(2)

SHARPSOFT

SHARPSOFT - USER NOTES

Issue number 2 June 1981

Thank you all for supporting these user notes. For a venture like the SHARPSOFT user notes to be a success it needs support from our readers. The initial response means that we are in a position where the first years overall costs, printing etc, are covered. Our aim is not to make money on these user notes but to offer M2-80k users a low cost publication where ideas can be exchanged. Most of the editorial work is done on a part time basis which helps to keep costs as low as possible. So if you have comments, ideas for articles or indeed articles do send them to us. In this issue we include a number of programs and articles from readers. Mr. P.L. Birch has been using "Word-Processor WPI. and has extended it for his own needs. These extensions are useful and we hope they will allow other users to improve their word processing Software

Over the last few months quite a number of M2-80k owners have written to us with ideas for the user notes and related topics. Here are two of the more general interesting points.

One reader requested that SHARPSOFT start a "listing service" for those M2-80k owners who do not have a printer. This we are prepared to do. initially for a trial period, to see if there is enough demand. To start with we propose to restrict the service to BASIC programs on cassette only. A small charge of 30p will be made for each cassette listing. So if you have a program on cassette which you would like listed send it to us with a strong 9" x 11" self-addressed stamped envelope, including the 30p charge, and we will return your cassette with a listing. One program per cassette please.

A second reader has suggested that many M2-80k owners might be interested in corresponding with other owners and could we generate a list of M2-80k owners who are prepared to exchange letters. This sounds like a good idea but, of course, does require us to generate a list of names and addresses. Again we are prepared to generate this list if there is enough demand from our readers. If you are interested write to SHARPSOFT and we will send out a form which you can fill in. Once returned to us these forms will be used to generate a list of M2-80k owners who are interested in corresponding with other users.

In this issue of the user notes we include a number of programs. SHARP (UK) have given us permission to publish the listing of their version of the game STAR TREK. This is for your personal use only and is a copyright program.

A TINY PILOT, with editor, program is also included. We hope you find these programs interesting and fun. The TINY PILOT was written to help those readers who have never used PILOT experiment with the language. It represents a collection of the best ideas previously published in the popular computing magazines. Again this program is for your personal use and must not be copied for commercial publication.

Since "Crystal Electronics" released their version of CP/M the amount of Software, particularly new languages, which has become available for the M2-80k has increased considerably. We are concious however, that the majority of our readers do not have discs or CP/M. At SHARPSOFT we use CP/M based M2-80k computers for program development and whenever possible attempt to make cassette versions of software available as well as disc.

New computer languages for the M2-80k is one of the areas we are concentrating on. Two languages, "Tiny-C" and "Forth", have so far been developed by us to run on the M2-80k. We are currently able to offer our users both cassette and disc versions of the Tiny-C language and a disc version of FORTH. In the next few months more work is needed to complete the cassette version of FORTH.

As soon as this becomes available we will inform our readers. A short article on "Tiny-C" is included in this issue of the user notes.

We are often asked why new languages? It is true that both the SHARP and XTAL versions of BASIC are very good. However, an important point worth remembering is that BASIC is not a structured language, Tiny-C is structured similar to PASCAL.

SHARP BASIC reference notes

DISC BASIC

The SHARP DISC BASIC (SP-6015) is an integrated package which consists of a set of BASIC commands, Keywords and functions similar to the SHARP cassette BASIC with, however, extensions for disc file creation and access, and a simple to use disc operating system

Each disc file is described by an extension, SHARP call these extensions Modes. File extensions are given the names

BTX	BASIC text file (program source)
BSD	BASIC sequential file (sequential data)
BRD	BASIC random access file (random access data)
OBJ	OBJECT code file (machine code programs or data)

The contents of a disc may be listed on the console V.D.U. by entering the command DIRn (1 - 4), followed by a carriage return; where (1 - 4) is the disc drive number.

Listed files with a "*" following the extension name indicate that the file is LOCKed and cannot be deleted from the disc using the DELETE command.

Disc Operating System Commands

LOCK — This command locks previously recorded files. If a file is locked, the file may not be DELETED or RENAMED until it is unlocked using the UNLOCK command.
The general form of the LOCK command is

LOCK FDd, "File name"

Where d is the disc drive number (1 - 4) Which contains the file called "File name".

UNLOCK — This command unlocks a previously recorded and LOCKed file. Its general form is

UNLOCK FDd, "File name"

Where d is the disc drive number which contains the file called "File name".

RENAME — This command is used to rename files. Its general form is

RENAME FDd, "File name 1", "File name 2"

The file on disc drive d called "File name 1" is renamed "File name 2". NOTE the contents of the file being renamed is not changed by the rename operation.

DELETE — This command deletes a named file from a disc. Its general form is

DELETE FDd, "File name"

In each of the above operating system commands, excluding the DIR command, if the disc drive number is NOT specified BASIC SP-6015 will assume the disc drive number used in the last DIRn (1 - 4) command. This short form of the operating system commands simplifies the use of the LOCK, UNLOCK, RENAME and DELETE commands. Also remember following a DIR operation full screen edit mode operation is allowed using the keyboard yellow keys to position the flashing cursor.

Files with the extension OBJ.

SHARP BASIC SP-6015 does NOT generate files with the extension OBJ. This extension is reserved for machine code program or machine code data files. These programs are normally generated using a Z80 Assembler, for example the SHARP SYSTEMS program package, or the SHARP MACHINE LANGUAGE program. Both these packages are cassette based programs where the final object code program is stored on magnetic tape in a form which can be directly loaded and run. To generate a disc file which contains an object code program the SHARP "tape to disc" transfer utility program must be used. Instructions for this operation are outlined in a latter section of these user notes. However, once a file with the extension OBJ is created it may be run from disc using the RUN command. Execution of the object code program is automatically started from its load address, normally 1200H. NOTE, using the LOAD command for a file with an OBJ extension will cause an error (file error 63).

Disc data files

There are two types of disc data files which may be created and accessed by a SHARP BASIC program; there are sequential files and random access files.

Sequential files

Sequential files are easier to create than random access files but are less flexible when accessing the data. The data that is written for a sequential file is stored in the order, i.e. in sequence, that it is written to the file and is read from the file in the same order.

The sequential disc file BASIC commands are:-

a. Writing data to a file.

File open command	WOPEN#n, "File name"
Data write command	PRINT#n, data
File close command	CLOSE#n
Cancel command	KILL#n

b Reading data from a file.

File open command	ROPEN#n, "File name"
Data read command	IMPUT#n, variable
File close command	CLOSE#n
File end Test	IF EOF (#n) THEN.....

Where n is called the file "logical number" which must be in the range 1 n 127.

The complete specification for the file open commands are

WOPEN # n, FDd @V "File name"
ROPEN # n, FDd @V "File name"

Here, d is the disc drive number and V is the disc volume number. NOTE, often when creating or accessing sequential files the d and V extensions are not used. They do however, allow, for example, the identification of individual discs in a disc library.

The following steps are required to create a sequential data file and accesses the data stored in the file.

1	Open the file - for writing	WOPEN#1, "TEST"
2	Write data for the file using the print statement	PRINT# 1, A$, B$, C$
3	To access the data stored in the file, the file must first be closed then reopened using the ROPEN statement.	CLOSE # 1 ROPEN # 1, "TEST"
4	Data is read from the file using the input statement	INPUT # 1, P1$, P2$, P3$.

The examples given in steps 1 to 4 show a file opened with logical number 1. Due to the high speed access of a disc, when compaired with a cassette tape, and the parallel arrangement of files on a disc, SHARP BASIC programs can simultaneously open and access a number of different files. SHARP BASIC allows a maximum of 10 files to be opened at one time.

The following example may help to clarify the sequential file access procedure.

```
1Ø WOPEN#5, "DATA"
2Ø INPUT "NAME"; N$
25 IF N$ = "END" THEN 1ØØ
3Ø INPUT "DATA 1 "; DI$
4Ø INPUT "DATA 2"; D2$
5Ø PRINT  5, N$, DI$, D2$
6Ø PRINT: GOTO 2Ø

1ØØ CLOSE#5 : END
```

BSD file creation program.

```
1Ø ROPEN#5, "DATA"
2Ø INPUT#5, N$, DI$, D2$
3Ø IF EOF (#5) THEN CLOSE  5: END
4Ø PRINT N$, D1$, D2$
5Ø GOTO 2Ø
```

Sequential access of a BSD file - output to a V.D.U.

Adding data to a sequential file.

If you have created a sequential data file on disc and wish to add more data to the end of the file then the following procedure could be used. We first open the orginal file, called "DATA" in the following example, and copy its contents for a tempory file called "TEMP". Finally, the orginal DATA file is deleted and the contents of TEMP are copied back to DATA to form the updated file.

```
1Ø ROPEN#1, "DATA"
2Ø WOPEN#2, "TEMP"
3Ø INPUT#1, A$
4Ø IF EOF (#1) THEN 1ØØ
5Ø PRINT#2, A$
6Ø GOTO 3Ø
1ØØ INPUT " NEW DATA"; A$
11Ø IF A$ = "END" THEN 2ØØ
12Ø PRINT#2, A$
13Ø GOTO 1ØØ
2ØØ CLOSE#1
21Ø CLOSE#2
22Ø DELETE DATA
3ØØ ROPEN#2, "TEMP"
31Ø WOPEN#1, "DATA"
32Ø INPUT#2, A$
33Ø IF EOF (#2) THEN 4ØØ
34Ø PRINT#1, A$
35Ø GOTO 32Ø
4ØØ CLOSE#1
41Ø CLOSE#2
5ØØ DELETE "TEMP"
6ØØ END
```

The above example could be improved, of course, but has been deliberately written in small blocks to demonstrate sequential file handling.

NOTE, the tempory file is deleted at the end of the up-date sequence in order to remove from the work disc an unwanted file. Also note that when an attempt is made to read beyond the end of a file the special function EOF (n) is set to "TRUE". This condition is tested using an IF...THEN statement, for example in lines 40 and 33Ø, where a branch to a latter section of the program occurs.

Random access files.

Creating and accessing random access files requires more program steps than sequential data files. The biggest advantage to random access files is that data can be accessed randomly anywhere on a disc and it is not necessary to read through all the information in a file to find a specific entry.

In a random access file data is stored and accessed in distinct units called records. Each record is numbered. When writing to or reading from random access files the PRINT and INPUT statements are modified to include a record number. The general form of these statements is

PRINT n (record number), data
INPUT n (record number), variable

Each record is recorded on disc with a fixed 16 byte length. It is inportant to remember that numeric variables will always fit into 16 bytes, even when expressed in exponential notation, but STRING variables can have a length up to 255 bytes. Hence, normally strings cannot be recorded in one record but extend over a number of records.

There is only one open statement for random access files, this is called XOPEN and has the general form

XOPEN n, FDd @V "File name"

Two examples, showing how to use the random access file commands are given on page 44 of the SHARP DISC BASIC manual. These examples should be studied in detail if you are not familiar with random access files.

It is also possible to write BASIC programs which use both sequential and random access data files in the same program. Complex and powerful file systems, for example in data base programs, often use sequential indexed random access files. These techniques are possible with SHARP BASIC and there readers interested in experimenting with file structures should consult any of the standard texts on the BASIC language.

CHAINing BASIC programs.

The storage capacity of a disc is far greater than the random access memory storage of the largest M2 - 80k computer. Hence, it is theoretically possible to store on disc BASIC programs which have a length greater than 48k. However, it is not possible to load and run such large programs. To overcome this problem SHARP disc BASIC is equiped with a special command called CHAIN.

