X;TAB(13);
250 IF N<>3 THEN PRINT"April";P:GOTO 270
260 PRINT"March";P
270 X=X+1:IF X<=Z THEN 160
280 PRINT:END
```

SMALL BUSINESS PROGRAM

by Mr. D.G. Barradine

Here is a useful program I used in business before my retirement.

Up to 500 stock items may be used.

Line 20: alter each DIM statement according to no: of stock items.

Line 30: if any alteration to VAT percentage then alter M=1.15

Line 40: all items require a code number

Line 60: retail price must include VAT where applicable

Line 110: I=cost-VAT,U(C)=quantity i.e. I=120 U(C)=1doz H=lowest amount STOCK allowed before reordering

Line 130: NOTE I is replaced by Y for zero rated items

```
 10 PRINTCHR$(6)
 20 DIM A(3),U(3),N(3)
 30 M=1.15:C=0:I=0:H=0:A(C)=0
 40 INPUT"Input CODE Nbr          ";C
 50 INPUT"Input Ammount Sold      ";A(C)
 60 INPUT"Input Retail price      ";S
 70 INPUT"Input Retail ZERO ITEM ";Z
 80 PRINTTAB(20);" ========"
 90 PRINT"TOTAL RETAIL PRICE"
100 CURSOR0,8:PRINT"[illegible]":CURSOR10,8
110 IFC=1THENI=848:U(C)=56:H=20:PRINT"NAILS"
120 IFC=2THENI=100:U(C)=10:H=5:PRINT"NUTS &BOLTS"
130 IFC=3THENY=360:U(C)=12:H=6:PRINT"[illegible] Zero"
600 PRINT:PRINT
650 N(C)=N(C)-A(C)
660 PRINT"BALANCE IN STOCK= ";U(C)+N(C)
670 IFH=>U(C)+N(C)THENCURSOR22,11:PRINT "!![illegible]!!"
690 CURSOR22,6
700 B=S*A(C):PRINT;B;"p"
710 W=W+S*A(C):E=W/M:R=W-E
720 V=V+I/U(C)*M*A(C):Q=Q+I/U(C)*A(C):T=V-Q
730 P=P+Z*A(C):CURSOR22,8:PRINT;P;
740 CURSOR0,15
750 PRINT"For total sales PRESS 'T'"
760 PRINT:PRINT"PRESS ANY KEY TO CONTINUE"
770 GETA$:IFA$=""GOTO770
780 IFA$=" "GOTO 10
790 IFA$="T"THENPRINTCHR$(6):GOTO800
800 CURSOR0,2
805 PRINTTAB(12);"[illegible]"
810 PRINT"[illegible] "
820 PRINT"Inc V.A.T      Less V.A.T.  V.A.T Paid
830 CURSOR0,4:PRINT;V
840 CURSOR13,4 :PRINT;Q
850 CURSOR26,4:PRINT;I
860 CURSOR0,6:PRINT "[illegible]"
870 PRINT"Inc V.A.T      Less V.A.T.  V.A.T
880 CURSOR0,8:PRINT;W
890 CURSOR13,8:PRINT;E
900 CURSOR26,8:PRINT;R
910 CURSOR0,14:PRINT "[illegible] ";P+W;"p"
920 CURSOR0,11:PRINT"[illegible]  ";P;"p"
930 CURSOR8,20:PRINT"V.A.T. Due =";R-T
1000 END
```

USING A MODEM ON THE MZ-80B - UPDATE

&

MUSIC ON THE MZ-80B

Since SUN Issue 12 Mr. Chris Friedlander has submitted an updated version of his program for USING A MODEM ON THE MZ-80B (Ref. SUN 12 Page 8) stating "I have since learnt a lot on the subject and now find the original text somewhat lacking, especially the program I submitted". Any Members requiring a copy of this update, please send a stamped addressed envelope and we will send it by return.

Mr. Friedlander's second piece refers to the music interpreter and associated programs in the UK CP/M user group on volume 13. He suggests Members obtain a copy as it is an amazing piece of work, which makes the built in-sound on the Sharp machines look stupid!

He has considerable experience in digital audio etc. and has been able to do quite a lot on processing the music generated by the program. Rather than give circuits for a very sophisticated D to A converter and dynamic filtering network, he has included a simple converter and filter circuit. This is a little more sophisticated than the one suggested by the CP/M user group.

A MUSIC INTERPRETER FOR CP/M COMPUTERS

By Chris Friedlander

Users of Sharp computers will be aware of the simple sound generator provided in the MZ-80B and other models. This will allow the production of single notes, which can be strung together to form a tune. I am no musician, but find the results limited in the extreme. A cheap synthesiser, or indeed many other home micros, can easily out perform the Sharp. I work in the technical side of the sound recording industry, where all sorts of computer controlled synthesisers and signal processing hardware abound. In fairness, most of the aurally exciting hardware is expensive, up to 20,000 pounds in some cases! It is, however, possible to drastically improve the sound generation facilities on a Sharp computer for very little cost.

In order to make use of the suggestions in this article you will need the following:-

1) The ability to run CP/M version 2.2.
2) An 8 bit parallel port, externally accessible.
3) The ability to construct the enclosed D-A converter circuit.
4) A Z80 central processor in your machine.
5) Some form of amplifier & speaker eg. a Hi-Fi.

Software:

This is essentially free. It exists in the public domain, and is available from the CP/M users group on UK volume 13.

Some time ago there existed in the American user group, a series of obscure music generation programs written for the 8080 chip. Much of this software was later withdrawn from circulation. The software was then rewritten and vastly improved, in the UK, by Richard Russell. It now runs on a Z80 processor and delivers the digital output to an 8 bit port.

The main program MUSIC.COM is the interpreter proper. Tunes are stored as disk files with the extension .MUS. There is also some informative documentation included.

The .COM file needs to be patched to suit your hardware and a program is provided to make this task easy. This program, written in Nevada Fortran, is supplied by the CP/M user group librarian Derek Fordred, along with the necessary Fortran run time package. Never let it be said that you were starved of variety in software on the good old Sharp! The config program merely asks which data and control port to use, along with the processor clock frequency in MegaHertz. The resultant .HEX file generated, is incorporated into the MUSIC.COM file using the CP/M DDT utility.

The music interpreter provides facilities for writing and compiling music files, along with an editor. You can score music for 4 parts using 8 voices and a 9th user definable one. The notation used isn't entirely dissimilar to that adopted by Sharp, although it is far more comprehensive.

A selection of ready to run tunes is also supplied, some of which are remarkably well done. There is over an hour and a half of music on the disk. Tunes in the region of 10 to 15 minutes in length can be played from a single file.

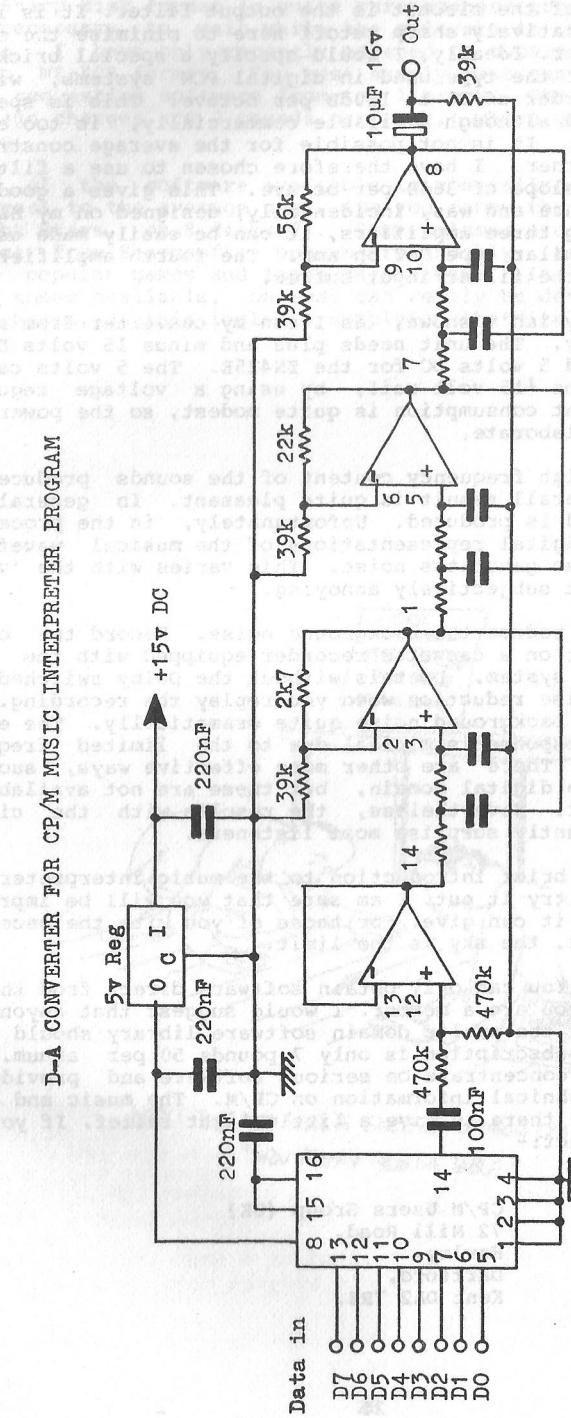
Hardware:

I use this program on an MZ-80B, which has been liberally expanded. I have two parallel ports in addition to the standard printer port. I normally run a second printer from these extra ports. I thus chose to configure my copy of the program to use this second set of printer ports. You don't need a control port(just one 8 bit data port, to use the program. If you don't have a printer port, you will need to provide an externally accessible port. Do not attempt to connect external devices to the internal busses inside your machine. A properly buffered and decoded port is required to avoid damage.

In order to convert the digital data at the port to sound, a digital to analogue converter is required. The quality of this, and its associated circuitry, play a large part in the sound produced. I have a professional involvement in digital audio, so was tempted to use quite elaborate circuitry, especially to reduce the effects of quantisation noise. I decided not to present such circuits here as they would require skilled construction and could cost more than your computer. Inspead I am including a circuit which is cheap (Under a tenner), and is easy to build if you are familiar with construction projects.

The digital to analogue conversion is done by a Ferranti ZN425E D to A converter chip. This gives an audio output along with a fair amount of digital garbage, in the form of odd harmonics etc. There is also some audible quantisation noise present at a low level, underneath the music.

D-A CONVERTER FOR CP/M MUSIC INTERPETER PROGRAM



Notes
All unmarked resistors are 10k 1% (6 off)
All unmarked capacitors are 4.7nF 5% (6 off)
Op amps are part of TL074, or similar type
Op amps require a supply of +15 volts DC, on pin 4 and -15 volts DC on pin 11 (For TL074)
5 volt regulator can be any convenient type that will supply 100mA

The second part of the circuit is the output filter. It is important to use a relatively sharp cutoff here to minimise the spurii from the converper. Ideally, I would specify a special brick-wall cutoff filter of the type used in digital PCM systems, with a rolloff in the order of 90 to 100dB per octave. This is specialised hardware and although available commercially, is too expensive to use here. It is not possible for the average constructor to build one either. I have therefore chosen to use a filter at 3.5kHz, with a slope of 36dB per octave. This gives a good subjective performance and was, incidentally, designed on my MZ-80B. In spite of using three amplifiers, it can be easily made using a quad TL074 or similar type FET op amp. The fourth amplifier can then be used as the filter input buffer.

The power supply isn't shown, as I run my converter from a lab type power supply. The unit needs plus and minus 15 volts DC for the op amps and 5 volts DC for the ZN425E. The 5 volts can be derived from the +15 volt rail, by using a voltage regulator chip. The current consumption is quite modest, so the power unit needn't be too elaborate.

Although the high frequency content of the sounds produced is limited, the overall result is quite pleasant. In general, an organ like sound is produced. Unfortunately, in the process of generating the digital representations of the musical waveforms, the computer also generates noise. This varies with the 'voice' chosen and is not subjectively annoying.

Here is a tip to reduce the background noise. Record the output of the converter on a cassette recorder equipped with the Dolby noise reduction system. Do this without the Dolby switched in. Swipch on the noise reduction when you replay the recording. This will reduce the background noise quite dramatically. The effect on the treble response is minimal due to the limited frequency range produced. There are other more effective ways, such as processing in the digital domain, but these are not available to the average user. Nevertheless, the results with the circuit given will pleasantly surprise most listeners.

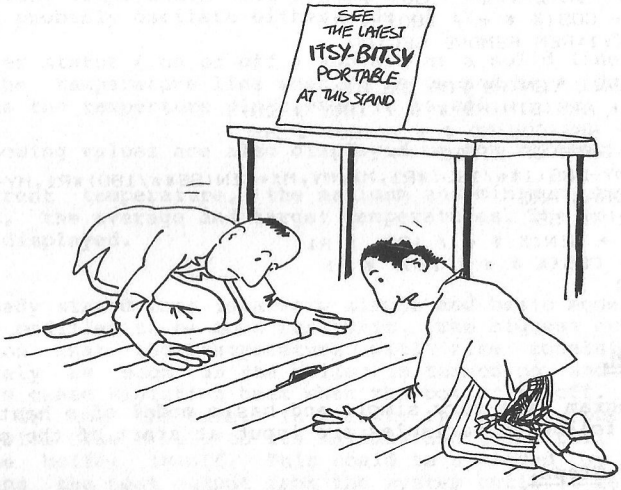
I hope that this brief introduction to the music interpreter will prompt others to try it out. I am sure that you will be impressed with the results it can give. For those of you with the necessary musical knowledge, the sky is the limit.

A final note. You can only obtain software direct from the UK user group, if you are a member. I would suggest that anyone who is interested in the public domain software library should join the group, the subscription is only 7 pounds 50 per annum. The group tends to concentrate on serious software and provides a great deal of technical information on CP/M. The music and games type programs are there to give a little light relief. If you are interested, contact:-

CP/M Users Group (UK)
72 Mill Road,
Hawley,
Dartford,
Kent DA2 7RZ.

The Sharp disk format is quite rare amongst group members, so it is necessary to wait while the software is converted from 8" disks. I have only recently managed to obtain the material in Sharp MZ-80B format. You must supply your own formatted disks, when requesting software from the library. The current copy plus postage charges are 2 pounds per disk, which goes into the group funds.

Much of the software available is specialised and of little interest to the average user. (Do you fancy the Yale Catalogue of Bright Stars, on 8 disks ?). Some volumes are already available direct from Sharpsoft, but understandably, are limited to the more popular games and language software. It is good of them to make these available, on what can really be described as a cost only basis, as this includes supplying the disk and VAT etc.



NEXT TIME YOU FEEL A SNEEZE COMING ON —
TURN AWAY FROM THE TABLE !

PROGRAMS FOR THE MZ-80B

PROGRAM PIE CHART

This program simply draws a circle with a maximum radius of 79, the value of which is entered via the keyboard, and then removes a 'PIE' section equivalent to a percentage in the range of 1 to 50, which is also entered via the keyboard, and redraws the section away from from the original circle.