The CHAIN statement is used to load and run a BASIC program from an already loaded program. An important feature of BASIC is the fact that all variables are global. When the CHAIN statement is interpreted by BASIC not only is the new program loaded by BASIC to RAM and run but the variables, and their values, in the previous program are preserved for use by the new program. This feature allows very large programs to be segmented into unique sections then run using a series of connecting chain statements. The general form of the CHAIN statement is

CHAIN FDd @V "Program name"

Note, the CHAIN statement can be considered as an extended GOTO statement with the disc acting as a virtual memory.

The SWAP Statement.

The SWAP statement provides a similar function to the CHAIN statement. It is interesting to note that the SWAP statement may be considered as an extended SUBROUTINE facility with the disc acting as a virtual memory.

When the SWAP statement is interpreted by BASIC the current program in memory is stored on the disc and a new program is loaded from disc and run i.e. SWAPed. On completion of the new program the reverse process takes place with the old program SWAPed back to memory. Interpretation of the original program then continues from the statement following the SWAP statement. Again variables are preserved during the SWAP sequence. Note, nesting of SWAP levels is NOT allowed. The SWAP statement is a non-standard feature of SHARP disc BASIC. The keyword SWAP is used in other BASIC'S,for example the Microsoft MBASIC interpreter and BASCOM compiler, but in general it has an entirely different meaning.

The general form of the SWAP statement is

SWAP FDd, @V, "File name.

DISC BASIC Utility Programs.

Three machine code utility programs are provided by SHARP with BASIC SP-6015. There utilities are used to initialize a new disc, copy programs from one disc to another and to transfer programs from cassette to disc.

Disc Initialization Utility.

This program is used to format a new disc. Discs must be formatted before they can be used to store programs or data.

Disc Copy Utility.

This program allows the contents of one disc, usually in drive 1, to be copied to a second disc, usually in drive 2. However, the program will not copy the master disc system and BASIC SP-6015 tracks. A special extended version of the copy program is available from SHARPSOFT which does allow the entire contents of a master disc to be copied. We use this extended version for making back-up copies of our master disc.

Cassette tape file to disc transfer utility.

This utility is in two parts. Part one is used to transfer files held on cassette to disc. The cassette files can be BASIC programs or data, or object code programs. Do remember that BASIC programs,developed using SP-5025, with POKES and PEEKS will often not run using SP-6015. Some slight changes are normal.

The second part of this utility is used to transfer, or convert, files on cassette with the OBJ extension to a disc file with the BSD extension. The BSD file can then be output to the VDU or printer with the hex memory dump represented by ASC II codes. The listing program is given on page 178 of the SHARP disc manual.

Machine code routines with BASIC.

Often with large programs or programs where a considerable number of calculations take place or programs which require constant updating of a graphics display then interpreted BASIC is simply not fast enough.

All modern BASIC interpreters allow users to include machine code routines in their programs. BASIC is linked to the machine code routine using the PEEK, POKE and USR statements.

Two points are important when using machine code routines;

1. You must have a working knowledge of Z80 machine code and Assembly code programming.

2. Machine code routines are much faster than BASIC - the speed improvement can be as high as 100 times faster.

Normally machine code routines are loaded with a BASIC program - this is to make the compete program self-contained. The machine code operation codes and data are placed in BASIC DATA statements, remember DATA statements expect the data in decimal not binary or hex. On running the BASIC program these DATA statements are READ and the data is POKED into memory - one byte at a time.

Normally machine code routines are POKED to memory at the top end of RAM well away from the BASIC interpreter and your BASIC program. To stop BASIC overwriting your machine code SHARP BASIC allows the user to specify the end of the BASIC work area with the LIMIT statement.

Once a machine code routine is installed in RAM it may be called from BASIC using the USR statement. The argument to the USR statement is the absolute memory address of the start of the machine code routine. Do remember the last operation code in your machine code routine must be a RET statement, otherwise control will not return to BASIC on completion of the machine code routine.

An example of a BASIC program with a machine code routine included is shown on page 119 of the SHARP MZ-80K user manual.

Further information on machine code routine linkage to BASIC is given in the SHARP disc BASIC and systems program manuals.

BASIC SP-5025 Hints.

1. Enter LOAD: RUN and a program will load and run itself.

2. USR (68) turns sound on.
   USR (71) turns sound off.
   USR (62) sounds a bell.

3. POKE 4465,1 to 40 moves the cursor axis.
   POKE 4466,1 to 24 controls the cursor axis.

   Thes POKES save typing in all those reverse field arrows for cursor movement.

4. POKE 57347,4 turns the front panel LED from green to red.

5. POKE 59555,Ø blanks the screen.
   POKE 59555,1 restores the video.

6. POKE 10167,1 removes the PEEK protect from SP-5025 BASIC.
   Note, an unprotected copy of BASIC can be made by the following steps.
   1. load BASIC
   2. enter POKE 10167,1
   3. enter the single live program
      1Ø USR (33) : USR (36)
   4. then enter RUN and a copy of the SHARP BASIC will be saved to tape.

7. BASIC programs will auto start if you enter POKE 10682,1 before saving to tape.

If you have any tips - POKES etc. write to us and we will include them in a later edition of the "user notes".

Letters

Our thanks to everyone who wrote to us since the publication of the first issue of the "user notes". Your comments and ideas will allow us to provide information which helps all M2-80k users get the best from their computer. Here are a few of the points from our readers.

I have found that if you run the program

```
1Ø GET A$ : PRINT A$: GOTO 1Ø
```

The key 'DEL' produces a spaceship, CR a  , SML a  and various other things from the yellow keys.

SIMON ROCKMAN

HA89QN

I would like to point out that the LET command can be used in BASIC SP-5025, but can be ommitted (in common with most BASICS).

In response to your suggestions for programming tips, POKE 57346,75 turns the front panel light red,and POKE 57346,79 turns it green. As the upper/lower case mode is not affected these have been useful for

1. Indicating an operator error (when flashing from red to green, and accompanied by sound).
2. Indicating when input is required.

B. TANNER

WR1A 1HN

There is an apparent error in numerical variable range quoted on page 22 (user notes issue 1). On my M2-80k with SHARP BASIC SP-5025 the range is approximately only 10-19 to 10+19.

As I am a scientist and frequently find the need to use higher range number than this I would be interested in whether the SHARP BASIC SP-5025 can be POKED to allow higher range numbers.

D.J. STEEL

BH22 8QU

Comments on SHARPSOFT User Notes:- very useful. Some suggestions for future editions.

a. More information on the use of PEEK, POKE and USR statements.
b. Because I cannot read music and cannot use the sound aspect of the M2-80k as much as I would like. How about some programs for popular melodies, light classical etc.
c. Programming tricks e.g. FOR I=1 TO 1ØØØ : PRINT " " ; NEXT I for generating a pause in the program.

There must be dozens of things like this.

B.P. WINDIBANK

L37 1PE

As a newcomer to the computing world I am sure that other beginners experience the same problem as I do, which is trying to find out what some of the simpler terminology actually means

e.g. Whatever is an "Editor/assembler/debugger"
"BASIC toolkit"
"Can I use XTAL on my basic M2-80k?"

I must say however that your catalogue goes a long way to answering some of these questions, but could you include in one of the issues of "SHARPSOFT USER NOTES".simple explanations to some of the termenology that would be of use to the layman who is starting from scratch with a computer.

L.M. BEADLE

OX14 2HG

Tiny-c for the M2-80K

Two versions of tiny-c are now available for the M2-80k; a cassette version for 48k machines without discs and a CP/M version.

Tiny-c is a subset of the C programming language developed at Bell Laboratories, Murry Hill, New Jersey, U.S.A.

Tom Gibson and Scott Gurthery of Tiny-c Associates designed the small c subset which is well adapted to the microcomputer environment. The subset is distributed by Tiny-c Associates in a hard copy form with instructions for implementing the language on 8080/280 microcomputers. Sharpsoft have written the input/output and cassette storage machine code routines needed to run tiny-c on the M2-80k. The cassette version of tiny-c is available as a package including a copy of the tiny-c owners manual and the language cassette which loads from the SP2001 monitor. A seperate CP/M version on disc configured to run under the XTAL CP/M operating system can also be supplied on request. This version uses disc program storage rather than cassette.

Tiny-c is really two seperate packages. These are called "Tiny-c one", and "Tiny-c Two". Tiny-c one is an interpreter for the C language subset and Tiny-c Two is a compiler for the C language subset. Tiny-c Two is only available from Tiny-c Associates to run on a CP/M machine with 8" discs. In the future we hope to offer a version of Tiny-c Two to run on both cassette and disc based M2-80k computers. Work has started at SHARPSOFT on configuring the Tiny-c Two package for the M2-80k. We anticipate that three to six months development time are needed before we can offer the compiler to our M2-80k users.

The interpreter and compiler are upward compatable products. Normally programs are developed using the interpreter then compiled using the Tiny-c compiler. The compiler produces code which gives a speed improvement of roughly "Ten"Times. This, of course, makes those "slow" programs much more viable.

C is versatile, expressive general - purpose programming language which offers economy of expression, modern structured control flow and data structures, and a rich set of operators. C is not a "very high level" language, or a "big" one, nor is it specialized to any particular area of application. Its absence of restrictions and its generality make it remarkably convenient and effective for a wide variety of computing tasks.

C is concise - you don't have to write a lot of code to get a job done. Yet at the same time, C programs are readable - you can understand what you, or someone else, have written. This combination of brevity and readability is rare in programming languages, and is part of the reason that C is so widely used.

Tiny-c retains C's expressiveness, conciseness and readability, yet sacrifices very few of its features. At the same time, Tiny-c provides a computing environment that will make it easier to develop programs. The language is provided with an editor and other library functions that together make a program preparation system.

A Tiny-C program

```
/ * guess a number between 1 and 100
/ * T.A. Gibson
guessnum [
     int guess, number
     number = random (1,1ØØ)
     pl "guess a number between 1 and 100"
     pl "type in your guess now"
     While (guess  ! = number) [
          guess = gn
          if (guess = = number) pl "right!!"
          if (guess   number) pl "too high"
          if (guess   number) pl "too low"
          pl " "; pl " "
        ]  /* end of game loop
```

```
]       /* end of program
/*
/* random - generates a random number
int seed, last /* globals used by random
random int, little, big [
    int range
if (seed = = ∅) seed = last = 99
range  = big - little + 1
last = last  seed
if (last  ∅) last = - last
return little + (last / 8)% range
]
```

If you believe "Tiny-c ONE" is for you contact us at SHARPSOFT for the latest information on availability and cost of the "Tiny-c ONE" package.

Games STAR TREK

If an opinion poll was taken amoung computer hobbyists regarding the most widely known programs then it is likely that the game STAR TREK would come in the top ten. STAR TREK is a classical computer game. Versions vary in size from 4k "Tiny treks" to a CP/M 46k "Megatrek". The rules tend to be similar for most versions and are typically as follows.

The object of STAR TREK is to destroy all the klingon ships within a given number of star dates. The galaxy is divided into 64 quadrants, with each quadrant split into 64 sections. The galaxy is initialized with stars, Klingons and a number of star bases placed randomly throughout the 64 quadrants. A quadrant is initialized each time it is entered by the Enterprise.

You as Captain Kirk control the Enterprise; its movement, photon torpedoes and defence screens. The star bases are available to replenish your supplies.

Six commands are available to you:-

1. Fire phasers.
2. Fire photon torpedo.
3. Move the Enterprise.
4. Display short-range sensor scan.
5. Display long-range sensor scan.
6. Display known galaxy.

You may change the computer readout, move the Enterprise or fire either a photon torpedo or your phasers at any time. If you are in a quadrant occupied by a klingon, he will fire at you and move each time you enter the quadrant, manoeuvre, or fire at him. We would like to thank SHARP (UK) for allowing us to publish the listing of their version of STAR TREK.