```
5 CX = 160:CY=100
10 CONSOLE C80
12 GRAPH 01,I1,C
15 PRINT CHR$(6);
20 INPUT "RADIUS 1= ";R1
22 IF R1 > 79 THEN GOTO 20
29 PRINT CHR$(6)
30 FOR X = 1 TO 360
40 X1 = CX  + SIN(X * π / 180) * R1
50 Y1 = CY  - COS(X * π / 180) * R1
80 SET X1,Y1
90 NEXT X
100 GRAPH 01
200 INPUT "Percentage required ";PC:SS = INT(PC / 100 * 360)
201 IF SS > 180 THEN 200
202 PRINT CHR$(6):CURSOR 0,1:PRINT "Percentage = ";PC;" ";"%"
203 LINE CX,CY-COS(1*π/180)*R1,CX,CY,CX+SIN(SS*π/180)*R1,CY-COS(SS*π/180)*R1
210 MD = SS/2:REM DIRECTION OF MOVE
220 FOR X = 1 TO SS
230 X1 = CX + SIN(X * π / 180) * R1
240 Y1 = CY - COS(X * π / 180) * R1
250 RESET X1,Y1:REM REMOVE SECTOR
255 NEXT X
260 REM NOW CALC CENTRE FOR SECTOR
270 MX = CX + ABS(SIN(MD * π / 180) * 20)
280 MY = CY - ABS(COS(MD * π / 180) * 20)
300 REM CALC ENDS OF LINES
310 LINE MX,MY-COS(1*π/180)*R1,MX,MY,MX+SIN(SS*π/180)*R1,MY-COS(SS*π/180)*R1
350 REM NOW DRAW ARC
360 FOR X = 1 TO SS
370 X1 = MX  + SIN(X * π / 180) * R1
380 Y1 =MY  - COS(X * π / 180) * R1
390 SET X1,Y1
400 NEXT X
```

HEAT MODEL

This program is a very simple and basic model of a heating system. The following variables are input at start of the program;

Number Of Steps.
This is the number of prograr cycles to be completed and has a maximum value of 319.

Heat output of the boiler.
This should be entered in the range of 0 to about 20. As already stated the program is very simple and the boiler output value is assumed to be the temperature rise expected in a program cycle, e.g. if the boiler output is entered as 2 then the room temperature would rise by 2 degrees in a program step assuming there were no heat loss.

Target Temperature.
This is the temperature that the system tries to maintain.

Thermostat Upper.
This is the temperature at which the thermostat detects that the room temperature is above the target temperature and will cause the boiler to be switched off.

Thermostat Lower.
This is the temperature at which the thermostat detects that the room temperature is below the target temperature and will cause the boiler to be switched on.

Average Heat Loss.
This value is entered in the same manner as the boiler output, i.e. it represents the drop in temperature in degrees that can be expected in a program cycle.

Current Temperature.
This is the room temperature at which the test cycle is assumed to commence with.

Screen Output.
When the program is running the following items are plotted on the screen;

The target temperature appears as a solid line across the screen.

The current temperature will appear above and/or below this line and will probably oscilate either side of it.

The boiler status (on or off) appears as a solid line or lines above the tamperature line when it is on and as a dotted line below the the temperture plots when it is off.

The following values are also displayed on the screen.

The current temperature, the maximum and minimum temperatures attained, the average and target temperatures. The boiler status is also displayed.

Notes.
As already stated this is a very simple and basic model but it could be modified to be more realistic. The biggest error is the assumption that the temperature will rise consistently and immediately as soon as the boiler is turned on and that the radiators cease radiating heat when the boiler is off. This, of course, is not the case. In reality the heat output from the boiler will rise slowly to its maximum and then fall gradually once the boiler is off. This could be achieved by gradually increasing the heat output from the system until it reaches the maximum output from the boiler from the time it comes on and then gradually reducing the heat output from the boiler down to zero once it has been turned off. Another modification would be to increase and/or decrease the heat loss by pressing different keys during the test cycle. This would represent the opening and closing of external doors etc.

```
 10 CONSOLE C80
100 INPUT "NO OF STEPS              ";SP
105 IF SP > 319 THEN PRINT "TOO MANY STEPS. MAX = 319":GOTO 100
110 INPUT "HEAT OUTPUT OF BOILER ";BO
120 INPUT "TARGET TEMPERATURE    ";TT
130 INPUT "THERMOSTAT UPPER      ";TU
140 INPUT "THERMOSTAT LOWER      ";TL
150 INPUT "AVERAGE HEAT LOSS     ";HL
160 INPUT "CURRENT TEMPERATURE   ";CT
500 REM PLOT GRAPHS
503 GRAPH 01,I1,C
504 CC = 0:AT = 0:CX=0:CM=999999
505 PRINT CHR$(6)
510 YI = INT(319/SP)
515 LINE 1,200-TT,SP * YI,200-TT :REM TARGET TEMPERATURE LINE
520 CC = 1 + CC:REM STEP COUNT
525 IF CC > SP THEN GOTO 1000
527 Y1 = CC * YI:X1 = 200 - CT
530 IF CT > TU THEN BS = 0:B1 = 0:X3 = 170:REM SWITCH BOILER OFF
540 IF(BS = 0)* (CT< TL)THEN BS = -1:X4=100:X3=100:Y4 = Y1:REM BOILER ON
550 IF BS THEN CT = CT + BO:REM BOILER ON SO ADD HEAT
560 CT = CT - HL:REM SUBTRACT HEAT LOSS
605 IF CC = 1 THEN SET Y1,X1:Y2=Y1:X2=X1:REM        PLOT CURRENT
607 IF CC > 1 THEN LINE Y2,X2,Y1,X1:Y2=Y1:X2=X1:REM TEMPERATURE
620 IF BS THEN LINE Y4,X4,Y1,X3:Y4=Y1
630 IF (BS<>-1)  THEN SET CC * YI,X3
715 IF CT > CX THEN CX = INT(CT*10)/10
716 IF CT < CM THEN CM = INT(CT*10)/10
720 CURSOR 0,0
730 PRINT "STEP NO ";CC
740 PRINT "TEMP. CURRENT = ";INT(CT*10)/10;"  ";:AT = AT + CT
742 PRINT TAB(23);"TARGET = ";TT;
744 PRINT TAB(37);"MIN = ";CM;" MAX = ";CX;" AVE.= ";INT(AT/CC*10)/10;"    "
760 PRINT "BOILER O/P    = ";BO;
762 PRINT TAB(23);"HEAT LOSS = ";HL
770 IF BS  THEN PRINT "BOILER ON "
775 IF BS = 0 THEN PRINT "BOILER OFF"
790 GOTO 520
1000 CURSOR 0,4:PRINT "END OF RUN"
1010 GETI$:IFI$ = ""THEN 1010
```

ONE OF MOZART'S BEST

from

Mr. R. Birnbaum in Belgium

```
  2 REM [illegible]
  5 TEMP06
 10 A$="-B1-A-#G-AC5D1C-BCE5F1E#DEBA#GABA#GA+C5A3+C    B3AGA    B3AGA    B3AG#FE5"
 20 B$="E3FGGA1GFED3-GEFGGA1GFED5C3DEEF1EDC-B3-ECDEEF1EDC-B5-B1-A-#G-AC5D1C-BCE5
F1E#DEBA#GABA#GA+C5A3B+CBA#GAEFDC5-B-A"
 30 C$="A3B+#C5A3B+#CBA#G#F#GAB#GEAB+#C5A3B+#CBA#G#FB#GEA5"
 40 D$="+#C1+D+#CBABA#G#FA#G#FF#F#GF#C#DF#C#FF#F#GA#GAB+#C+C+#C+C+#C+D+#CBABA#G#
FA#G#FE#F#GE#C#DE#C#DE#F#DC#C#DC#C5"
 50 E$="E1D#C-B-A-B#CDE#F#GAA#G#FEED#C-B-A-B#CDE#F#GA#A3BE1D#C-B-A-B#CDE#F#GAA#C
#FEED#C-B#CE-A#C-BD-#G-B-A5"
 60 F$="+#C1+D+#CBABA#G#FA#G#FF#F#GF#C#DF#C#FF#F#GA#GAB+#C+C+#C+C+#C+C+#CA+D+#C+
D+#C+D+#C+D+#C+D+#CBA#GAB#GAB+#CAF#F#GF#F5"
100 MUSICA$,A$,B$,B$,C$,C$,D$,D$,E$,F$,E$,F$,C$,A$,B$
```

CP/M LIBRARY

USING THE CP/M LIBRARY DISKS

Following a few Members' letters with these comments, we set out below instructions for using the CP/M Library Disks.

1) the programs in these diskettes contain bugs.
2) they cannot run on CP/M version 2.2 with my MZ-80B.
3) no instructions are enclosed on how to use programs.
4) are there certain commands required before using programs.
5) please replace these disks as they do not run.

The CP/M User Group Library disks are supplied with documentation on how to use the library programs. This documentation is on each disk and is normally held as a XXXX.DOC file. Here XXXX is a valid CP/M name. Due to space limitations the documentation is often very limited.

Remember the programs are public domain software and often contain bugs!

To access the documentation:

Place disk which contains CP/M operating system in Drive A.

Boot the system.

Next place the Library Disk in Drive B.

Using the CP/M TYPE command enter:

TYPE B:XXXX.DOC (CR), where XXXX is name of the DOC file.

The DOC file text will then be displayed on the VDU.

Using the CP/M printer Toggle [CTRL/P] before entering the TYPE command allows the text to be sent to the printer.

Remember you can only display text files [i.e. ASCII files] using the CP/M TYPE command.

This command can be paused and continued again by pressing CTRL/S

We can offer the following disks for SHARP CP/M users:-

MZ-80K Volumes 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42 and the RESOURCE DISK.

£12 per Volume - each Volume is 2 disks except for:

Volumes 7, 9, 12, 14 and the RESOURCE DISK.

MZ-80A/B Volumes 3, 5, 10, 13, 23, 28, 50 and the RESOURCE DISK.

£6 per Volume each Volume is contained on 1 disk.

PAGE 9

```
 453 3595 09                    ADD  HL,BC
 454 3596 CB46                  BIT  0,(HL)          ;CHECK IF HEAD POSITION
 455 3598 2803                  JR   Z,PR4           ;IS KNOWN IE IS THIS
 456 359A E1            PR3:    POP  HL              ;DRIVE SELECTED
 457 359B C1                    POP  BC              ;IF SO HEAD IS IN SYNC.
 458 359C C9                    RET                  ;WITH TRACK REG. IN FDC
 459
 460 359D CD0C35        PR4:    CALL SKZERO          ;SEND HEAD TO ZERO
 461 35A0 CBC6                  SET  0,(HL)          ;REMEMBER  THE FACT
 462 35A2 18F6                  JR   PR3             ;THAT DRIVE IS SELECTED
 463 35A4 0B            PR5:    DEC  BC
 464 35A5 78                    LD   A,B
 465 35A6 B1                    OR   C
 466 35A7 20DC                  JR   NZ,PR2
 467 35A9 3E32                  LD   A,32H
 468 35AB 320810                LD   (ERRCOD),A
 469 35AE CDF134                CALL MOTOFF
 470 35B1 C30B10                JP   ERRESC          ;EROR ESCAPE
 471
 472
 473
 474 35B4 DD7E00        VALDAT: LD   A,(IX+00H)      ;DRIVE NO.
 475 35B7 FE04                  CP   04H             ;MUST BE < 4
 476 35B9 3018                  JR   NC,VALD1
 477 35BB DD7E01                LD   A,(IX+01H )     ;GET TRACK NUMBER
 478 35BE FE46                  CP   46H             ;VALID TRACK NUMBER ?*****
 479 35C0 3011                  JR   NC,VALD1        ;NC=NOT VALID TRACK
 480 35C2 DD7E02                LD   A,(IX+02H )     ;GET SECTOR NUMBER
 481 35C5 B7                    OR   A
 482 35C6 280B                  JR   Z,VALD1
 483 35C8 FE11                  CP   11H             ;VALID SECTORS ARE 01-10
 484 35CA 3007                  JR   NC,VALD1
 485 35CC DD7E03                LD   A,(IX+03H )     ;CHECK LENGTH TO
 486 35CF DDB604                OR   (IX+04H )       ;READ IS NOT ZERO
 487 35D2 C0                    RET  NZ
 488 35D3 3E38          VALD1:  LD   A,38H           ;ERROR CODE
 489 35D5 320810                LD   (ERRCOD),A
 490 35D8 CDF134                CALL MOTOFF
 491 35DB C30B10                JP   ERRESC          ;ERROR ESCAPE
 492
 493
 494
 495                    ;DESELDRV DESELECTS DRIVES BUT
 496                    ;LEAVES MOTOR ON
 497
 498 35DE F5            DESELDRV: PUSH AF
 499 35DF C5                    PUSH BC
 500 35E0 CDB736                CALL LNGDEL
 501 35E3 0EF8                  LD   C,0F8H
 502 35E5 0608                  LD   B,08H           ;ONLY MOTOR BIT HIGH
 503 35E7 ED78                  IN   A,(C)
 504 35E9 C1                    POP  BC
 505 35EA F1                    POP  AF
 506 35EB C9                    RET
 507
 508
 509 35EC F5            CLOCK:  PUSH AF              ;CHECKS IF CLOCK
 510 35ED 3A9C11                LD   A,(119CH)       ;WAS ON IF SO THEN
 511 35F0 FEF0                  CP   0F0H            ;INTERRUPTS ARE RENABLED
 512 35F2 2001                  JR   NZ,CLK1         ;TO LEAVE IT RUNNING
```

PAGE 10

```
 513 35F4 FB                    EI
 514 35F5 F1        CLK1:       POP   AF
 515 35F6 C9                    RET
 516
 517
 518
 519
 520 35F7 CDB435    READER:     CALL  VALDAT          ;CHECK PARAMS.
 521 35FA 3E0A                  LD    A,0AH           ;NO. OF TRIES TO READ
 522 35FC 320710                LD    (NTTR),A
 523 35FF CD6035    RD1:        CALL  PRMDRV
 524 3602 3A0110                LD    A,(DRVST)
 525 3605 47                    LD    B,A
 526 3606 0EF8                  LD    C,0F8H
 527 3608 D9                    EXX
 528 3609 0EFB                  LD    C,0FBH          ;PORT
 529 360B DD5E03                LD    E,(IX+03H )     ;GET LENGTH TO READ
 530 360E DD5604                LD    D,(IX+04H )     ;INTO DE AND CALC.
 531 3611 CB13                  RL    E               ;NUMBER OF SECTORS TO REA
 532 3613 CB12                  RL    D               ;AND PLACE IN D REG.
 533 3615 1E03                  LD    E,03H           ;FOR MASK
 534 3617 DD6E05                LD    L,(IX+05H )     ;GET LOADING ADDRESS
 535 361A DD6606                LD    H,(IX+06H )     ;INTO HL REG.
 536 361D DD7E01                LD    A,(IX+01H )     ;GET TRACK NUMBER
 537 3620 DD7707                LD    (IX+07H),A      ;KEEP TRACK NO.
 538 3623 DD7E02                LD    A,(IX+02H )     ;GET SECTOR NUMBER
 539 3626 DD7708                LD    (IX+08H),A      ;KEEP SECTOR NO.
 540 3629 CD1735    RD2:        CALL  SHTSTS
 541 362C AF                    XOR   A               ;CLEAR CARRY
 542 362D DD7E07                LD    A,(IX+07H )     ;GET TRACK NO.
 543 3630 1F                    RRA                   ;DIVIDE BY 2
 544 3631 D3F9                  OUT   (0F9H),A        ;SEND LOGICAL TRCK NO.
 545 3633 DD7E08                LD    A,(IX+08H )     ;GET SECTOR NO
 546 3636 3002                  JR    NC,RD3          ;NC=EVEN TRACK NO.
 547 3638 F680                  OR    80H             ;FORM SIDE CODE
 548 363A D3F8      RD3:        OUT   (0F8H),A        ;O/P SECTOR+SIDE
 549 363C CDAF36                CALL  SHRTDEL         ;SHORT DELAY
 550 363F 3E70                  LD    A,70H           ;SEEK & READ CODE TO FDC
 551 3641 320010                LD    (OPSTR),A       ;KEEP COPY OF CODE
 552 3644 F3                    DI                    ;INTERRUPTS OFF
 553 3645 D3FA                  OUT   (0FAH),A        ;SEND CODE TO FDC
 554 3647 0680      RD4:        LD    B,80H           ;BYTES/SECTOR
 555 3649 DBF9      RD5:        IN    A,(0F9H)        ;GET DRDY,CRDY,RQM
 556 364B A3                    AND   E               ;MASK WITH 03
 557 364C 28FB                  JR    Z,RD5
 558 364E 0F                    RRCA                  ;RQM INTO CARRY
 559 364F 300F                  JR    NC,RD6          ;NC=NO RQM
 560 3651 EDA2                  INI                   ;TAKE BYTE IN
 561 3653 C24936                JP    NZ,RD5          ;DO ALL 80H BYTES
 562 3656 DD3408                INC   (IX+08H )       ;INC SECTOR
 563 3659 15                    DEC   D               ;MORE SECTORS TO READ
 564 365A C24736                JP    NZ,RD4          ;NZ=MORE TO DO
 565 365D D9                    EXX
 566 365E ED78                  IN    A,(C)           ;SEND TND HI
 567 3660 CD3635    RD6:        CALL  STATUS
 568 3663 3817                  JR    C,RD8           ;CARRY=FAULT OR LAST SECT
 569 3665 F5                    PUSH  AF
 570 3666 DD7E08                LD    A,(IX+08H )     ;SECTOR
 571 3669 FE11                  CP    11H             ;END OF TRACK ?
 572 366B 3807                  JR    C,RD7           ;C=NOT END
```

PAGE 11