Good hunting !

```
1 REM***STAR TREK Copyright SHARP ELECTRONICS (UK) LTD.***
2 DIM S1$(6),T1$(6),A1$(6),R1$(6),E1$(6),K1$(6),H(7)
3 DIM D(5),K1(7),K2(7),K3(7),S(7,7),Q(7,7),D$(5),R$(5)
5 DIM K$(2),E$(2),B$(2),S$(2),C$(2),T$(2),W$(2),Z$(2)
20 G1$=" SSS▒▒▒▒▒S▒▒▒S▒▒▒▒▒▒S▒SSS▒S▒▒▒▒▒▒S▒SSS▒S▒▒▒▒▒"
25 G2$="TTTTT▒▒▒▒T▒▒T▒▒T▒▒T▒▒T▒▒T▒▒▒▒▒▒▒▒"
30 G3$="  A▒▒▒A A▒▒▒▒▒A   A▒▒▒▒▒▒AAAAA▒▒▒▒▒▒A   A"
35 G4$="▒▒▒▒▒▒A    A▒▒▒▒▒▒A    A▒▒▒▒▒▒":G5$=G3$+G4$
40 G6$="RRRR▒▒▒▒▒R    R▒▒▒▒▒▒R    R▒▒▒▒▒▒RRRR▒▒▒▒▒R    R▒▒▒▒▒▒"
45 G7$="R   R▒▒▒▒▒▒R   R▒▒▒▒▒▒":G8$=G6$+G7$
50 G9$="EEEEE▒▒▒▒▒▒E▒▒E▒▒EEEE▒▒▒▒▒E▒▒E▒▒EEEEE▒▒▒▒▒▒"
55 G0$="K   K▒▒▒▒▒▒K   K▒▒▒▒▒K K▒▒▒▒KK▒▒▒▒K   K▒▒▒▒▒K   K▒▒▒▒▒▒K   K▒▒▒▒▒▒"
60 PRINT"▒▒▒";G1$;"  ";G2$;"  ";G5$;"  ";G8$;"  "
65 PRINT:PRINT"▒▒▒▒▒▒▒":PRINT TAB(7);G2$;"   ";G8$;"   ";G9$;"  ";G0$
70 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒BY SHARP CORPORATION"
75 FOR K=1 TO 200:NEXT K
80 MM$="¯B0R¯BR¯BR¯BR¯BR":M1$="¯B0R":M2$="C3R0C3R0C3R"
85 M3$="C0DEFGAB¯CCDEFGAB¯C":M4$="¯CBAGFEDC":TEMPO 5
90 G$="4▒▒3▒▒2▒▒▒▒▒▒\ | /▒▒▒▒▒▒▒5─   ─1▒▒▒▒▒▒▒/ | \▒▒▒▒▒▒▒6  7  8"
95 K$(0)=" ┌┬┐ ":K$(1)="└X┘"
100 B$(0)=" ┌─┐ ":B$(1)="<※>":E$(0)="┐┌─":E$(1)="▀▀ "
105 S$(0)=" * ":S$(1)="   ":C$(0)=" . ":C$(1)="   ":T$(0)="▀▀▀":T$(1)=T$(0)
110 R$(1)=C$(0):R$(2)=E$(0)+"▒▒▒▒"+E$(1)+"▒":R$(3)=K$(0)+"▒▒▒▒"+K$(1)+"▒"
120 R$(4)=B$(0)+"▒▒▒▒"+B$(1)+"▒":R$(5)=S$(0)
130 D$(0)="WARP ENGINES":D$(1)="SHORT RANGE SENSORS"
131 D$(2)="LONG RANGE SENSORS":D$(3)="PHASERS":D$(4)="PHOTON TORPEDOES"
140 D$(5)="GALACTIC MAP"
145 DEF FND(N)=SQR(ABS((K1(I)-S1))↑2+ABS((K2(I)-S2))↑2)
150 GOTO 5000
155 I=ASC(E$)
160 I=I-11*INT(I/11):FOR J=0 TO I:K=RND(1):NEXT:GOSUB 3120
180 GOSUB 710:GOSUB 530:Q1=X:Q2=Y:X=8:Y=1:X1=.2075:Y1=6.28:X2=3.28
190 Y2=1.8:A=.96:C=100:W=10:K9=0:B9=0:S9=400:T9=3451:GOTO 210
200 K=K+(N<X2)+(N<Y2)+(N<.28)+(N<.08)+(N<.03)+(N<.01):K9=K9-K:GOTO 230
210 T0=3421:T=T0:E0=6000:E=E0:P0=15:P=P0:FOR I=0 TO 7
220 FOR J=0 TO 7:K=0:N=RND(Y):IF N<X1 THEN N=N*64:K=(N<Y1)-Y:GOTO 200
230 B=(RND(Y)>A):B9=B9-B:Q(I,J)=K*C+B*W-INT(RND(Y)*X+Y):NEXT J,I
240 IF K9>(T9-T0) THEN T9=T0+K9
250 IF B9>0 THEN 270
260 GOSUB 530:Q(X,Y)=Q(X,Y)-10:B9=1
270 PRINT"▒":MUSIC MM$: K0=K9:PRINT"♦  OBJECTIVE: DESTROY";K9;
275 PRINT" KLINGON BATTLE CRUISERS IN";T9-T0;
280 PRINT" YEARS.":PRINT"▒♦ THE NUMBER OF STARBASES IS ";B9;" ▒▒";
290 FOR TX=1 TO 10:NEXT
300 GOSUB 2100
310 A=0:IF (Q1<0)+(Q1>7)+(Q2<0)+(Q2>7)THEN N=0:S=0:K=0:GOTO 330
320 N=ABS(Q(Q1,Q2)):Q(Q1,Q2)=N:S=N-INT(N/10)*10:K=INT(N/100)
330 B=INT(N/10-K*10):GOSUB 530:S1=X:S2=Y
340 FOR I=0 TO 7:FOR J=0 TO 7:S(I,J)=1:NEXT J,I:S(S1,S2)=2
350 FOR I=0 TO 7:K3(I)=0:X=8:IF I<K THEN  GOSUB 540:S(X,Y)=3:K3(I)=S9
360 K1(I)=X:K2(I)=Y:NEXT:I=S
370 IF B>0 THEN GOSUB 540:S(X,Y)=4
380 IF I>0 THEN GOSUB 540:S(X,Y)=5:I=I-1:GOTO 380
385 FOR J=1TO2000:NEXT
390 GOSUB 650
```

```
400 IF E<=0 THEN 1590
410 I=1:IF D(I)>0 THEN 3300
411 IF K>0 THEN GOSUB 20100
420 PRINT"▣";:MUSIC MM$:PRINT"▣":PRINT"LIBRARY COMPUTER  (STATUS REPORT)"
430 PRINT:M$=" > ":MUSIC M1$:PRINT M$;"YEARS=";T9-T
435 PRINT:MUSIC M1$:PRINT M$;"STARDATE=";T:PRINT
440 MUSIC M1$:PRINTM$;"CONDITION=";C$:PRINT:MUSICM1$
443 PRINTM$;"QUADRANT=";Q1+1;"-";
445 PRINT Q2+1:PRINT:MUSIC M1$
450 PRINT M$;"SECTOR=";S1+1;"-";S2+1:PRINT
465 MUSIC M1$:PRINT M$;"ENERGY=";E
466 FOR U=1 TO E/160:PRINT TAB(4);"% ";:NEXT U:PRINT:PRINT
467 MUSIC M1$:PRINT M$;D$(4);"=";P
468 FOR V=1 TO P:PRINT TAB(4);"♠ ";:NEXT V:PRINT:PRINT
470 MUSIC M1$:PRINT M$;"KLINGON ";K9:MUSIC M1$
472 FOR Z=1 TO K9:PRINT TAB(4);CHR$(96);" ";:NEXT Z
480 FOR TX=1 TO 5000:NEXT
500 GOSUB 1900
510 GOTO 750
530 X=INT(RND(1)*8):Y=INT(RND(1)*8):RETURN
540 GOSUB 530:IF S(X,Y)>1 THEN 540
550 RETURN
560 IF K<1 THEN RETURN
570 IF ((E<>6000)+(P<>15))*(C$="DOCKED") THEN 580
572 IF (E=6000)*(P=15)*(C$="DOCKED") THEN RETURN
573 IF A=5 THEN PRINT"▣▣▣▣"
575 GOTO 600
580 C$="DOCKED":E=E0:P=P0
590 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣  ▣ DOCKING TO STAR BASE"
592 IF (FG=5)+(FG=4) THEN 595
593 GOSUB 32300
595 RETURN
600 FOR I=0 TO 7:IF K3(I)<=0 THEN NEXT:RETURN
610 H=K3(I)*0.4*RND(1):K3(I)=K3(I)-H:H=H/(FND(0)↑0.4):E=E-H
620 E$=" ENTERPRISE FROM KLINGON ":N=E:GOSUB 630:NEXT:RETURN
630 PRINT H;" UNIT HIT ON ";E$;" AT SECTOR";K1(I)+1;"-";K2(I)+1
640 PRINT" (";INT(N);" LEFT)":FOR Z=1 TO 500:NEXT: RETURN
650 FOR I=S1-1 TO S1+1:FOR J=S2-1 TO S2+1
660 IF (I<0)+(I>7)+(J<0)+(J>7) THEN 680
665 IF ((E<>E0)+(P<>P0))*(S(I,J)=4) THEN GOSUB 710:GOSUB 580:RETURN
670 IF S(I,J)=4 THEN C$="DOCKED":GOSUB 710:RETURN
680 NEXTJ,I:IF K>0 THEN C$=CHR$(105):RETURN
690 IF E<(E0*0.1) THEN C$=CHR$(103):RETURN
700 C$=CHR$(104):RETURN
710 FOR N=0 TO 5:D(N)=0:NEXT:RETURN
720 MUSIC M2$: PRINT D$(I);" DAMAGED"
730 PRINT"**";D(I);" YEARS ESTIMATED FOR REPAIR."
740 IF A=1 THEN RETURN
750 MUSIC MM$:INPUT "♦ COMMAND ? ";A$
760 IF (ASC(A$)<49)+(ASC(A$)>54) THEN MUSIC M2$:GOTO 785
770 IF FG=6 THEN PRINT"▣"
775 A=VAL(A$):FG=A
777 IF (A=1)+(A=5) THEN GOSUB 7000
780 ON A GOTO 830,390,1510,1320,800,1550
785 PRINT"▣COMMAND IS ▣"
```

```
790 FOR I=0 TO 5:PRINTI+1;"= ";D$(I):NEXT:PRINT "■":GOTO 750
800 IF D(4)<=0 THEN 820
810 MUSIC M2$:PRINT"TO STRIKED AGAINST A METEORITE !":I=4:GOTO 730
820 N=15:IF P<1 THEN MUSIC M2$:PRINT " NO TORPEDOES LEFT !":GOTO 750
830 IF A=5 THEN MUSIC M1$:PRINT" PHOTON TORPEDOES";
840 MUSIC M1$:INPUT" COURSE(1-8.9) ";C:IF C<1 THEN 830
850 IF C>=9 THEN 830
860 IF A=5 THEN P=P-1:GOTO 3400
870 MUSIC M1$:INPUT"■ WARP (0.125-12) ";W:IF (W)=0.125)*(W<=12) THEN 880
873 PRINT"■■"
875 PRINT"WARP IS FROM 0.125 TO 12                INPUT AGAIN !!."
877 PRINT"■■":GOTO 870
880 IF (W<0.250001 )+(D(0)<=0) THEN 910
890 I=0:MUSIC M2$:PRINT D$(I);" DAMAGED";
900 MUSIC M2$:PRINT" - MAX WARP IS 0.25 ":GOSUB 730:GOTO 870
910 FOR I=1 TO 900:NEXT: PRINT"■":GOSUB 2100:IF (E-W)<0 THEN 1590
920 TX=0:IF RND(1)>0.25 THEN 1000
930 X=INT(RND(1)*6):IF RND(1)>0.5 THEN 960
940 D(X)=D(X)+INT(6-RND(1)*5):GOSUB 2850
950 IF X<>1 THEN MUSIC M2$:PRINT"■■■■";D$(X);" DAMAGED !":PRINT
953 I=X
955 IF X<>1 THEN GOSUB 730:GOTO 958
956 GOSUB 740
958 D(X)=D(X)+1:GOTO 1000
960 FOR I=0 TO 5:IF D(I)>0 THEN 970
965 GOTO 997
970 D(I)=.5
980 IF TX>0 THEN 997
990 MUSIC M1$:GOSUB 28010
991 PRINT
992 PRINT" ** SPOCK USED A NEW REPAIR TECHNIQUE **"
993 PRINT"■■■"
997 NEXT
998 TX=0
1000 FOR I=0 TO 5:IF D(I)=0 THEN 1020
1010 D(I)=D(I)-1:IF D(I)<=0 THEN D(I)=0:MUSIC M1$:GOTO 1015
1011 GOTO 1020
1015 PRINT"■    ♦ ";D$(I);" ARE FIXED !          "
1020 NEXT:N=INT(W*8):E=E-10*N+0.5:T=T+1:S(S1,S2)=1
1030 V1=S1+0.5:X1=S2+0.5:IF T>T9 THEN 1590
1040 V=(C-1)*0.785398:X=COS(V):V=-SIN(V)
1050 FOR I=1 TO N:V1=V1+V:X1=X1+X:V2=INT(V1):X2=INT(X1)
1060 IF (X2<0)+(X2>7)+(V2<0)+(V2>7) THEN PRINT:GOTO 1260
1070 IF A=5 THEN GOSUB 2300
1080 IF S(V2,X2)=1 THEN NEXT:GOTO 1210
1100 IF S(V2,X2)<=3 THEN 1120
1110 ON S(V2,X2)-3 GOTO 1180,1160
1120 IF A=1 THEN PRINT"♦";:GOTO 1200
1130 FOR I=0 TO 7:IF V2<>K1(I) THEN 1150
1140 IF X2=K2(I) THEN K3(I)=0
1150 NEXT:K=K-1:K9=K9-1:GOTO 1220
1160 IF A=5 THEN S=S-1:GOTO 1220
1170 GOSUB 7100:GOTO 1203
1180 IF A=5 THEN B=B-1:GOTO 1220
1190 PRINT"■■■";
```

```
1195 GOTO 1203
1200 PRINT" BLOCKED BY KLINGON"
1203 Y2=INT(Y1-Y):X2=INT(X1-X)
1205 FOR Z=1 TO 1000:NEXT
1210 S1=Y2:S2=X2:S(S1,S2)=2:A=2:GOTO 390
1220 GOSUB 2400:IF Y2=7 THEN 1240
1230 FOR M=1 TO 3*(7-Y2):PRINT:NEXT
1240 PRINT"■■■■":S(Y2,X2)=1:Q(Q1,Q2)=K*100+B*10+S
1245 IF K9<1 THEN 1620
1250 GOSUB 560:IF E<=0 THEN 1590
1255 GOSUB 650:GOTO 750
1260 IF A<>5 THEN 1300
1290 MUSIC M2$:PRINT:PRINT"■■■■■■■■■■■■■■■■■■■■  MISSED !!---■■■":GOTO 1250
1300 Q1=INT(Q1+W*Y+(S1+0.5)/8):Q2=INT(Q2+W*X+(S2+0.5)/8)
1310 Q1=Q1-(Q1<0)+(Q1>7):Q2=Q2-(Q2<0)+(Q2>7):GOTO 310
1320 I=3:IF D(I)>0 THEN 720
1330 MUSIC M1$:PRINT'■";D$(3);" READY:ENERGY UNIT TO FIRE";
1340 INPUT X:IF X<=0 THEN 750
1350 IF X>E THEN MUSIC M2$:PRINT"ôv1 /# k kb1 f":E:GOTO 1330
1355 IF D(1)<=0 THEN GOSUB 1900:GOTO 1360
1356 PRINT"■■■■♦ ";D$(1);" DAMAGED  !":PRINT D(1);"YEARS ESTIMATED FOR REPAIR"
1360 Y2=S1:X2=S2:GOSUB 2600
1370 E=INT(1000*(E-X))/1000:Y=K:FOR I=0 TO 7:IF K3(I)<=0 THEN 1400
1375 IF Y=0 THEN K3(I)=0 :GOTO 1400
1380 H=X/(Y*(FND(0)↑0.4)):K3(I)=K3(I)-H:H(I)=H
1390 GOSUB 2800
1400 NEXT
1410 IF Y2=7 THEN 1430
1420 FOR M=1 TO 3*(7-Y2):PRINT:NEXT
1430 PRINT"■■■■■■■■■■■■■■■■■■■■"
1440 FOR I=0 TO 7:IF K3(I)=0 THEN 1490
1450 E$="KLINGON":N=K3(I):H=H(I):GOSUB 630
1460 IF K3(I)>0 THEN 1490
1470 PRINT"---KLINGON DESTROY---"
1480 K=K-1:K9=K9-1:S(K1(I),K2(I))=-1:Q(Q1,Q2)=Q(Q1,Q2)-100:K3(I)=0
1490 NEXT:IF K9<1 THEN 1620
1500 GOTO 1250
1510 I=2:IF D(I)>0 THEN 720
1520 PRINT"■  "D$(I);"( QUADRANT";Q1+1;"-"Q2+1;")"
1530 GOSUB 1700
1540 GOTO 750
1550 I=5:IF D(I)>0 THEN 720
1560 PRINT"■";D$(I);" STARDATE ";T
1570 GOSUB 2900:GOTO 750
1580 PRINT"■■■■■■■■■■■■■■■■■♦ sTARDATE ";T;" ♦";"■■■":RETURN
1590 GOSUB 1580:PRINT"THANKS TO YOUR BUNGLING. THE FEDERATION ■ WILL BE";
1600 PRINT" CONQUERED BY THE REMAINING■":PRINTTAB(8);K9;" KLINGON CRUISERS !"
1610 PRINT"■■■■■YOU ARE DEMUTED TO CABIN BOY  !■■":GOSUB 32101:GOTO 1660
1620 GOSUB 1580:PRINTTAB(6); "FEDERATION HAS BEEN SAVED !":PRINT
1630 PRINTTAB(5); "YOU ARE PROMOTED TO ADMIRAL !!":PRINT
1640 PRINT " YOU DESTROYED";K0;" KLINGONS IN";T-T0;" YEARS."
1650 PRINTTAB(12);"■■RATING=";INT(K0/(T-T0)*1000);"■"
1655 GOSUB 32201
1660 INPUT"TRY AGAIN ?  (YES=Y, NO=N)";E$
1670 IF LEFT$(E$,1)="Y" THEN 100
```

```
1680 GOTO 150
1700 PRINT"■":MUSIC MM$: M1=5:PRINT TAB(M1);
1710 FOR M0=Q2-1 TO Q2+1:M1=M1+4:PRINT M0+1;TAB(M1);:NEXT:PRINT
1720 PRINT TAB(4);"┌──┬──┬──┐"
1730 FOR M0=Q1-1 TO Q1+1
1740 PRINT M0+1;TAB(4);:FOR M1=0 TO 3:PRINT"│  ";:NEXT M1:PRINT
1750 PRINT TAB(4);"├──┼──┼──┤":NEXT M0
1760 PRINT"■■";:PRINT:PRINT TAB(4);"└──┴──┴──┘":PRINT"■■■■■■■";:PRINT
1770 FOR I=Q1-1 TO Q1+1:M1=5:PRINT TAB(M1);
1780 FOR J=Q2-1 TO Q2+1
1790 IF (I<0)+(I>7)+(J<0)+(J>7) THEN PRINT"***";:GOTO 1820
1800 Q(I,J)=ABS(Q(I,J))
1810 E$=STR$(Q(I,J)):PRINT RIGHT$("00"+E$,3);
1820 M1=M1+4:PRINT TAB(M1);:NEXT J:PRINT:PRINT:NEXT I
1830 PRINT"■■■■■";:PRINT:PRINT TAB(12);:TX=0
1840 PRINT"■■■";
1850 GOSUB 2750:ON TX GOTO 1870,1880,1870,1880,1870,1880,1870,1880
1860 PRINT"■■■■":RETURN
1870 USR(62):PRINT"   ";:GOTO 1840
1880 IF (Q1<0)+(Q1>7)+(Q2<0)+(Q2>7) THEN PRINT"***";:GOTO 1840
1890 E$=STR$(Q(Q1,Q2)):PRINT RIGHT$("00"+E$,3);:GOTO 1840
1900 PRINT"■■";:MUSIC MM$
1910 FOR I=0 TO 7
1920 PRINT TAB(3);STR$(I+1);
1930 FOR J=0 TO 7:PRINT TAB(7);R$(S(I,J));:NEXT:PRINT:PRINT
1940 NEXT:RETURN
2100 PRINT"■■■■■■":PRINT"
2110 PRINT"
2120 PRINT"
2130 PRINT"
2140 PRINT"
2150 PRINT"
2160 PRINT"          ENTERPRISE
2170 PRINT"
2180 PRINT"
2190 PRINT TAB(7);"■
2200 PRINT TAB(7);"
2210 PRINT TAB(7);"
2220 PRINT TAB(7);"
2230 PRINT TAB(7);"
2240 PRINT
2250 TX=0
2260 PRINT"■■■■■■■■■■";TAB(4);
2265 TX=TX+1:ON TX GOTO 2275,2280,2275
2270 FOR M=0 TO 8:PRINT:NEXT:RETURN
2275 PRINT TAB(4);"┼┼┼";TAB(24);"■ ":PRINT TAB(4);"┼┼┼■■":GOTO 2265
2276 PRINT TAB(4);"┼┼┼■■":GOTO 2265
2280 MUSIC M3$ :PRINT"        ";TAB(24);"■ ":PRINT"          ■■":GOTO 2265
2300 PRINT"■■":IF Y2=0 THEN PRINT"■":GOTO 2320
2310 FOR M=2 TO 2*Y2:PRINT:NEXT
2320 MUSIC"MB0":PRINT TAB(3*X2+7);"  ■":PRINT TAB(3*X2+7);" ● "
2325 PRINT TAB(3*X2+7);"  "
2340 RETURN
2400 ON S(Y2,X2)-3 GOTO 2420,2430
2410 FOR I=0 TO 2:W$(I)=K$(I):GOTO 2440
```

```
2420 FOR I=0 TO 2:W$(I)=B$(I):GOTO 2440
2430 FOR I=0 TO 2:W$(I)=S$(I)
2440 NEXT:TX=0
2450 PRINT"■":IF Y2=0 THEN PRINT"■";:GOTO 2470
2460 FOR M=2 TO 2*Y2:PRINT:NEXT
2470 TX=TX+1:ON TX GOTO 2500,2490,2500
2480 RETURN
2490 PRINT:FOR M=0 TO 1:MUSIC"_C3_G0":PRINT TAB(3*X2+7);W$(M):GOTO 2510
2500 PRINT:FOR M=0 TO 1:PRINT TAB(3*X2+7);C$(M)
2510 NEXT:GOTO 2450
2600 FOR M=0 TO 1:W$(M)=E$(M):Z$(M)=T$(M):GOTO 2620
2610 FOR M=0 TO 1:W$(M)=K$(M):Z$(M)=C$(0)
2620 NEXT:TX=0
2630 PRINT"■":IF Y2=0 THEN PRINT;:GOTO 2650
2640 PRINT:FOR M=2 TO 2*Y2:PRINT:NEXT
2650 TX=TX+1:ON TX GOTO 2680,2670
2660 RETURN
2670 PRINT"■":MUSIC M4$:FOR M=0 TO 1:PRINT TAB(3*X2+7);W$(M)
2675 NEXT M:GOTO 2690