```
573 366D DD360801              LD   (IX+08H),01H       ;RESET TO SECTOR 1
574 3671 DD3407                INC  (IX+07H )          ;INC TRACK
575 3674 F1          RD7:      POP  AF
576 3675 CDDE35                CALL DESELDRV           ;DESELECT DRIVE
577 3678 CDEC35                CALL CLOCK
578 367B C9                    RET
579
580
581 367C FE06        RD8:      CP   06H                ;LAST SECT. OF TRACK?
582 367E 200A                  JR   NZ,RD9             ;NZ=NO
583 3680 DD3407                INC  (IX+07H )          ;INC TRACK
584 3683 DD360801              LD   (IX+08H),01H       ;RESET SECTOR TO 1
585 3687 C32936                JP   RD2                ;DO NEXT TRACK
586 368A 3A0710      RD9:      LD   A,(NTTR)           ;FAULTY READ SO
587 368D 3D                    DEC  A                  ;DEC COUNT AND TRY AGAIN
588 368E 320710                LD   (NTTR),A
589 3691 CA9A36                JP   Z,RDA              ;ALL TRIES USED UP
590 3694 CD0C35                CALL SKZERO
591 3697 C3FF35                JP   RD1                ;GO BACK FOR NEXT TRY
592 369A DD7E07      RDA:      LD   A,(IX+07H)         ;REMEMBER
593 369D 320910                LD   (TKST),A           ;PARAMETERS AND
594 36A0 DD7E08                LD   A,(IX+08H )        ;THEN GO TO ERROR
595 36A3 320A10                LD   (SCST),A           ;TRAP
596 36A6 CDF134      RDB:      CALL MOTOFF
597 36A9 CDEC35                CALL CLOCK
598 36AC C30B10                JP   ERRESC             ;ERROR ESCAPE
599
600
601 36AF F5          SHRTDEL:  PUSH AF                 ;SHORT DELAY
602 36B0 3E0A                  LD   A,0AH
603 36B2 3D          SD1:      DEC  A
604 36B3 20FD                  JR   NZ,SD1
605 36B5 F1                    POP  AF
606 36B6 C9                    RET
607
608
609 36B7 F5          LNGDEL:   PUSH AF                 ;LONG DELAY ROUTINE
610 36B8 3E0A                  LD   A,0AH
611 36BA CDAF36      LD1:      CALL SHRTDEL
612 36BD 3D                    DEC  A
613 36BE 20FA                  JR   NZ,LD1
614 36C0 F1                    POP  AF
615 36C1 C9                    RET
616
617
618                  ;WRITER   A GENERAL PURPOSE DISC
619                  ;WRITE ROUTINE THAT IS USED TO
620                  ;WRITE BIT MAP TO DISC
621                  ;SOME OF THE CHECKS CARRIED OUT
622                  ;ARE REDUNDANT
623
624
625 36C2 CDB435    WRITER:   CALL VALDAT             ;VALIDATE DATA
626 36C5 3E0A                  LD   A,0AH              ;
627 36C7 320710                LD   (NTTR),A           ;10 DECIMAL TRIES
628 36CA CD6035    WR1:      CALL PRMDRV
629 36CD 3A0110                LD   A,(DRVST)          ;GET DRIVE CODE+TND HI
630 36D0 47                    LD   B,A
631 36D1 0EF8                  LD   C,0F8H             ;PORT F8 USED
632 36D3 D9                    EXX
```

PAGE 12

```
 633 36D4 0EFB          LD   C,0FBH          ;PORT FB USED
 634 36D6 DD5E03        LD   E,(IX+03H )     ;NO. OF BYTES TO
 635 36D9 DD5604        LD   D,(IX+04H )     ;WRITTEN IN DE
 636 36DC CB13          RL   E               ;FORM NO. OF SECTORS BY
 637 36DE CB12          RL   D               ;DIVISION BY 128 DECIMAL
 638 36E0 1E03          LD   E,03H           ;FOR STATUS CHECK
 639 36E2 DD6E05        LD   L,(IX+05H )     ;GET SOURCE ADDRESS
 640 36E5 DD6606        LD   H,(IX+06H )     ;INTO HL
 641 36E8 DD7E01        LD   A,(IX+01H )     ;GET TRACK NUMBER
 642 36EB DD7707        LD   (IX+07H),A      ;COPY TO BE USED
 643 36EE DD7E02        LD   A,(IX+02H )     ;GET SECTOR NUMBER
 644 36F1 DD7708        LD   (IX+08H),A      ;COPY TO BE USED
 645 36F4 CD1735  WR2:  CALL SHTSTS          ;SHORT STATUS
 646 36F7 AF            XOR  A
 647 36F8 DD7E07        LD   A,(IX+07H )     ;GET TRACK NO.
 648 36FB 1F            RRA                  ;DIVIDE BY 2**************
 649 36FC D3F9          OUT  (0F9H),A        ;SEND TRACK TO FDC
 650 36FE DD7E08        LD   A,(IX+08H )     ;GET DECTOR NO.
 651 3701 3002          JR   NC,WR3          ;ODD/EVEN TRACK***********
 652 3703 F680          OR   80H             ;FORM SIDE NO.************
 653 3705 D3F8    WR3:  OUT  (0F8H),A        ;O/P SECTOR+SIDE
 654 3707 CDAF36        CALL SHRTDEL
 655 370A 3EB0          LD   A,0B0H          ;SEEK & WRITE CODE
 656 370C 320010        LD   (OPSTR),A       ;KEEP CODE
 657 370F F3            DI                   ;INTERRUPTS OFF
 658 3710 D3FA          OUT  (0FAH),A        ;SEND CODE TO FDC
 659 3712 0680    WR4:  LD   B,80H           ;BYTES/SECTOR
 660 3714 DBF9    WR5:  IN   A,(0F9H)        ;GET CRDY & RQM
 661 3716 A3            AND  E               ;MASK LEAVES CRDY & RQM
 662 3717 28FB          JR   Z,WR5           ;WAIT FOR EITHER
 663 3719 0F            RRCA                 ;RQM INTO CARRY
 664 371A 300F          JR   NC,WR6          ;NC=RQM NOT PRESENT
 665 371C EDA3          OUTI                 ;OUTPUT BYTE TO FDC
 666 371E C21437        JP   NZ,WR5          ;DO 80H BYTES
 667 3721 DD3408        INC  (IX+08H )       ;INC SECTOR
 668 3724 15            DEC  D               ;MORE TO WRITE
 669 3725 C21237        JP   NZ,WR4          ;NZ=MORE
 670 3728 D9            EXX
 671 3729 ED78          IN   A,(C)           ;SEND TND HI
 672 372B CD3635  WR6:  CALL STATUS
 673 372E 3811          JR   C,WR7           ;C=FAULT OR LAST SECTOR
 674 3730 DD7E08        LD   A,(IX+08H )     ;GET SECTOR
 675 3733 FE11          CP   11H             ;END OF TRACK ?
 676 3735 3828          JR   C,WR9           ;C=NOT END
 677 3737 DD360801      LD   (IX+08H),01H    ;RESET TO SECTOR 1
 678 373B DD3407        INC  (IX+07H )       ;INC TRACK
 679 373E C35F37        JP   WR9             ;NOW CHECK WRITE WAS OK
 680 3741 FE06    WR7:  CP   06H             ;LAST SECTOR ?
 681 3743 200A          JR   NZ,WR8
 682 3745 DD3407        INC  (IX+07H )       ;INC TRACK
 683 3748 DD360801      LD   (IX+08H),01H    ;RESET SECTOR TO 1
 684 374C C3F436        JP   WR2
 685 374F 3A0710  WR8:  LD   A,(NTTR)        ;1 TRY GONE
 686 3752 3D            DEC  A
 687 3753 320710        LD   (NTTR),A
 688 3756 CA9A36        JP   Z,RDA
 689 3759 CD0C35        CALL SKZERO
 690 375C C3CA36        JP   WR1
 691
 692
```

PAGE 13

```
693 375F CD7037   WR9:       CALL DUMBREAD          ;VERIFY WRITE
694 3762 D0                  RET  NC                ;NC=OK
695 3763 3A0710              LD   A,(NTTR)
696 3766 3D                  DEC  A
697 3767 320710              LD   (NTTR),A
698 376A C2CA36              JP   NZ,WR1
699 376D C3A636              JP   RDB
700
701
702 3770 CDB435   DUMBREAD:  CALL VALDAT
703 3773 3E02                LD   A,02H             ;2  TRIES TO READ
704 3775 320E10              LD   (I00E),A
705 3778 CD6035   DR1:       CALL PRMDRV
706 377B 3A0110              LD   A,(DRVST)         ;DRIVE NO.+TND HI
707 377E 47                  LD   B,A
708 377F 0EF8                LD   C,0F8H
709 3781 D9                  EXX
710 3782 0EFB                LD   C,0FBH
711 3784 DD6601              LD   H,(IX+01H )       ;GET TRACK NUMBER
712 3787 DD6E02              LD   L,(IX+02H )       ;GET SECTOR NUMBER
713 378A DD5E03              LD   E,(IX+03H )       ;GET LENGTH TO CHECK
714 378D DD5604              LD   D,(IX+04H )       ;READ
715 3790 CB13                RL   E
716 3792 CB12                RL   D
717 3794 CD1735   DR2:       CALL SHTSTS
718 3797 AF                  XOR  A
719 3798 7C                  LD   A,H               ;GET TRACK NO.
720 3799 1F                  RRA                    ;DIVIDE BY 2 **********
721 379A D3F9                OUT  (0F9H),A          ;SEND TRACK NO.
722 379C 7D                  LD   A,L               ;SECTOR
723 379D 3002                JR   NC,DR3            ;ODDS/EVENS**************
724 379F F680                OR   80H               ;FORM SIDE CODE
725 37A1 D3F8     DR3:       OUT  (0F8H),A          ;SEND SIDE+SECTOR
726 37A3 CDAF36              CALL SHRTDEL
727 37A6 3E70                LD   A,70H             ;SEEK & READ CODE
728 37A8 320010              LD   (OPSTR),A         ;KEEP IT
729 37AB F3                  DI
730 37AC D3FA                OUT  (0FAH),A          ;SEND SEEK & READ
731 37AE 0680     DR4:       LD   B,80H             ;BYTES/SECTOR
732 37B0 DBF9     DR5:       IN   A,(0F9H)          ;CRDY+RQM
733 37B2 E603                AND  03H               ;MASK LEAVES CRDY & RQM
734 37B4 28FA                JR   Z,DR5             ;WAIT FOR EITHER
735 37B6 0F                  RRCA                   ;RQM INTO CARRY
736 37B7 300C                JR   NC,DR6
737 37B9 ED78                IN   A,(C)             ;TAKE IN BYTE
738 37BB 10F3                DJNZ DR5               ;DO ALL 80H BYTES
739 37BD 2C                  INC  L                 ;INC SECTOR
740 37BE 15                  DEC  D                 ;DEC SECTORS TO DO
741 37BF 20ED                JR   NZ,DR4            ;NZ=MORE TO DO
742 37C1 D9                  EXX
743 37C2 ED78                IN   A,(C)             ;SEND TND HI
744 37C4 D9                  EXX
745 37C5 CD3635   DR6:       CALL STATUS
746 37C8 3807                JR   C,DR8
747 37CA CDDE35   DR7:       CALL DESELDRV
748 37CD CDEC35              CALL CLOCK
749 37D0 C9                  RET
750
751
752 37D1 FE06     DR8:       CP   06H               ;LAST SECTOR?
```

```
753 37D3 2006           JR   NZ,DR9          ;NZ=NOT LAST SECTOR
754 37D5 24             INC  H               ;INC TRACK
755 37D6 2E01           LD   L,01H           ;RESET TO SECTOR1
756 37D8 C39437         JP   DR2             ;DO NEXT TRACK
757
758 37DB 7C       DR9:  LD   A,H
759 37DC 320910         LD   (TKST),A
760 37DF 7D             LD   A,L
761 37E0 320A10         LD   (SCST),A
762 37E3 3A0E10         LD   A,(I00E)        ;PICK UP TRIES COUNT
763 37E6 3D             DEC  A
764 37E7 320E10         LD   (I00E),A
765 37EA CAF337         JP   Z,DRA           ;Z=FAILURE TO READ
766 37ED CD0C35         CALL SKZERO
767 37F0 C37837         JP   DR1             ;TRY AGAIN
768 37F3 37       DRA:  SCF                  ;FLAG FAILURE TO READ
769 37F4 C3CA37         JP   DR7
770
771
772
773 37F7 CD6035   FORMAT: CALL PRMDRV
774 37FA 110000         LD   DE,0000
775 37FD CD1735   F1:   CALL SHTSTS
776 3800 0610           LD   B,10H           ;10 BYTES TO WRITE
777 3802 0EFB           LD   C,0FBH          ;PORT FB
778 3804 3E00           LD   A,00H
779 3806 D3F8     F2:   OUT  (0F8H),A        ;CLEAR SECT REG.
780 3808 CDAF36         CALL SHRTDEL
781 380B 21B738         LD   HL,BUFF3        ;LIST OF SECTOR NOS.
782 380E 7A             LD   A,D             ;GET TRACK NO.
783 380F D3F9           OUT  (0F9H),A        ;SEND TRACK
784 3811 3EA0           LD   A,0A0H          ;WRITE INDEX/ID CODE
785 3813 320010         LD   (OPSTR),A       ;KEEP IT
786 3816 F3             DI
787 3817 D3FA           OUT  (0FAH),A        ;SEND WRITE INDEX/ID
788 3819 DBF9     F3:   IN   A,(0F9H)        ;GET CRDY & RQM
789 381B E603           AND  03H             ;MASK LEAVES CRDY & RQM
790 381D 28FA           JR   Z,F3            ;WAIT FOR EITHER
791 381F 0F             RRCA                 ;RQM INTO CARRY
792 3820 3019           JR   NC,F7           ;NC=NO RQM=TRACK DONE
793 3822 ED51           OUT  (C),D           ;SEND TRACK NO.
794 3824 DBF9     F4:   IN   A,(0F9H)        ;GET CRDY & RQM
795 3826 0F             RRCA                 ;RQM INTO CARRY
796 3827 30FB           JR   NC,F4           ;WAIT RQM
797 3829 ED59           OUT  (C),E           ;SEND 00
798 382B DBF9     F5:   IN   A,(0F9H)        ;CRDY & RQM
799 382D 0F             RRCA                 ;RQM INTO CARRY
800 382E 30FB           JR   NC,F5           ;WAIT FOR RQM
801 3830 EDA3           OUTI                 ;SEND SECTOR  AS (HL)
802 3832 DBF9     F6:   IN   A,(0F9H)        ;CRDY & RQM
803 3834 0F             RRCA                 ;RQM INTO CARRY
804 3835 30FB           JR   NC,F6           ;WAIT FOR RQM
805 3837 ED59           OUT  (C),E           ;SEND 00
806 3839 18DE           JR   F3              ;DO ALL SECTORS
807 383B CD3635   F7:   CALL STATUS
808 383E D8             RET  C
809 383F 78             LD   A,B             ;CHECK IF ALL TRACKS DONE
810 3840 B7             OR   A               ;BY SEEING IF B HOLDS 00
811 3841 2004           JR   NZ,F8           ;WHEN RQM NOT DETECTED
812 3843 3E80           LD   A,80H           ;SIDE 2 CODE**********
```

PAGE 15

```
 813 3845 18BF                  JR   F2                ;NOW DO SIDE 2
 814 3847 14          F8:       INC  D                 ;INC TRACK NO.
 815 3848 7A                    LD   A,D
 816 3849 FE23                  CP   23H               ;ALL DONE YET ?
 817 384B 38B0                  JR   C,F1              ;C=NOT FINISHED
 818
 819                ;ALL INDEX MARKS NOW WRITTEN
 820
 821                ;NOW WRITE TO ALL SECTORS
 822
 823 384D 3E0A                  LD   A,0AH
 824 384F 320710                LD   (NTTR),A          ;NO. OF TRIES
 825 3852 110100    F9:         LD   DE,0001           ;TRACK 00 SECT 01
 826 3855 CD1735    F10:        CALL SHTSTS
 827 3858 AF                    XOR  A
 828 3859 7A                    LD   A,D               ;GET TRACK NO.
 829 385A 1F                    RRA                    ;DIVIDE BY 2**********
 830 385B D3F9                  OUT  (0F9H),A          ;SEND TRACK NO.
 831 385D 7B                    LD   A,E               ;GET SECTOR NO.
 832 385E 3002                  JR   NC,F11            ;**********************
 833 3860 F680                  OR   80H               ;ODDS/EVENS FOR SIDES****
 834 3862 D3F8      F11:        OUT  (0F8H),A          ;SEND SIDE+SECTOR
 835 3864 CDAF36                CALL SHRTDEL
 836 3867 0EFB                  LD   C,0FBH            ;PORT
 837 3869 3EB0                  LD   A,0B0H            ;SEEK & WRITE CODE
 838 386B 320010                LD   (OPSTR),A         ;KEEP IT
 839 386E F3                    DI
 840 386F D3FA                  OUT  (0FAH),A          ;SEND SEEK & WRITE
 841 3871 0680      F12:        LD   B,80H             ;BYTES/SECTOR
 842 3873 DD6E05                LD   L,(IX+05H )       ;HL TO HOLD SOURCE
 843 3876 DD6606                LD   H,(IX+06H )       ;ADDRESS =33CB
 844 3879 DBF9      F13:        IN   A,(0F9H)          ;CRDY & RQM
 845 387B E603                  AND  03H               ;MASK LEAVES CRDY & RQM
 846 387D 28FA                  JR   Z,F13             ;WAIT FOR EITHER
 847 387F 0F                    RRCA                   ;RQM INTO CARRY
 848 3880 3007                  JR   NC,F14            ;NC=NORQM=TRACK DONE
 849 3882 EDA3                  OUTI                   ;SEND BYTE
 850 3884 20F3                  JR   NZ,F13            ;DO WHOLE SECTOR
 851 3886 1C                    INC  E                 ;INC SECTOR
 852 3887 18E8                  JR   F12               ;DO NEXT SECTOR
 853 3889 CD3635    F14:        CALL STATUS
 854 388C FE06                  CP   06H               ;LAST SECTOR ?
 855 388E 200F                  JR   NZ,F15            ;NZ=FAULT
 856 3890 1E01                  LD   E,01H             ;RESET SECTOR
 857 3892 14                    INC  D                 ;INC TRACK
 858 3893 7A                    LD   A,D
 859 3894 FE46                  CP   46H               ;ALL TRACKS DONE ?*******
 860 3896 38BD                  JR   C,F10             ;C=NO SO CONTINUE
 861 3898 CDDE35                CALL DESELDRV
 862 389B CDEC35                CALL CLOCK
 863 389E C9                    RET
 864
 865
 866
 867 389F 7A        F15:        LD   A,D               ;KEEP TRACK & SECTOR
 868 38A0 320910                LD   (TKST),A
 869 38A3 7B                    LD   A,E
 870 38A4 320A10                LD   (SCST),A
 871 38A7 3A0710                LD   A,(NTTR)
 872 38AA 3D                    DEC  A                 ;ONE TRY USED
```

PAGE 16

```
 873 38AB 320710                LD    (NTTR),A
 874 38AE CAA636                JP    Z,RDB              ;Z=ALL TRIES USED UP
 875 38B1 CD0C35                CALL  SKZERO
 876 38B4 C35238                JP    F9
 877
 878
 879                ;BUFF3 LIST OF SECTOR NUMBERS
 880
 881 38B7 01020304 BUFF3:       DB    01H,02H,03H,04H
 882 38BB 05060708              DB    05H,06H,07H,08H
 883 38BF 090A0B0C              DB    09H,0AH,0BH,0CH
 884 38C3 0D0E0F10              DB    0DH,0EH,0FH,10H
 885
 886
 887
 888                MONITOR:    EQU   00000H
 889                CRLF:       EQU   0009H
 890                MESSAGE:    EQU   00015H
 891                USER:       EQU   00003H
 892                I00E:       EQU   0100EH
 893                H302:       EQU   00302H
 894                K8A9:       EQU   038A9H
 895
 896
 897
 898
 899
 900
 901
 902                ;USE OF IX+D IN PROG.
 903
 904                ;IX ALWAYS HOLDS DVBUF=32C2H
 905
 906                ;IX+00     DRIVE NUMBER
 907                ;IX+01     TRACK
 908                ;IX+02     SECTOR
 909                ;IX+03 & IX+04  NUMBER OF BYTES
 910                ;                TO READ OR WRITE
 911                ;IX+05 & IX+06  ADDRESS TO READ
 912                ;                FROM OR WRITE TO
 913                ;IX+07     TRACK STORE
 914                ;IX+08     SECTOR STORE
 915
 916
 917                ;IX+07 & IX+08 ARE CHANGED AS
 918                ;THE READ OR WRITE PROGRESS
 919
 920                ;IX+01 & IX+02 REMAIN AT THE
 921                ;INITIAL VALUES TO ENABLE MULTIPLE
 922                ;ATTEMPTS TO BE MADE
 923
 924
 925                ;OTHER ADDRESSES USED
 926
 927
 928                OPSTR:      EQU   1000H            ;OP. IN PROGRESS
 929                DRVST:      EQU   1001H            ;DRIVE NO. IN USE
 930
 931
 932
```

PAGE 17

```
                MOTFLG:     EQU  1002H          ;01=ON 00=OFF

                DSTR1:      EQU  01003H
                DSTR2:      EQU  01004H
                DSTR3:      EQU  01005H
                DSTR4:      EQU  01006H

                NTTR:       EQU  1007H          ;TRIES COUNTER

                ERRCOD:     EQU  1008H          ;ERROR CODE STORE

                TKST:       EQU  1009H          ;TRACK STORE
                SCST:       EQU  100AH          ;SECTOR STORE
                ERRESC:     EQU  100BH          ;ERROR ESCAPE
                ;100CH 100DH ERROR EXIT ADDR.