2680 FOR M=0 TO 1:PRINT TAB(3*X2+7);Z$(M):NEXT M:MUSIC M4$
2690 GOTO 2630
2750 TX=TX+1:RETURN
2800 Y2=K1(I):X2=K2(I)
2810 IF K3(I)>0 THEN GOSUB 2610:GOTO 2840
2820 FOR M0=0 TO 1:W$(M0)=K$(M0):NEXT M0:TX=0
2830 GOSUB 2450
2840 RETURN
2850 MUSIC "_C3_G0_C3_G0_C3_G0_C3_G0"
2855 PRINT"■■■■■";:FOR I=10TO18:FOR J=1 TO I:PRINT" ";:NEXT
2865 FOR J=1 TO 19-I:PRINT"└";:NEXT J
2866 FOR J=12TO4STEP-1:POKE 4514,J:USR(68):NEXT
2870 FOR J=1 TO 19-I:PRINT"┘";:NEXT J
2871 FOR J=4TO12:POKE 4514,J:USR(68):NEXT
2885 FOR KA=1 TO J-1:PRINT" ";:NEXT KA
2887 PRINT:NEXT I
2889 FOR I=1 TO 7:PRINT:NEXT I
2890 PRINT"■■■■■";"■■■■■■■■■■■■■** SPACE STORM **":RETURN
2900 PRINT"■":MUSIC MM$ : M1=4:PRINT TAB(M1);
2910 FOR M0=1 TO 8:M1=M1+4:PRINT M0;TAB(M1);:NEXT:PRINT
2920 PRINT"■■■┌──";:FOR M0=0 TO 6:PRINT"┬──";:NEXT:PRINT"┐"
2930 FOR M0=1 TO 8
2940 PRINT M0;" ";:FOR M1=0 TO 8:PRINT"|   ";:NEXT M1:PRINT
2950 PRINT"■■■├──";:FOR M1=0 TO 6:PRINT"┼──";:NEXT M1:PRINT"┤":NEXT
2960 PRINT"■■■■└──";:FOR M0=0 TO 6:PRINT"┴──";:NEXT:PRINT"┘"
2970 FOR M0=0 TO 15:PRINT"■■";:PRINT:NEXT
2980 FOR I=0 TO 7:M1=4:PRINT TAB(M1);
2990 FOR J=0 TO 7
3000 IF Q(I,J)<0 THEN PRINT"***";:GOTO 3020
3010 E$=STR$(Q(I,J)):PRINT RIGHT$("00"+E$,3);
3020 M1=M1+4:PRINT TAB(M1);:NEXT J:PRINT:PRINT:NEXT I
3030 IF (Q1<0)+(Q1>7)+(Q2<0)+(Q2>7) THEN RETURN
3040 FOR M0=0 TO 17:PRINT"■";:NEXT:PRINT
3050 FOR M0=0 TO Q1*2:PRINT:NEXT:TX=0
3060 PRINT TAB(Q2*4+7);
3070 PRINT"■■■";:GOSUB 2750
```

```
3080 ON TX GOTO 3100,3110,3100,3110,3100,3110,3100,3110
3090 FOR M0=1 TO (9-Q1)*2:PRINT:NEXT:RETURN
3100 USR(62):PRINT" "::GOTO 3070
3110 E$=STR$(Q(Q1,Q2)):PRINT RIGHT$("00"+E$,3);:GOTO 3070
3120 PRINT"▨▨":MUSIC MM$
3130 FOR M=0 TO 1:PRINT TAB(3);E$(M):NEXT
3140 PRINT"▨▨";TAB(11);"<ENTERPRISE>"
3150 PRINT"▨":FOR M=0 TO 1:PRINT TAB(3);B$(M):NEXT
3160 PRINT"▨▨";TAB(11);"<STARBASE>"
3170 PRINT"▨":FOR M=0 TO 1:PRINT TAB(3);K$(M):NEXT
3180 PRINT"▨▨";TAB(11);"<KLINGON>"
3190 PRINT:PRINT:FOR M=0 TO 1:PRINT TAB(3);S$(M):NEXT
3200 PRINT"▨▨";TAB(11);"<STAR>"
3210 PRINT:PRINT:FOR M=0 TO 1:PRINT TAB(3);C$(M):NEXT
3220 PRINT"▨▨";TAB(11);"<SPACE>"
3230 PRINT"▨";TAB(26);"QUADRANT ♦ 5-3"
3240 FOR M=1 TO 8:PRINT TAB(24);M;"........":NEXT
3250 PRINT"▨▨▨▨";TAB(28);"♦";"▨▨▨▨";"o"
3260 PRINT"▨";TAB(26);"SECTOR ♦ 7-6"
3270 FOR M=1 TO 8:PRINT TAB(24);M;"........":NEXT
3280 PRINT"▨▨";TAB(31);"♦"
3290 PRINT"▨";TAB(26);"12345678":PRINT:RETURN
3300 MUSIC M2$:PRINT D$(I);" DAMAGED"
3310 PRINT"**";D(I);"YEARS ESTIMATED FOR REPAIR.":GOTO 750
3400 IF D(1)<=0 THEN GOSUB 1900:GOTO 1030
3410 MUSIC M2$:PRINT"▨";D$(1);" DAMAGED":PRINT"**";
3415 PRINTD(1);" YEARS ESTIMATED FOR REPAIR."
3420 GOTO 1030
5000 PRINT"▨** STARTREK ** STARTREK ** STARTREK **"
5020 GOSUB 2100
5115 GOSUB 32001
5120 FOR I=1 TO 1000 :NEXT
5125 PRINT "▨":MUSIC MM$
5127 PRINT"▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨****************************"
5130 PRINT"▨              * GAME START *"
5140 PRINT"▨          ( PLEASE ENTER ANY KEY )"
5145 PRINT"▨▨▨▨▨▨▨▨▨****************************"
5150 TI$="000000"
5160 GET E$:II=LEN(E$):IF II=0 GOTO 5180
5170 GOTO 155
5180 VV=VAL(TI$):IF VV>60 GOTO 5000
5190 GOTO 5160
7000 PRINT"▨":PRINT TAB(33);"▨▨▨▨▨▨▨";G$;"▨"
7040 PRINT:PRINT:RETURN
7100 MUSIC M2$:PRINT"▨";"▨ STAR DESTROYED !!"
7110 PRINT TAB(15);" * AT SECTOR ";Y2+1;"-";X2+1
7200 PRINT"▨▨▨▨▨▨"
7210 PRINT"  ┌┌▀▀▀▀▀▀▀▀▀▀▀▀▀┐          ▁▁▄▄▄▄▀▀▀▁▁▁▁▁"
7220 PRINT"  └└▀▀   ▀▀▀▀▀▀┘     //▄▄█OO█████▒▒▒▒▒OO_\\\\"
7230 PRINT"      ▀▀▚      \\\\\\\\\\\_\_/\_/\_//////"
7240 PRINT"       ▀▚   ▀▀▚  █▀▄▀/\/ "
7250 PRINT"      ▁   ▀▚\▁▁▁▁▁/▁/▔\/\ "
7260 PRINT"    \\ ▀█████████▀▀▀▀▀▀▒▒▒▄▌█   ▙   ▌"
7270 PRINT"       ▁▀▒█ENTERPRISE █ <     ▀ \ "
7280 PRINT"         ▔\▁▁▁▁▁▁▁▄▀█ /  █    |  *"
```

```
7290 PRINT"   *            ▀──────┐  ◢  |"
7300 PRINT"          *       ───  ╲▀▄  ◤╱ "
7310 PRINT"  *                 ─╲──╱   *"
7320 PRINT"                  *"
7335 PRINT"            *"
7400 RETURN
20100 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣"
20200 FOR ID=0 TO 4*K
20300 PRINT TAB(25);"◣▣▣▣▣◤"
20510 PRINT"▣▣▣▣"
20600 PRINT TAB(25);"╲▣▣ ▣▣╱"
20810 PRINT"▣▣▣▣"
20900 NEXT ID
21000 RETURN
28010 CV=20:PRINT"▣▣▣▣▣▣DR.SPOCK"
28020 FOR CX=1 TO 5:SV=CV-CX
29000 PRINT"▣▣▣▣▣";
29100 PRINT TAB(SV);")    ▒▒▒▒ "
30000 PRINT TAB(SV);"   ╱ ▒▒▒  "
30010 PRINT TAB(SV);"(  ╱╱  ╱┤ ▒ "
30020 PRINT TAB(SV);" ╱ ┤ ╱▒▒ "
30030 PRINT TAB(SV);"▌┐  ╱▒▒▒ "
30040 PRINT TAB(SV);"└──┐ |▒▒▒▒ "
30050 PRINT TAB(SV);"▌╱‾          "
30060 PRINT TAB(SV);"  ◤╱◆    ╲"
30070 PRINT TAB(SV);"    ┤   ╱  ╲ "
30080 PRINT TAB(SV);"    ┤ ▣◣╱   ╲ "
30090 PRINT TAB(SV);"    ┤ | |    ╲"
30100 PRINT TAB(SV);"    ┤ ‾‾      ╲"
31010 PRINT TAB(SV);"   └───────── ╲ "
31020 PRINT"▣";TAB(SV);"   ▄▣ ▄▣  "
31030 NEXT CX
31040 TX=1:RETURN
32001 L1$="¯D4¯D0R¯D2R0¯D5A3¯#F4¯#F0R¯#F2R0¯#F5¯D3¯A5¯#F3¯A¯#F¯D"
32002 L2$="A4A0RA2R0A6":L3$="¯A4¯A0R0¯A2R0¯A6"
32003 L4$="A5¯D3A5¯#F3A5¯D3A5¯#F3A5¯#F3¯D¯#F¯AA4A0RA2R0A6"
32004 L5$="A4A0RA2R0A5#F3G4G0RG2R0G5E3#F4#F0R#F3AG#F"
32005 L6$="E6R":L7$="A4A0RA3¯D6"
32010 TEMPO 5:MUSIC L1$,L2$,L1$,L3$,L4$,L5$,L6$:RETURN
32101 F1$="G0AB4A3G2R0G3#FEDB5A3G¯CBA5D0GB4A3G2R0G3#FED"
32102 F2$="G5#F3DE#FG5B0¯C¯D4¯C3B¯C1¯D0B¯D1¯E¯D3B"
32103 F3$="¯E5¯D3B¯E1¯D¯CBA5D0G¯D4¯C3B¯C1¯D0B¯D1¯E¯D3B¯C5B3AG#FG5"
32110 TEMPO 1:MUSIC F1$,F2$,F3$,:TEMPO 5:RETURN
32201 U1$="G5":U2$="¯C8¯D2G¯D1¯E4¯E0R¯E4¯E0R¯E4¯E0R¯E2¯F¯C¯E6¯D3¯C5R"
32202 U3$="¯D3¯E2¯E0¯E4¯D1¯C3R¯D4¯E0R"
32203 U4$="¯E5¯D4¯E0R¯E5¯C4¯D0R¯D5A4A0RA5B"
32204 U5$="¯E5¯D4¯E0R¯E5¯D2¯E¯D1¯C5R7":U6$="¯C2G¯C1¯D4G0RG7"
32205 U6$="¯C2G¯C1¯D4G0RG7"
32206 U7$="¯C2G¯C1¯D4¯G0R¯G5¯G4¯D0R¯D5¯D4G0RG3G0RGRG3G0RGRG3G0RG1"
32210 TEMPO 3:MUSIC U1$,U2$,U3$,U4$,U2$,U3$,U5$,U6$,U6$,U7$,U2$,U3$,U5$
32220 TEMPO 5:RETURN
32300 FOR Z=1 TO 500:NEXT:PRINT"▣▣▣▣▣▣▣▣▣▣▣▣▣":FOR CY=1 TO 3
32310 FOR Z=1 TO 13:PRINT TAB(39);"▪";"▣▣";:NEXT:PRINT"▣"
32315 PRINT TAB(26);"▣▣"
32320 FOR Z=1 TO 13:PRINT TAB(39);"▪";"▣▣";:NEXT:PRINT"▣"
32330 NEXT CY:RETURN
```

MODIFICATIONS TO WORD-PROCESSOR WPI

by

P.L. BIRCH

If you are fortunate enough to possess a printer for your computer a word-processing program is an extremely useful addition to your software 'armoury'. Full scale commercial word-processors are, however, very expensive and probably much more sophisticated than is required for casual, non-business use. The WPI 'mini' word-processor marketed by Sharpsoft is a much simpler (and cheaper!) beast, but I have found it extremely useful for normal correspondence and such jobs as writing this short article. A further advantage is that, being written in Basic, the program may quite easily be amended to tailor it to one's own particular needs. As an inveterate tinkerer, I could not resist this temptation, and I thought that perhaps other 'User Notes' readers might be interested in some of my modifications and additions to WPI.

The major additions are a routine for formatting text to any desired number of characters/line (up to 79 maximum), and the ability to print repeat copies of a letter with the addition of names added from a tape file. Various other minor changes have also been made and I will detail these by going through the accompanying listing more or less in sequence. The listing includes only those lines which have been changed or added to the orginal program.

LINE 5

This loads the short machine-code routine at 1700-1730 which is used by the 'pause' facility dealt with later.

LINES 215/217

'xA' calls the new text formatting function. I couldn't think of a suitable mnemonic initial - A for 'Arrange' perhaps? 'xW' calls the routine for preparing files of names on tape.

LINES 310-316 & 430-436

These routines, in conjunction with the subroutines at 1700-1760 add a pause function to the screen listing and printing functions. Pressing the 'x' key will cause the listing or printing to stop. Key 'R' will return to the main program for a new command, repeated pressing of 'x' will single-step the listing, and any other key will cause normal listing to resume.

LINES 350-415

The 'title' function is a useful facility in WPI, but there are times when it is not required. These changes make it optional, the program prompting for a 'Y' or 'N' before printing.

LINES 450-455

This section provides a repeat printing facility. When requested for hard copy during the 'Print' function, the program will prompt for the number of copies required and print these automatically. Entering any non-numeric character in reply to this prompt will access the routine which enters names from a tape file (see next section).

LINES 480-495 & 1770-1960

Sending the same letter to a number of different people is a function which is very conveniently carried out with a word-processor. It is rather tedious to change the addressee's name manually and print out each letter individually. This routine avoids this by reading in the names sequentially from a tape file and inserting them into the appropriate position in the text. Text is prepared for this by using the symbol '@' (second key from right in bottom row with SML/CAP lock in operation) to indicate the position in text where the name is to be inserted. The same symbol is used to mark the start of the line in which the insertion is to be made. This avoids the need for the program to search every line of text. The first line of a letter might therefore be entered as '@Dear@,'. This would appear as (for instance) 'Dear Mr. Smith,' on printing. The '@' symbol is also used as an end of file marker by using it to terminate the last name in the tape file. Tape files may be prepared using the routine provided at line 1900, or generated by an external program. I use a mailing list program which prints address labels and generates the name file at the same time. The routine is called by entering a non-numeric character in response to the 'number of copies?' prompt in the 'Print' routine.

LINE 890

This enables text containing commas to be saved to, and loaded from, tape. Previously this caused truncation of the text at the comma. This is avoided by writing a double quote as the first character of each line of text.

LINES 1105-1107

This simply adds the new commands to the menu.

LINES 1330/1340

These minor changes correct the tendency to truncate lines which previously occurred when the '^F' function attempted to replace a string which was found more than once in the same line. I have also deleted line 1320 which prevents overflow to more than 79 characters per line as a result of replacement. I now permit this overflow to occur and correct the line length subsequently using the new '^A' command.

LINES 1400-1670

This is the text formatting routine, called by '^A'. It permits text to be rearranged to any number of characters/line (maximum 79) by shifting words between lines. Sections of text may be formatted by entering the start and finish line numbers. 'A' (All) and 'E' (End) may be used as in the 'list' and 'Print' commands. Lines with leading spaces are not merged with the preceding lines during formatting, so paragraphing is preserved.

Finally, a totally unrelated point which chanced to come to light during preparation of the listing for this article. The 'LIST/P' command in Sharp Basic causes a form feed character to be output to the Printer before the listing starts. This can be inconvenient and expensive in paper when listing short sections of code. POKE 15478,0 will disable this function and POKE 15478,13 will cause a line feed instead of a form feed. POKE 15478,15 restores normal functioning.

```
5 GOSUB1700

215 IFC$="A"THEN1400
217 IFC$="W"THEN1900

310 FORI=XTOY:PRINTI;"":A$(I)
312 P=0:USR(LT+10):IFPS<>0THEN1=PS
314 IFP1=124THENGOSUB1750
315 IFP1=82THEN70
316 NEXTI:GOTO70

350 INPUT"FROM ? ";X$:IFASC(X$)=65THENX=1:Y=L:GOTO393

380 IFASC(Z$)=69THENY=L:GOTO393

393 INPUT"INCLUDE TITLE ?";T$
396 T=0:IFASC(T$)=89THENT=1

410 IF<>ITHEN430
415 GOSUB920:PRINTI1$:PRINTI2$:PRINTI3$

430 FORI=XTOY:PRINTA$(I)
432 P1=0:USR(LT):PS=PEEK(LT+10):IFPS<>0THEN1=PS
434 IFP1=124THENGOSUB1750
435 IFP1=82THEN70
436 NEXTI:PRINT:PRINT

450 INPUT"N°. OF COPIES ?"NC$:IFVAL(NC$9=0THEN NC=1:RF=1:JF=0:GOTO452
451 NC=VAL(NC$):RF=0
452 FORP1=1TONC:IFT<>1THEN470
455 PRINT/P:PRINT/P:PRINT/P"■"OGOSUB920

480 FORI=XTOY:IF(RF=1)*(LEFT$(A(I),1="x")GOSUB1770
485 PRINT/PA$(I):IFRF=2THENA$(I)=A1$:RF=1
486 IFEF=1THENRF=0
487 NEXTI
490 PRINT/P:NEXTP1:IFRF<>0THENRF=1:GOTO452
495 GOTO70

890 FORX=ATOI:POKE6350,0:PRINT/TCHR$(34)+A$(X):POKE6350,34:NEXTX

1105 PRINT" A= FORMAT TEXT"
1107 PRINT" W= WRITE NAME FILE TO TAPE"

1330 IFRL=1THENA$(M)=A$:LC+LN-LO
1340 PRINTM:"";A$(M):LC=LC+LN:GOTO1270
```

```
1400 PRINT"@FORMAT TEXT"
1402 INPUT"CHARS./LINE (MAX 79) ?";CL
1404 IF(CL>79)+(CL<0) THEN1402
1410 INPUT"FROM ?";X$:IFASC(X$)=65THENX=1:Y=L:GOTO1460
1420 X=VAL(X$)
1430 INPUT "TO ? ";Z$
1440 IFASC(Z$)=69THENY=1:GOTO1460
1450 Y=VAL(Z$):IFY>LTHENPRINT"PAST END OF FILE":GOTO1430
1460 IFX>YTHEN70
1470 IFLEN(A$(X))=CLTHENX=X+1:GOTO1460
1480 IFLEN(A$(X))<CLTHEN1580
1490 FORI=LEN(A$(X))TO0STEP-1
1500 IF(MID$(A$(X),I,1)="")*(LEN(LEFT$(A$(X),I))<=CL)THEN1520
1510 NEXTI:PRINT"LINE";X;"TOO LONG-NO SPACES":PRINTA$(X):GOTO70
1520 A$=RIGHT$(A$(X),LEN(A$(X))-I):A$(X)=LEFT$(A$(X),I-1)
1530 X=X+1:IF(LEFT$(A(X),1)="")+(X=Y)THEN1550
1540 A$(X)=A$+""+A$(X):GOTO1460
1550 L=L+1:Y=Y+1
1560 FORI=LTOX+1STEP-1:A$(I)=A$(I-1):NEXTI
1570 A$(X)=A$:GOTO1460
1580 IF(LEFT$(A$(X+1),1)="")+(A$(X+1)="")THENX=X+1:GOTO1460
1590 IFLEN(A$(X)+A$(X+1))<CLTHENA$(X)=A$(X)+""+A$(X+1):GOTO1650
1600 FORI=1TOLEN(A$(X+1))
1610 IF(MID$(A$(X+1),I,1)="")*(LEN(A$(X)+LEFT$(A$(X+1),I-1))<CLTHEN163
1615 IFLEN(A$(X)+LEFT$(A$(X+1),I))>CLTHENX=X+1:GOTO1460
1620 NEXTI:X=X+1:GOTO1460
1630 A$(X)=A$(X)+""+LEFT$(A$(X+1),I,1)
1640 A$(X+1)=RIGHT$(A$(X+1),LEN(A$(X+1))-I):GOTO1460
1650 L=L-1:Y=Y-1
1660 FORI=X+1TOL:A$(I)=A$(I+1):NEXTI
1670 GOTO1460

1700 LT=53200:LIMIT LT
1710 FORI=0TO7:READ MC:POKELT+1,MC:NEXT
1720 DATA 205,27,0,50,218,207,201,0
1730 RETURN
1750 GETPS$:IFPS$=""THEN1750
1760 P1=ASC(PS$):RETURN

1770 EF=0:IFJF=0THENPRINT"INSERT NAMEFILE TAPE":ROPEN:JF=1
1780 INPUT/TNM$
1790 IFRIGHT$(NM$,1)="@"THENM$=LEFT$(NM$,LEN(NM$)-1):EF=1
1795 A1$=A$(I):RF=2
1800 A$(I)=RIGHT$(A$(I),LEN(A$(I))-1)
1810 FORJ=1TOLEN(A$(I))
1820 IFMID$(A$(I),J,1)="@"THEN1840
1830 NEXTJ:PRINT"@NOT FOUND IN LINE";I:RF=0:A$(I)=A1$:CLOSE:RETURN
1840 A$(I)=LEFT$(A$(I),J-1)+NM$+RIGHT$(A$(I),LEN(A$(I))-J)
1860 RETURN
1900 PRINT"@INSERT TAPE AND ENTER NAMES AS PROMPTED"
1910 PRINT"TERMINATE LAST NAME WITH '@'.":PRINT
1920 WOPEN"NAMEFILE"
1930 INPUT"ENTER NAME";NM$
1940 PRINT/TNM$
1950 IFRIGHT$(NM$,1)="@"THENCLOSE:GOTO70
1960 GOTO1930
```

Binomial Distribution

by

Paul Wood.

The "Binomial Distribution" program simulates an experiment in which marbles are rolled through a series of pins arranged in a triangle. At each pin the marble can either go to the left or the right. If the probability of these events is equal then the majority of the marbles will reach the central position at the bottom of the pins. The number of marbles on each side of this position decreases rapidly according to the binomial distribution curve: Thus the program visually shows how a binomial distribution is formed. This has many uses in education when teaching probability distributions.

Added features are:-

1. The facility to vary the probability of a marble going to the left or right. The two individual probabilities must, however, have a sum of one. This results in a binomial distribution offset from the central position.
2. The execution speed of the program can be varied. This allows a great many marbles to be rolled in a short time or at a slower the path of each marble can be followed.
3. The number of marbles can be varied - the higher the number of marbles the closer the experimental distribution is to the theoretical distribution.
4. The mean and standard deviation of the distribution are also displayed on the screen.