                ;IX+00 HOLDS  00 FOR DRIVE 1
                ;             01 FOR DRIVE 2
                ;             02 FOR DRIVE 3
                ;             03 FOR DRIVE 4

                ;1001 HOLDS   1C FOR DRIVE 1
                ;             1D FOR DRIVE 2
                ;             1E FOR DRIVE 3
                ;             1F FOR DRIVE 4

                ;GENERAL LAYOUT OF MASTER DISC
                ;TRACK 00 SECTORS 01-0EH SECONDARY
                ;BOOT CODE THAT LOADS AT 9800H

                ;TRACK 00 SECTORS 0FH-10H BIT MAP

                ;TRACKS 01-03 ALL SECTORS FOR
                ;DIRECTORY (2 DIRECTORY ENTRIES
                ;IN EACH SECTOR

                ;TRACKS 04-0E BASIC INTERPRETER

                ;REMAINDER FREE FOR USE



                        END
```

INSIDE THE MZ-80K DISC

by Peter Sydenham

Introduction

The purpose of these brief notes is to give the reader an understanding of exactly what is happening when discs are in use on the MZ-80K. It should be realised that in general what is said is with reference to the discs as used within BASIC 6015. When Sharp FDOS, CP/M or any other DOS is in use there may be considerable differences in how files are held. Everything in this article has been deduced from a study of the hex. code used within BASIC 6015 and the Sharp supplied utilities, together with considerable experimentation. Much of the work was carried out using the Sharp "Machine language" tape and latterly with the disassembler written by R. Tanswell.

Tracks and Sectors

A disc is divided up into 70 tracks of 16 sectors each to give a total capacity of 1120 sectors. One sector can contain 128 bytes of data to give an overall capacity of 140K, the figure quoted by Sharp. Tracks are numbered from 0 to 69 with track 0 nearest the outer edge of the disc. Sectors are numbered from 1 to 16. In the Sharp system all even numbered tracks are on the top face of the drive and all odd numbered tracks in the lower face (when the disc is in the drive).

Not all the tracks are available for file storage, in fact the system uses tracks 0, 1, 2 and 3 for its own purposes. Thus a freshly initialised disc gives 1056 sectors ie. 132K of user file space. Tracks 1, 2 and 3 are reserved for the directory of what files are held on the disc and track 0 is used in the following way. Sectors 15 and 16 contain the bit map giving which sectors of the whole disc are free and which ones are in use for file storage. The volume number and a count of the number of the potential 1056 sectors that are in use is also held in sector 15.

A master disc has the secondary boot code in track 0 sectors 1-14. A non-master does not have this code but the sectors cannot be accessed for file storage and so represent wasted space. further space is taken up on a master by 6015 BASIC in tracks 4-13 includive, although no directory entry is present for the BASIC.

File Storage

When a file is stored on disc an entry is made in the directory of the file name and the whereabouts of the first sector of data. At the end of the first data sector of the file is the address of the next sector and so on. These links use 2 bytes giving the track number and sector number, both in hex. Thus only 126 bytes of each sector can be used for data. Figure 1 shows this linking diagramatically. The important of these links will become more apparent in later paragraphs for although in many instances a file is stored in consecutive sectors this will not always be the case.

62 BYTES			126 BYTES		
DIRECTORY ENTRY	T	S	FIRST SECTOR OF FILE	T	S

TRACK & SECTOR ADDRESS OF 1ST SECTOR OF FILE

TRACK & SECTOR ADDRESS OF NEXT SECTOR IN FILE

Bit Map

It is important that the system can determine which sectors on the disc are in use so that when a file is stored it does not overwrite part of what is already on the disc. This is achieved by keeping a map of all the sectors that may be used for file storage and indicating in the map which sectors are in use and which are free. As stated before there are 1056 sectors for file storage. One bit is used to represent each sector hence the map is 132 bytes long with 2 bytes ie. 16 bits to represent each track. Track 0 sectors 15 and 16 are used for this bit map with some of the spare bytes holding the disc volume number and a count of the number of sectors in use. In the map a free sector is represented by the relevant bit being 0 and a sector that is in use being represented by the bit being 1. Track 4, the first track available for file storage, is mapped into bytes 4 and 5 counting the first byte of the sector as byte 0. Byte 4, bit 0 is sector 1, byte 4 bit 1 is sector 2 etc. until bit 7 is sector 8. Byte 5 bit 0 is sector 9, byte 5 bit 1 is sector 10 etc. Writing bytes 4 and 5 in binary as below makes it clear.

```
0 0 0 0   0 0 0 0          0 0 0 0   0 0 0 0
  ↑         ↑                ↑         ↑
Sector 8  Sector 1         Sector 16  Sector 9

      BYTE 4                     BYTE 5
```

Bytes 6 and 7 represent track 5, bytes 8 and 9 track 6 etc. through to track 70. It shuld be noticed that tracks 0, 1, 2, 3 do not appear in the bit map since they are used by the system and hence never available for file storage. Byte 0 of sector 15 is unused. Byte 1 holds the volume number in hex. eg. if the volume number was 127 byte 1 would contain 7F. The next 2 bytes hold the number of sectors used for file storage. When the directory is displayed the number of free sectors is calculated and the free sector count displayed. Thus the overall picture is:

Byte no:	Use
0	unused
1	volume number in hex
2)	
3)	count of sectors used Z80 format
4)	
5)	bit map track 4
6)	
7)	bit map track 5
:	
:	etc.

Directory

The purpose of the directory entry is to enable the system to "know" enough about a file to be able to load it into RAM correctly. The directory entry is directly analogous to the header written to a cassette when using tape storage with BASIC 5025. Figure 2 below shows the function of each byte in the directory entry. Note that each entry is 64 bytes long and hence each sector of the directory can hold 2 entries giving a total possible of 3 x 16 x 2 = 96 file entries.

Byte No:	Function	Notes
0	file type code	02 = BASIC program
1	file name	01 = Object code program
2	in ASCII.	03 = serial data file (BASIC)
:	16 characters	04 = random data file (BASIC)
:	max.	
15		
16		
17	lock byte	00 = unlocked, 01 = locked
18	file length	
19	in Z80 hex	
20	file start addr.	)
21	in Z80 hex.	) all 0 for BASIC programs
		)
22	file execute addr.	)
23	in Z80 hex.	
24	)	
:	)	
:	)unused	
:	)	
62	)	
63	Track) link	address of first sector of
64	Sector)	the file on the disc.

File erasure

From the previous sections most of what has to be done when a file is erased shuld be apparent; it is

(a) Find all sectors on the disc that were part of the file and clear to zero the appropriate bits within the bit map.

(b) Remove the directory entry.

(c) Update the 'sectors in use' count.

In the MZ-80K when one entry in the directory is removed all subsequent entries are moved down one place to close the gap formed. Thus after a file is removed all later entries are rewritten one place down. This can result in the final entry in the directory being duplicated. To prevent ambiguity arising the now redundant entry has a zero as the file type code digit. When the system is reading the directory a zero at this place indicates that the end of the directory has been reached.

By following the link in the directory entry of the file to be erased and following the links in the file all the sectors used for it are found. The bit map is then updated together with the 'sectors in use' count. (It is this following of links that makes erasure so slow). When a new file is saved on disk free sectors are used up in the order they appear on the disc. After many file erasures and fresh files being saved, some files may occupy sectors scattered across the disc, hence the importance of the links at the end of each sector.

Start Up

When FD is typed from monitor, the following sequence of events takes place. The monitor examines the contents of address $F000. If it is found to be other than 00, the monitor retains control. (on the disc controller card ROM is fitted that starts at address $F000 and the contents of the first byte is 00.) If the controller card is fitted the code beginning at $F000 is executed. This code is the primary boot which reads track 0 sectors 1 to 14 inclusive and loads this secondary boot into RAM starting at address $9800. The first byte of this code is then examined and if it is $C3 control is passed to $9800. This code then reads tracks 4-13 inclusive into RAM starting at address $1200. BASIC 6015 has now been loaded and control is handed over to it.

Dear Sharpsoft,

SUN 11 Page 13 Bit Printing to Epson MX80

Thank you for your letter, enclosing letters from R.G. Newton with his query regarding error 4 in line 16 of my Bit Printing program.

I have gone through the program, and can find no error here, it is a simple routine to poke a machine code routine into a particular area of memory. The routine is to send a signal to the printer.

This program was written on an MZ-80K with Disc Basic 6015. I note that Prof. Newton is using an MZ-80A with a modification, and with Basic 6510. As I am not conversant with this machine etc., I can only suggest that this is the cause.

I have been trying to find the time to write to you with a few amendments to the program to make it easier to use and correct any errors that have been entered, I now take this opportunity to do so.

Alter the following lines as follows:-

```
 17 NEXT:T$="X":CT=0:REM("X"=shift '/')
 51 PRINT" (17 spaces)    4 = 11111111":GOSUB 209
 65 change(GOTO 56) to (GOTO 55)
 66 IF P$="→"THEN J=J-1:PRINT"■■";:A$(L)=LEFT$(A$,J-1):GOTO58
145 GOSUB 202:PRINT"WHICH LINE IS TO BE ALTERED";
146 INPUT WL:IF WL > R3 THEN 146
147 add to end of line(:IF WB > R(WL-1)THEN147)
170 POKE LM+WA,R1:IF CT=1 THEN 145
```

Add the following lines:-