**** BINOMIAL DISTRIBUTION ****

```
0005 REM SET VARIABLES
0010 PRINT"■":DIM A(20):P=54009
0020 INPUT"Input probability of turning right: ";PR
0030 INPUT"■Input speed (F/S): ";SP$
0040 INPUT"■Input number of marbles: ";NM
0045 REM SETS PINS ON SCREEN
0050 PRINT"■"
0060 X=40:Y=2:N=19
0070 FOR T=1 TO 19
0080 X1=X:Y1=Y:FOR I=1 TO N
0090 SET X1,Y1
0100 X1=X1-2:Y1=Y1+2
0110 NEXT I
0120 X=X+2:Y=Y+2:N=N-1:NEXT T
0125 REM ROLLS MARBLES DOWN SCREEN
0130 PRINT"H↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓Number of marbles:-"
0140 FOR N=1 TO NM
0145 PRINT"■";N;"■■"
0150 PM=53268
0160 FOR I=1 TO 19
0170 POKE PM,72
0180 IF RND(1)>PR THEN MS=39:GOTO 200
0190 MS=41
0200 POKE PM,0:POKE PM+MS,72:PM=PM+MS
0210 IF SP$="S" THEN GOSUB 600
0220 NEXT I:I=INT((PM-P)/2):A(I)=A(I)+1
0230 NEXT N
0240 Z=0:FOR I=0 TO 20
0250 IF A(I)>Z THEN Z=A(I)
0260 NEXT I:SC=20/Z
0265 REM PLOTS GRAPH AXIS
0270 PRINT"■"
0280 FOR I=1 TO 21
0290 PRINT"|"
0300 NEXT I
0310 PRINT"0¯1¯2¯3¯4¯5¯6¯7¯8¯9¯0¯1¯2¯3¯4¯5¯6¯7¯8¯9"
0335 REM PLOTS GRAPH
0340 FOR I=0 TO 19:I1=I*2+80+P-1
0350 IF A(I)*SC=0 THEN 390
0360 FOR J=0 TO INT(A(I)*SC+0.5)
0370 POKE I1,239:I1=I1-40
0380 NEXT J
0390 NEXT I
0395 REM CALCULATES MEAN AND STD.DEVIATION
0400 H=0:S=0:SK=1000
0410 FOR I=0 TO 19
0420 H=H+A(I)*I:NEXT I
0430 AV=H/NM
0440 FOR I=0 TO 19
0450 S=S+A(I)*I*I:NEXT I
0460 S=S/NM-AV*AV
0470 AV=INT(SK*AV+0.5)/SK
0480 SD=INT(SK*SQR(S)+0.5)/SK
0490 PRINT"H↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓Mean =";AV;"    Std.Deviation =";SD;"↑

0500 GOTO 500
0600 FOR D=1 TO 200:NEXT D:RETURN
```

Bird Noises

by

Y. Shah.

This short contributed program allows your M2-80k to produce "bird noises" !

```
1 REM
2 REM   BIRD NOISES BY Y.SHAH
3 REM
4 PRINT"■"
5 PRINT"■
6 PRINT"                    /O  O\
7 PRINT"                    |    |
8 PRINT"                    |  <>|
9 PRINT"                     \__/
10 PRINT"                  /      \
11 PRINT"                 /  |  |  \
12 PRINT"                / /  \  \ \
13 PRINT"               /_/_/\_/\_\_\
14 PRINT"                    | |
15 PRINT"                    | |
16 PRINT"                    | |
17 PRINT"                    | |
18 PRINT"                    \_/
19 FORA=1TO1000:NEXT
20 POKE4514,1
21 FORA=1TO100:FORY=1TO100:NEXT
22 PRINT"■
23 PRINT"                   /   \
24 PRINT"                   | ^ |
25 PRINT"                   |/_\|
26 PRINT"                    \_/
27 PRINT"                  /     \
28 POKE4514,1:FORC=1TO8:FORB=255TO1STEP-20:POKE4513,B:USR(68):NEXTB,C
29 FORC=1TO4:FORB=255TO1STEP-6:POKE4513,B:USR(68):NEXTB,C
30 PRINT"■
31 PRINT"                   /■■■■\
32 PRINT"                   |    |
33 PRINT"                   |  <>|
34 PRINT"                    \__/
35 PRINT"
36 PRINT"                  /      \
37 USR(71):NEXTA
```

TINY PILOT

PILOT is a text oriented interactive language for use by teachers and children on small computers. Its simple syntax and free format encourage innovation and use by those frightened by computers or who lack time to learn a more complex language.

PILOT was developed in 1969 at the University of California Medical Center by John Starkweather to meet instructional needs.

TINY PILOT is a menu-driven programming language which includes the following options.

1. HELP - List the menu of options on the V.D.U.
2. LIST - List the stored PILOT program.
3. RUN - Run the PILOT program stored in memory.
4. CHANGE - Change one of the stored program lines.
5. NEW - Type in a new program - clearing the memory first.
6. SAVE - Save the PILOT program on cassette.
7. GET - Get the PILOT program from cassette - clearing the memory first.
8. INSERT - Insert a new line in the PILOT program stored in memory.
9. FIND - Find a string of characters in a PILOT program stored in memory.
10. DELETE - Delete a line in the PILOT program stored in memory.
11. EXIT - Exit to BASIC.
12. PRINTER - Turn the printer on or off.

After LOADing TINY PILOT the user menu will be displayed. Enter your selection command as requested; in full or the first letter followed by ⟨CR⟩.

TINY PILOT is a subset of the PILOT language and is intended as a programming system to introduce people to PILOT. A TINY PILOT program consists of a series of program statements, one per line. A maximum of 79 charaters is allowed on each line. A program may have a maximum of 255 lines. The TINY PILOT syntax is as follows :-

1. The statement layout.

⟨instruction⟩ ⟨modifier⟩ : ⟨operand⟩

where ⟨instruction⟩ is a single letter indicating the instruction type.

⟨modifier⟩ is an optional condition, which if false, causes this instruction to be ignored.

Two types of modifier are allowed :-

a. After a numeric variable has been defined in a compute (C) statement it may be tested against any number in a modifier which is enclosed by round brackets.

b. Inclusion of a Y or N after ⟨instruction⟩ allows a yes (true if flag set) or no (true if flag is reset) test following the last match statement.

2. Labels.

Labels have a * as their first character. Labels are referenced without the * in JUMP (J) instructions and subroutine jump (U) instructions.

3. TYPE (T).

The type statement prints on the V.D.U. the ⟨operand⟩ . Simple formatting is included.

a. accepted string variables, for example

```
T : Hello, what is your name?
A : N$
T : Hello N$
```

b. Screen clear if the first two characters of openand are ! H.

4. Accept (A).

The accept statement halts the program and prompts (outputs to the V.D.U. a ?) the user for an input. A string variable may be used with this statement, for example

```
A : P$
```

5. Match (M).

This statement searches the last accepted input for a match and sets the match flag, if a match is true. The special characters & and ! represent logical AND logical OR respectively.

6. Jump (J).

This statement causes a jump to the named label.

7. Subroutine jump (U).

This statement is similar to the jump statement allowing a jump to a labelled statement.

8. Subroutine return (E).

This statement is placed in a subroutine to return program control to the line following the last U statement.

9. Compute (C).

In TINY PILOT the compute statement has the following form:-

a. C : ⟨numeric variable⟩ = ⟨number⟩

b. C : ⟨numeric variable⟩ = ⟨same variable⟩ + ⟨number⟩

c. C : ⟨numeric variable⟩ = ⟨same variable⟩

Its main purpose is loop and modifier control.

10. Remark (R).

This statement is ignored by the TINY PILOT interpreter. Its main use is for comments in a program.

11 Quit (Q).

Inclusion of the quit statement in a TINY PILOT program will cause the program to halt at this statement - if executed.