```
202 PRINT"[C]TO CHANGE TO & FROM CONTINUOUS MODE TYPE 'C'"
203 PRINT"  TO CONTINUE TYPE ANY KEY":CURSOR 24,2:
      PRINT"[";T$;"]"
204 GET CT$:IF CT$=""THEN 204
205 IF(CT$="C")*(CT=0)THEN T$="C":CT=1:GOTO 207
206 IF(CT$="C")*(CT=1)THEN T$="X":CT=0
207 CURSOR 25,2:PRINT T$
208 RETURN
209 PRINT"          →   = DELETE LAST DIGIT"
210 PRINT"                OR IF 1st, LAST LINE"
211 RETURN
```

Many, many thanks for the update. EDITOR

B. GOLD
LEYTONSTONE

Dear Sirs,

I have a SIRS disc purchased from you a year ago for the MZ-80B. I understand you have a batch update module available now and would like to have the price on this. Could you also tell me when the other SIRS modules will be available and at what price. If they have been discontinued or you do not intend to proceed with them would you let me know this? If the disc file output module is available would it work with WORDPRO (KUMA)?

I also have your MZ-80B Utilities disc and in Appendix G of the manual you mention that future releases will include all disc utilities in both CP/M and Sharp format. Can I now have my present master disc upgraded in this way and cost please.

Finally, in Appendix F of the disc utilities manual, you mention that to get a CP/M disc to boot on the B track 0 sector 1 (IPL sector) must be in Sharp format. Could you advise me how to do this as I have a couple of CP/M discs that won't boot and I suspect this is the reason.

A. THORES
FIFE

The author of the Disk Utility is currently tied up with producing routines for the new Sharpsoft High-Res Graphics Board for the MZ-700, so upgrades are not expected in the near future.

The information in Appendix F of the disk utility just explains how the system works. All it means is that when the MZ-80B is turned on (or the IPL reset pressed) the IPL Monitor automatically reads Sector 1 of track 0 from the disk. Hence this part of the disc needs to be in a Sharp format for it to be recognised by IPL. If a CP/M Master disk is being loaded, the initial data read in by UP, from Sector 1, Track 0, changes the system to a CP/M format, enabling it to read the rest of the disk. This ony applies to a CP/M Master disk and not one that only contains programs and data. There is nothing you can do to read your CP/M disks other than buy the CP/M operating system and thereunder use the CP/M program SYSGEN.COM to allow data disk to boot.

SIRS Version 2.0 is now available on the MZ-80B. The versions for the PC 3201 and MZ-3500 being released early August. It is not proposed to update the MZ80A version unless there is interest shown by sufficient MZ80A users.

SIRS V.2.0 Improvements over V.1.0 and V.1.1

FILE ENHANCEMENTS

Up to 99 files or data bases now supported.
Up to 99 FIELDS per data base file.
Field names and file names can now be changed at any time.
When creating a new data base the format of an existing data base can be copied.
It is also hoped to increase the capacity of the sort program so that larger selections can be sorted. N.B. this option may not be available by the dates given above.

NUMERICAL ACCURACY

Numerical accuracy is now specified at file creation. The number of decimal positions required is specified and the total numerical field length can be specified to a maximum of 10 characters.

AUTOMATIC KEY GENERATION

This means that the next sequential key number is generated automatically if required. This is useful if no specific key is being used e.g. where data is being stored temporarily and is deleted after use.

COMPULSORY FIELDS

Fields can be specified as being compulsory so that when records are being added or amended the field cannot be passed or left blank. The compulsory status can be changed at any time.

VALID CHARACTER SPECIFICATION

A field can be set up so that only certain characters can be entered in the field, these characters being specified by the user. For example, assume a field is to hold a department code and only departments A,B,C and D exists, the field can be set up so that only A,B,C and D can be entered. The specified characters can be changed at any time. If the Compulsory bield facility is used (see above) then A,B,C or D MUST be entered.

SELECTIVE PRINTING

When printing records, fields can be selected for printing so that tha whole record need not be printed.

Current V.1.0 and V.1.1 users may have their original SIRS Master Disc updated for £20.00. SIRS V.2 is £80 for first time buyers.

AVAILABLE ADD-ON MODULE

BATCH UPDATE PROGRAM £20.00

The S.I.R.S. Batch Update Program gives the facility for fast input of large numbers of file and/or record updates. Also Batch modification of numeric fields is possible using addition, subtraction, multiplication and division functions.

FURTHER SIRS MODULES

Following requests from users the following S.I.R.S programs are also available. They were written to solve special problems that users had, but may be useful to others. Each of these modules would require a) a certain amount of 'tailoring' for specific applications and or b) a sound working knowledge of SIRS basic

database. If you are interested in discussing these modules with the Author please write or ring with your day and evening telephone number.

SIRSCALC PROGRAM
A powerful calculation and processing program which will work on either an entire file, single or selected groups of records. Full four function maths, alpha and date field processing. A maximum of eight formulae containing up to five elements. Four accumulators and four work areas are available, their contents being printed at run end. Processes allow movement of information from field to field, with the result of calculations stored in a separate field.

FILE MERGE
This program reads several S.I.R.S. files of the SAME format and creates a composite S.I.R.S. system from them. Originally designed to merge floppy files onto a Winchester Hard Disk, it will also work on several smaller floppy based files. Having created the larger file, the program then recreates the Key file, sorts it and reports on any duplicate keys that may exist.

KEY FILE RE-CREATION
This program recreates the key file from a data base file and was originally written for a customer who accidentally erased a key file from a S.I.R.S. system. Due to the design of S.I.R.S. disk files, corrupt or damaged systems can often be rescued without loss of data.

S.I.R.S. INPUT PROGRAM
This program takes input from an external source (e.g. a word processor or some other user program) and reformats the data ready for input to the S.I.R.S. batch input system. This was written to link a customers accounting system sith S.I.R.S.

SHARPSOFT

LEICESTERSHIRE LOCAL SHARP USER GROUP MEETINGS

VENUE:	The Old Grammar School Market Harbough Leicestershire	DATES:	WED 13 JUNE 1984 WED 12 SEPT 1984 WED 12 DEC 1984
TIMES:	7:30 pm - 11:00 pm		EVERY 3 MONTHS
COST:	Approx. £1.00 (depending upon number attending!)		
DETAILS:	Telephone: Mr. D. Peniston-Bird (0536) 514282		

Dear Sharpsoft,

In SUN Issue 9, M. Bellamy writes saying he cannot save a 'modified Disk BASIC'.

I enclose a program which will write MZ-80K SP-6015 Disk BASIC to tape, although I cannot create a back-up copy on disk to replace the Master Disk which seems to be wearing out.

I might be able to change the Slave disk copier present on the Master Disk, since I think it checks the first byte on the disc to see if its 'C3' - therefore a Master Disk.

Thank you for the SUN, it brightens up our lives.

```
5 REM BY JAMES HOLLAND JUNE'84
10 POKE8048,1:REMOVE PEEK PROTECT
20 POKE4354,44:POKE4355,83:REM BYTE LENGTH OF DISK BASIC
30 POKE4356,0:POKE4357,18:REM START ADDRESS-$1200
40 POKE4358,250:POKE4359,33:REM EXECUTION ADDRESS-$21FA
45 REM STORE PROGRAM NAME
50 FORX=4337TO4353:POKEX,13:NEXTX
60 A$="DISK BASIC"
70 FORX=1TOLEN(A$):POKEX+4336,ASC(MID$(A$,X,1)):NEXTX
75 REM SAVE DISK BASIC TO TAPE
80 USR(33):USR(36)
```

J. HOLLAND
STOCKPORT

Dear Sharpsoft,

I wrote to you in March asking you to give information on the MZ-80K output port address, and to advise on any obtainable publication giving details of I/O device use. At this date I have not received a reply. I hope I am not requesting anything outside the scope of membership. Failing any reply I must seriously consider whether to rejoin on expiry of my present year.

D. HALLIDAY
DUMFRIES

Information on Port addresses and I/O device use is very hard to come by for the MZ-80K (and most other Sharp computers). In our opinion the best sources of material are the "Commented Assembly Listing for the MZ-80K SP-2001 Monitor", early Issues of our User Notes (1981's Issues + a few others) and for really technical details the Sharp MZ-80K I/O Box Service Manual.

Sharp use Z80 port numbers with the eighth bit set to 1 (i.e. numbers above 128) for their peripheral equipment. If you intend to use the Sharp I/O card then setting the switches to port numbers between 0 and roghtly 100 will not cause a conflict with the standard peripherals.

Unfortunately we cannot guarantee a speedy reply to the hundreds of queries we receive, however, we do try very hard to offer a good service in a field of computing which lacks support.

EDITOR

Dear Sir,

In SUN 12 Page 38. G.H. Bent asked for suggestions to speed up his golf club partner-selection program. I suggest the following.

The main problems seem to lie with the series of four nested loops between lines 850 to 960. These loops take the form:

```
850 FOR A=1 TO N
860 FOR B=A TO N1
870 FOR C=B TO N2
880 FOR D=C TO N3
920
930 NEXT D
940 NEXT C
950 NEXT B
960 NEXT A
```

The rejection tests cause the 'action' to be skipped if, for example, IFA$(A)="". However, because this test is in the innermost loop the action is skipped for every value of the D loop from C to N3! Worse, it is skipped not for one D loop but for all D loops within the C loop from B to N2!! Worse still, it is skipped for all D loops within all C loops within the B loop from A to N1!!! Even when IA$(A) "" the test is performed unnecessarily in all these loops.

The place to put the test is immediately after A has been given a value, i.e. in line 850. If IA$(A) fails the test then ALL the other loops can be bypassed by going straight to line 960. If it passes the test it does not need to be tested again in the inner loops. Similar comments apply to tests for IB$(B), IC$(C) etc. Thus, the above program segment might become.

```
850 FOR A=1 TO N : IF IA$(A)="" THEN 960
860 FOR B=A TO N1 : IF IB$(B)="" THEN 950
870 FOR C=B TO N2 : IF IC$(C)="" THEN 940
880 FOR D=C TO N3 : IF ID$(D)="" THEN 930
890
920
930 NEXT D
940 NEXT C
950 NEXT B
960 NEXT A
```

Always put loop skipping tests OUTSIDE as many loops possible, never save them all for the innermost loop.

Another problem lies in the values of N1, N2 and N3 in lines 860 to 880 above. In the original program these take the values N-1, N-2 and N-3 respectively. Consider what happens to the D loop when C takes the value N-2. Line 880 becomes, in effect: FOR D=N-2 TO N-3. Now though values of D greater than N-3 are not meaningful to the program, this loop will be performed once with D taking the value N-2 (since FOR..NEXT loops are ALWAYS evaluated at least once), Similarly, when B is N-1 the C and D loops will be performed with C and D being N-1; and when A is N the B,C and D loops will be performed with B,C and D being N. To avoid this the loops are perhaps best evaluated "inside-out" i.e.

```
850 FOR D=1 TO N-3
860 FOR C=D TO N-2
870 FOR B=C TO N-1
880 FOR A=B TO N
930 NEXT A
940 NEXT B
950 NEXT C
960 NEXT D
```

I have learned both of the above fetures the "hard" way. I hope they are of some use to Mr. Bent and others.

I enclose a few miscellaneous "quickies" below. Long listings tend to reflect more interesting programs but typing them in is tedious. The mixed bag below won't set the world alight, but they won't take all night to enter either!

A. STEVENS
DERBY

Whytoff's Game.
You and the computer alternately take 'stones' from one or the other or both lines. If both, must be same number from each. Whoever takes last one WINS. You go first.

```
 1 DEF FNA(X)=INT(C*(Y-X))-X: DEF FNB(X)=INT(X*RND(1))+5
 2 DIM L$(3): L$(1)="o": L$(2)="x": L$(3)="both": P$="I": A$=">": B$=">"
 3 C=1.61803: A=FNB(10): X=A: B=A+FNB(-3): Y=B
 4 FORI=1TOA: A$=A$+"o": NEXT: FORI=1TOB: B$=B$+"x": NEXT: PRINT"■"
 5 A$=LEFT$(A$,A+1):PRINTTAB(20-A);A$;A:B$=LEFT$(B$,B+1):PRINTTAB(20-B);B$;B
 6 INPUT"Which line?  1 = 'o', 2 = 'x', 3 = both ";Q: IF (Q<1)+(Q>3) THEN 6
 7 INPUT"How many? ";N: IF (N<1)+(N>Y) THEN 7
 8 IF (Q=1)*(N>A)+(Q=2)*(N>B)+(Q=3)*(N>X) THEN 7
 9 GOSUB13: IF A+B<1 THEN P$="You": GOTO 12
10 M=FNA(X): GOSUB14: PRINT"I take";N;" of ";L$(Q)
11 GOSUB13: IF A+B>0 THEN 5
12 PRINT P$;" win.": FORI=1TO3000: NEXT: RUN
13 A=A+N*(Q<>2): B=B+N*(Q<>1): Y=B+(B-A)*(B<A): X=A+B-Y: RETURN
14 IF M<0 THEN N=-M: Q=3: RETURN
15 Q=1-(B>A): IF M=0 THEN N=1: RETURN
16 T=Y: Y=INT(X*C): Z=INT(X/C+0.5): F=FNA(X): N=T+Y*(F=0)+Z*(F<>0): RETURN
```

+0·5

The 15 Game.
You and the computer alternately take 1 to 4 stones from the line. Whoever takes the last one LOSES. You go first.

```
 1 DEF FNA(X)=X-5*INT(X/5): DEF FNB(X)=X+(X>1)-4*(X=0)
 2 A=15: A$="oooooooooooooo": P$="You": PRINT"■"
 3 GOSUB7: INPUT" You take? (1-4) ";N: IF (N<1)+(N>4) THEN 3
 4 A=A-N: IF A<1 THEN P$="I": GOTO 6
 5 N=FNB(FNA(A)): GOSUB7: PRINT" I take";N: A=A-N: IF A>0 THEN 3
 6 PRINT P$;" win.": FOR I=1 TO 3000: NEXT: RUN
 7 A$=LEFT$(A$,A): PRINT TAB(15-A);A$;: RETURN
```

Hex (H$) to decimal (D) single-line subroutine:

```
 1 D=0: FORI=1TOLEN(H$): A=ASC(MID$(H$,I,1))-48: D=D*16+A+(A>9)*7: NEXT: RETURN
```

Decimal (D) to hex (H$) three-line subroutine:

```
 1 H$="": X$="0123456789ABCDEF"
 2 N=INT(D/16): H$=MID$(X$,1+D-N*16,1)+H$: D=N: IF D THEN 2
 3 RETURN
```

```
Generate the Fibonacci sequence (ad nauseam!):
 1 PHI=(SQR(5)+1)/2: N=1/SQR(5)
 2 FOR I=0 TO 1 STEP 0: N=N*PHI: PRINT INT(N+0.5);: NEXT


Ulam's problem. Do all numbers end up in loop 1-4-2-1 ?
 1 DEF FNA(N)=N/2-INT(N/2): C=0: INPUT"Whole number =? ";N
 2 M=FNA(N): N=-(3*N+1)*(M<>0)-(N/2)*(M=0):C=C+1: PRINTN;: IF N>1 THEN 2
 3 PRINT C
```

Dear Sharpsoft,

The following may be of some interest to fellow readers of SUN.

A feature of Sharp Basic is that if a program is to display text both on the VDU and on a Printer, two sets of identical program lines are needed, one using the PRINT statement and the second using PRINT/P.

On the MZ-80A using BASIC SA-5510 it is possible to alter this situation and direct output at will to either the VDU or Printer using only the normal PRINT statement.

Memory location $2DBB normally contains the value 80 hex and this is used to direct the output to the VDU when the PRINT statement is used. Changing the value to 82 hex will ccause the PRINT statement to direct the output to the Printer.

The enclosed test program demonstrates this feature.

Care must be taken when dealing with text which contains cursor control characters as these can have undesirable effects when the output is direct to the printer.

It is possible to obtain an annotated listing of BASIC SA-5510 and, if so, what would be the cost?

```
10 REM *** OUTPUT TO VDU/PRINTER ***
15 TX$="TEXT TO BE PRINTED"
20 PRINT"KEY [1] FOR OUTPUT TO VDU"
25 PRINT"KEY [2] FOR OUTPUT TO PRINTER"
30 GETA$:IF A$=""THEN30
35 IFA$="2"THENGOSUB60
40 PRINT:PRINTTX$
45 REM *** RESTORE OUTPUT TO VDU ***
50 POKE$2DBB,$80
55 END
60 REM *** CHANGE PRINT TO PRINT/P ***
65 POKE$2DBB,$82:RETURN
```

Unfortunately Sharp will not release any listings of their system software nor will they allow anyone to publish them.

SHARPSOFT

A DICE SIMULATION GAME FOR THE MZ-80A

by Simon Jones

```
1 REM      ┌──────────────────────┐
2 REM      |   DICE SIMULATION    |
3 REM      |                      |
4 REM      | written By S.Jones   |
5 REM      |                      |
6 REM      |    15th June 84      |
7 REM      └──────────────────────┘
8 REM
9 DIM S$(6),RD(6)
10 S$(1)="┌──────┐  ┌──────┐  ┌──────┐  ┌──────┐"
20 S$(2)="┌──────┐  ┌──────┐  ┌──────┐  ┌──────┐"
30 S$(3)="┌──────┐  ┌──────┐  ┌──────┐  ┌──────┐"
40 S$(4)="┌──────┐  ┌──────┐  ┌──────┐  ┌──────┐"
50 S$(5)="┌──────┐  ┌──────┐  ┌──────┐  ┌──────┐"
60 S$(6)="┌──────┐  ┌──────┐  ┌──────┐  ┌──────┐"
100 PRINT"■"
110 PRINT TAB(12);"Dice Simulation"
120 PRINT TAB(12);"──────────────"
130 PRINT"■"
140 PRINT"  This program shows the probability"
150 PRINT"■  of dice throwing. The computer will"
160 PRINT"■  randomly throw the dice, and display"
170 PRINT"■  them at the top of the screen. At the"
180 PRINT"■  bottom of the screen is a bar chart"
190 PRINT"■  showing this in the form of a graph."
210 PRINT"■■  You may restart the program at any"
220 PRINT"■  time by pressing the SPACE bar."
230 PRINT"■■";TAB(7);"PRESS ANY KEY TO CONTINUE";
240 USR($09B3)
250 PRINT"■"
260 GOSUB 460
265 REM
270 REM ==========
280 REM MAIN LOOP
290 REM ==========
300 REM
310 RA=INT(RND(1)*6+1)
320 CURSOR 28,4:PRINT S$(RA)
330 GOSUB 610
340 RA=INT(RND(1)*6+1)
350 CURSOR 34,4:PRINT S$(RA)
360 GOSUB 610
390 FOR Y=1 TO 50:GETT$:IF T$=" "THEN RESTORE:GOTO 250
395 NEXT
400 GOTO 310
410 REM
420 REM ==============
430 REM SET UP SCREEN
440 REM ==============
450 REM
460 CURSOR 12,0:PRINT"Dice Simulation."
464 REM
465 REM ==========
466 REM BAR-CHART
467 REM ==========
468 REM
```

```
480 CURSOR 14,21:PRINT"                "
490 CURSOR 14,22:PRINT"1 2 3 4 5 6"
500 FOR AD=1 TO 6
510 READ T
520 RD(AD)=T
530 POKE RD(AD),212
540 NEXT:RETURN
550 DATA 54062,54064,54066,54068,54070,54072
560 REM
570 REM =================
580 REM UP-DATE BAR CHART
590 REM =================
600 REM
610 IF PEEK(RD(RA))=212 THEN POKE RD(RA),208:GOTO 620
615 IF PEEK(RD(RA))=208 THEN RD(RA)=RD(RA)-40:POKE RD(RA),212
620 IF RD(RA)<53407 THEN 640
630 RETURN
640 CURSOR 13,23:PRINT"HIT ANY KEY.";
650 USR($09B3)
660 RESTORE:GOTO 250
```

Dear Sharpsoft,

I am pleased to say that I've just received S.U.N. Issue No. 11. I discovered Sharpsoft on 4-4-84 when passing through London on a hurried business trip.

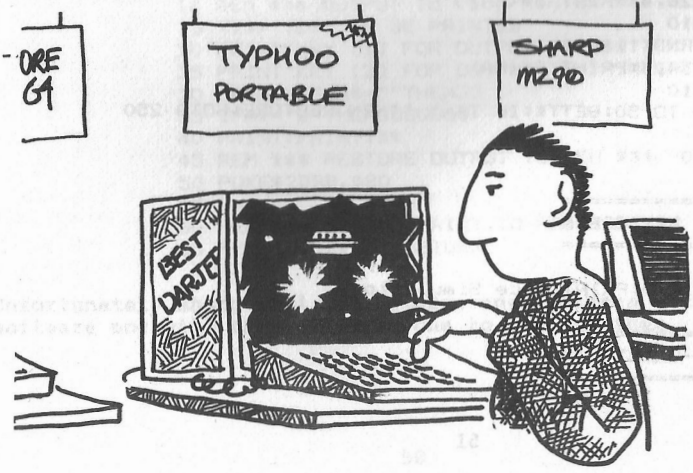
I have the MZ-80B, floppy drives and a P5 printer.

I use the machine for two main purposes, firstly, engineering and secondly to assist me with the membership and financial records of the local cine society of which I am the Honorary Treasurer.

As my main interest is in engineering and because one rarely sees any serious engineering programs listed in the various weekly and monthly magazines I am prompted to ask if there are any fellow SUN members with engineering programs which they may wish to share through the S.U.Notes.

I would also appreciate some guidelines for changing listings for A & K machines so that they can run on the MZ-80B. (PASCAL (Hisoft) as well as BASIC).

G.D. WAY
AUSTRALIA



Sharpsoft Ltd., 86-90 Paul Street, London EC2A 4NE
Printed by Oldham Press (T.U.) Chatham, Kent.