The TINY PILOT listing is given at the end of these notes. The program, although written for the M2-80k, was based on the best of the ideas previously published in the popular computing magazines. We have attempted to test TINY PILOT for errors. However, with a program of this size it is impossible to trace all run-time bugs. If you use the program and find bugs or make modifications please write to SHARPSOFT user notes and we will publish your comments in a future edition.

```
1 REM************ Tiny Pilot ************
2 REM ****Copyright SHARPSOFT Ltd.****
30 PRINT"@":PRINT:PRINT
70 DIMP$(255),L$(20),L(20)
80 H$="TMARJEUCQ"
90 DIMQ(20),Q$(20),S(10)
95 PP=0
100 DIMV$(20)
105 FORX=1TO40:PRINT"▒";:NEXT
110 PRINT"▒          << Tiny Pilot >>              ▒";
120 FORX=1TO40:PRINT"▒";:NEXT:PRINT:PRINT
130 GOTO3010
200 INPUT"Command ? ";C$
250 C$=LEFT$(C$,1)
260 IFC$="N"THEN1000
265 IFC$="D"THEN4000
270 IFC$="L"THEN2000
275 IFC$="I"THEN6000
280 IFC$="C"THEN5000
285 IFC$="F"THEN7000
290 IFC$="R"THEN8000
295 IFC$="H"THEN3000
297 IFC$="E"THENEND
300 IFC$="S"THEN20000
310 IFC$="G"THEN20100
350 IFC$="P"THEN30000
800 GOTO200
1000 PRINT"@NEW"
1010 INPUT"Are you sure ? ";C$
1020 IFLEFT$(C$,1)="Y"THEN1040
1030 GOTO200
1040 P=1:P1=0
1050 GOSUB1500
1070 P$(P)=E$
1080 IFC$="×"THEN200
1090 IFLEN(E$)=1THEN1050
1100 P=P+1
1110 GOTO1050
1500 INPUT" > ";E$
1510 IFE$="←"THENRETURN
1520 C$=LEFT$(E$,1)
1530 IFC$="×"THENP1=P:RETURN
1540 FORJ=1TOLEN(E$)
1550 IFMID$(E$,J,1)=":"THEN1610
1560 NEXTJ
1570 IFLEN(E$)=1THEN1660
1580 IFC$="*"THENRETURN
1590 PRINT"——> Syntax error"
1600 GOTO1500
1610 FORJ=1TO9
1620 IFC$=MID$(H$,J,1)THENRETURN
1630 NEXTJ
1640 PRINT"——> Syntax error"
1650 GOTO1500
1660 PRINT"——> Syntax error"
1670 RETURN
2000 PRINT"@LIST"
2010 INPUT"From ? ";X$:IFASC(X$)=65THENX=1:Y=P1:GOTO2060
```

```
2020 X=VAL(X$)
2030 INPUT"To ? ";Z$
2040 IFASC(Z$)=69THENY=P1:GOTO2060
2050 Y=VAL(Z$):IFY>P1THENPRINT"Past end of program":GOTO2030
2060 IFPP=1THEN PRINT#P"<< Tiny Pilot >>":PRINT#P
2065 PRINT:PRINT"<< Tiny Pilot >>":PRINT
2070 FORI=XTOY:PRINTI:" > ";P$(I)
2075 IFPP=1THENPRINT#PI:" > ";P$(I)
2080 NEXTI:GOTO200
3000 PRINT"@          <<  Tiny Pilot  >>"
3005 PRINT
3010 PRINT"            Commands"
3020 PRINT"              are"
3025 PRINT
3030 PRINT"            H    =    HELP"
3040 PRINT"            L    =    LIST"
3050 PRINT"            R    =    RUN"
3060 PRINT"            C    =    CHANGE"
3070 PRINT"            N    =    NEW"
3080 PRINT"            S    =    SAVE"
3090 PRINT"            G    =    GET"
3100 PRINT"            I    =    INSERT"
3110 PRINT"            F    =    FIND"
3120 PRINT"            D    =    DELETE"
3130 PRINT"            E    =    EXIT"
3140 PRINT"            P    =    PRINTER"
3499 PRINT:PRINT:PRINT
3500 GOTO200
4000 PRINT"@DELETE"
4010 INPUT"Line number ? ";X:IFX>P1THENPRINT"Past end of program":GOTO4010
4020 PRINT"DELETE TEXT ---->"
4030 PRINTSPC(14);"+":PRINT
4040 PRINTP$(X):PRINT
4050 INPUT"  OK ? ";A$:IFASC(A$)<>89THEN200
4060 P1=P1-1:FORV=XTOP1:P$(V)=P$(V+1):NEXTV:GOTO200
5000 PRINT"@CHANGE"
5010 INPUT"Line number ---> ";CL
5020 PRINT:PRINT:PRINTP$(CL):PRINT
5030 PRINT"Enter new line"
5035 PRINT:PRINT:PRINT
5040 GOSUB1500
5050 P$(CL)=E$
5060 PRINT:PRINT:PRINT
5110 INPUT"Change another line ? ";C$
5120 IFASC(C$)=89THENGOTO5000
5130 GOTO200
6000 PRINT"@INSERT"
6010 INPUT"After line number ? ";X
6015 IFX>P1THENPRINT"Past end of program ":GOTO6010
6020 PRINT"New text ---->"
6030 PRINTSPC(11);"+":PRINT
6040 GOSUB1500
6050 P1=P1+1:FORV=P1TOX+2STEP-1:P$(V)=P$(V-1):NEXTV
6060 P$(X+1)=E$:GOTO200
7000 PRINT"@FIND"
7010 INPUT"OLD TEXT ? ";O$:LO=LEN(O$)
7020 INPUT"NEW TEXT ? ";N$:LN=LEN(N$)
7030 INPUT"From ? ";X$:IFASC(X$)=65THENX=1:Y=P1:GOTO7080
```

```
7040 X=VAL(X$)
7050 INPUT"To   ? ";Z$
7060 IFASC(Z$)=69THENY=P1:GOTO7080
7070 Y=VAL(Z$):IFY>P1THENPRINT"Past end of program":GOTO7050
7080 RL=0:INPUT"REPLACE ? ";X$:IF ASC(X$)=89THENRL=1
7085 IFPP=1THENPRINT/P"<< Tiny Pilot >>":PRINT/P
7086 PRINT"<< Tiny Pilot >>":PRINT
7090 FORM=XTOY:LS=LEN(P$(M)):LC=0
7100 FORN=1TOLO:P=LC+N:IFMID$(O$,N,1)=MID$(P$(M),P,1)THEN7130
7110 LC=LC+1:IFLC+LO<=LSTHEN7100
7120 NEXTM:GOTO200
7130 NEXTN
7140 A$=MID$(P$(M),1,LC)+N$+MID$(P$(M),LC+LO+1,LS-LC-LO)
7150 IFLEN(A$)>79THENPRINT"LINE ";M;" TOO LONG":GOTO7120
7160 IFRL=1THENP$(M)=A$
7165 IFPP=1THENPRINT/PM;" > ";P$(M)
7170 PRINTM;" > ";P$(M):LC=LC+1:GOTO7100
8000 PRINT"RUN"
8010 L1=0
8100 FORP=1TOP1
8110 IFLEFT$(P$(P),1)="*"THEN8130
8120 GOTO8160
8130 L1=L1+1
8140 L$(L1)=MID$(P$(P),2,LEN(P$(P)))
8150 L(L1)=P
8160 NEXTP
8200 V1=0:P=0:M=0
8210 S1=0:Q1=0
8230 IFP1=0THENPRINT"No program":GOTO200
8300 P=P+1
8320 C$=LEFT$(P$(P),1)
8330 IFC$="."THENPRINT"End of program":GOTO200
8340 IF(C$="*")+(C$="R")THEN8300
8400 IF(MID$(P$(P),2,1)="Y")*(M=0)THEN8300
8410 IF(MID$(P$(P),2,1)="N")*(M=1)THEN8300
8420 FORI=1TOLEN(P$(P))
8430 IFMID$(P$(P),I,1)=":"THEN8500
8440 NEXTI
8450 GOTO8300
8500 FORJ=1TOI
8510 IFMID$(P$(P),J,1)="("THEN8600
8520 NEXTJ
8530 GOTO8900
8600 FORK=JTOI
8610 D$=MID$(P$(P),K,1)
8620 IF(D$="=")+(D$="<")+(D$=">")THEN8650
8630 NEXTK
8640 PRINT"Syntax error, LINE ";P:GOTO200
8650 IFQ1=0THENPRINT"Variable error, LINE ";P:GOTO200
8700 FORN=1TOQ1
8710 IFMID$(P$(P),J+1,K-J-1)=Q$(N)THEN8800
8720 NEXTN
8730 PRINT"Variable error, LINE ";P:GOTO200
8800 J=VAL(MID$(P$(P),K+1,I-K-2))
8810 IFD$=">"THEN8830
8820 GOTO8850
8830 IFQ(N)>JTHEN8900
8840 GOTO8300
```

```
8850 IFD$="<"THEN8870
8860 GOTO8890
8870 IFQ(N)<JTHEN8900
8880 GOTO8300
8890 IFQ(N)=JTHEN8900
8895 GOTO8300
8900 D$=MID$(P$(P),I+1,LEN(P$(P)))
8905 IFC$="T"THENGOSUB11000:GOTO8300
8910 IFC$="M"THENGOSUB12000:GOTO8300
8920 IFC$="A"THENGOSUB13000:GOTO8300
8930 IFC$="J"THENGOSUB14000:GOTO8300
8940 IFC$="U"THENGOSUB15000:GOTO8300
8950 IFC$="E"THENGOSUB16000:GOTO8300
8960 IFC$="Q"THENPRINT:PRINT:PRINT:PRINT"End of program":GOTO200
8970 IFC$="C"THENGOSUB17000:GOTO8300
9000 PRINT"Error in line ":P
9030 GOTO200
11000 IFLEN(D$)<2THEN11310
11010 IFLEFT$(D$,2)="!H"THEN PRINT"■"
11020 IFLEN(D$)=2THENRETURN
11030 D$=MID$(D$,1,LEN(D$))
11100 I=1
11110 I=I+1
11120 IFV1=0THEN11300
11122 IFMID$(D$,I,1)="$"THEN11130
11124 GOTO11300
11130 FORJ=1TOV1
11140 IFLEFT$(V$(J),2)=MID$(D$,I-1,2)THEN11160
11145 GOTO11300
11160 IFLEN(D$)<>2THEN11180
11170 D$=E$:GOTO11200
11180 IFI<>2THEN11200
11190 D$=E$+MID$(D$,I+1):GOTO11300
11200 IFLEN(D$)<>ITHEN11220
11210 D$=LEFT$(D$,I-2 )+E$:GOTO11300
11220 E1$=LEFT$(D$,I-2)+E$
11230 D$=E1$+MID$(D$,I+1,LEN(D$))
11240 GOTO11300
11250 I=I+LEN(V$(J))-2
11260 GOTO11300
11270 NEXTJ
11300 IFI<=LEN(D$)THEN11110
11310 PRINT D$
11320 RETURN
12000 M=0:M1=0:M2=0
12010 GOSUB12200
12020 FORI=1TOLEN(D$)
12030 IFMID$(D$,I,1)="!"THEN12040
12035 GOTO12070
12040 GOSUB12500
12050 IFM=1THENRETURN
12060 M1=I
12070 NEXTI
12080 GOSUB12500
12090 RETURN
12200 IFLEN(D$)<2THENRETURN
12210 D1=0:I=0
```

```
12220 I=I+1
12230 IFMID$(D$,I,2)="  "THEN12250
12240 GOTO12290
12250 D1=1
12260 IFI>1THEN12280
12270 D$=MID$(D$,2,LEN(D$)):GOTO12220
12280 D$=LEFT$(D$,I-1)+MID$(D$,I+1,LEN(D$))
12290 IFI<>LEN(D$)THEN12220
12300 IFD1=1THEN12210
12310 IFLEFT$(D$,1)=" "THEN12330
12320 GOTO12340
12330 D$=MID$(D$,2,LEN(D$))
12340 IFRIGHT$(D$,1)=" "THEN12360
12350 RETURN
12360 D$=LEFT$(D$,LEN(D$)-1)
12370 RETURN
12500 M3=0:M2=M1
12510 FORJ=M1+1TOI-1
12520 IFMID$(D$,J,1)="&"THEN12530
12525 GOTO12560
12530 GOSUB12700
12540 M2=J
12550 IFM=0THENRETURN
12560 NEXTJ
12565 GOSUB12700
12570 RETURN
12700 FORK=M3+1TOLEN(A$)-J+M2+2
12710 IFMID$(D$,M2+1,J-M2-1)=MID$(A$,K,J-M2-1)THEN12730
12720 GOTO12750
12730 M=1
12740 M3=K+J-M2-2
12745 RETURN
12750 NEXTK
12760 M=0
12770 RETURN
13000 INPUT"? ";E$
13010 IFD$=""THEN13030
13020 GOTO13050
13030 A$=E$
13040 RETURN
13050 Z$=D$:D$=E$
13060 GOSUB12200
13070 A$=D$:D$=Z$
13080 IFV1=0THEN13125
13090 IFLEN(D$)=2THEN13100
13095 IFRIGHT$(D$,1)="x"THEN13100
13097 PRINT"Variable error, LINE ";P:RETURN
13100 FORI=1TOV1
13110 IFD$=V$(I)THENPRINT"Duplicate variable error, LINE ";P;"and ";I:RETURN
13120 NEXTI
13125 V1=V1+1
13130 V$(V1)=D$+A$
13140 RETURN
14000 GOSUB12200
14010 FORI=1TOL1
14020 IFD$=L$(I)THEN14050
```

```
14030 NEXTI
14040 PRINT"No label error, LINE ";P:RETURN
14050 P=L(I)
14060 RETURN
15000 S1=S1+1:S(S1)=P:GOSUB14000:RETURN
16000 P=S(S1):S1=S1-1
16010 IFS1<0THENPRINT"Subroutine return error, LINE ";P:RETURN
16020 RETURN
17000 IFLEN(D$)<2THENPRINT"Syntax error, LINE ";P:RETURN
17010 FORI=1TOLEN(D$)
17020 IFMID$(D$,I,1)="="THEN17050
17030 NEXTI
17040 PRINT"Syntax error, LINE ";P:RETURN
17050 IFQ1=0THEN17090
17060 FORJ=1TOQ1
17070 IFLEFT$(D$,I-1)=Q$(J)THEN17120
17080 NEXTJ
17090 Q1=Q+1
17100 J=Q1
17110 Q$(J)=LEFT$(D$,I-1)
17120 FORK=I+1TOLEN(D$)
17130 IF(MID$(D$,K,1)="+")+(MID$(D$,K,1)="-")THEN17200
17140 NEXTK
17150 IFI=LEN(D$)THENPRINT"Syntax error, LINE ";P:RETURN
17160 Q(J)=VAL(MID$(D$,I+1,LEN(D$)))
17170 RETURN
17200 Q(J)=Q(J)+VAL(MID$(D$,K,LEN(D$)))
17210 RETURN
20000 PRINT"@Save file"
20010 INPUT"File name ? ";X$:A$=X$+",PLT"
20020 WOPEN A$
20030 FORX=1TOP1:PRINT/TP$(X):NEXTX
20040 A$="x":PRINT/TA$
20050 CLOSE:GOTO200
20100 PRINT"@Get file"
20110 FORX=1TO255:P$(X)=""
20120 INPUT"File name ? ";X$:A$=X$+",PLT"
20130 ROPENA$:L=1
20140 INPUT/TA$:IFLEFT$(A$,1)="x"THENCLOSE:P1=L:P$(L)="x":GOTO200
20150 P$(L)=A$:L=L+1:PRINTA$:GOTO20140
30000 PRINT"@PRINTER"
30010 INPUT"Printer on ? ";X$
30020 IFASC(X$)=89THENPP=1:GOTO200
30030 PP=0:GOTO200
```

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