BBC Micro Programs in BASIC

Derrick Daines

Newnes Technical Books

Newnes Technical Books
is an imprint of the Butterworth Group
which has principal offices in
London, Boston, Durban, Singapore, Sydney, Toronto, Wellington

First published in 1984

British Library Cataloguing in Publication Data

Daines, Derrick
BBC Micro programs in BASIC
1. Electronic games 2. BBC Microcomputer
--Programming 3. Basic (Computer program language)
I. Title
794.8'028'5424 GV1469.2
ISBN 0-408-01415-6

Library of Congress Cataloging in Publication Data

Daines, Derrick
BBC Micro programs in BASIC
1. BBC Microcomputer--Programming 2. Basic
(Computer program language) I. Title II. Title:
B.B.C. Micro programs in B.A.S.I.C.
QA76.8.B35D35 1984 001.64'.2 83-23712

ISBN 0-408-01415-6

Photoset by Butterworths Litho Preparation Department
Printed in England by Whitstable Litho Ltd, Whitstable, Kent

Contents

The publishers gratefully acknowledge the assistance of Pace Software Supplies Ltd, X-Data Ltd and Cumana Ltd in the preparation of this book.

1

Introduction

Anyone can write a long program; what is really difficult is to write an interesting program in only one line. That is why I make no apologies for opening this book with what may well be the shortest games program on record. Oh, it really works - and it's interesting, too, for it times your reactions to the nearest one-hundredth of a second. Try it at any time, especially when you've had a few drinks! When prompted, tap any key as fast as you can.

Here is the entire listing:

```
F.X=1TORND(1000):N.:P."GO!":TI.=0:X$=GE.:P.TI./100;" secs."
```

One is able to write like that because Acorn have provided the BBC Microcomputer with two interesting and useful features that affect us both - myself as the writer and you as the reader of this book. The first is that any number of statements may be placed in one multi-command line, extending to no less than six print lines on the screen, with automatic wrap-around taking place at the end of the line. This means that in many published listings there are some fairly heavy-looking blocks of text to be copied into your computer. I have tried to avoid that, but here and there this avoidance in fact lengthens the listing because particular logic structures cannot be used. Therefore, although expert readers will be able to shorten the listings a little; neophytes will find them easier to read and understand.

The second point is that the BBC computer allows shortened forms of command, like 'P.' for 'PRINT', 'E.' for 'ENDPROC' and so on. It is well worth your while knowing these, for they can cut your typing time enormously. They are listed in the User Guide.

If you wish to check the single-line program above, the following is the extended form; at least it allows you to follow more easily

what the program does. Line 10 ensures a random start, eliminating anticipation.

```
10 FOR X=1 TO RND(1000):NEXT
20 PRINT"GO!!"
30 TIME=0
40 X$=GET$
50 PRINT "You took ";TIME/100;" seconds"
```

Of course, the game is not perfect, and we ought to include an *FX15,1 command at the end of line 30, but at least it illustrates the point: that writers have to steer a narrow course between on the one hand making listings so dense that they become impossible to understand - and easy to lose your way in - and on the other hand making them so spread out that the book is soon filled and the reader does not get full value for money.

There is another point of particular interest to owners of OS versions 1.0 and up: the use of the colour graphics commands in Mode 7. By holding down the SHIFT key and pressing the Function Keys f1 to f9, various print colours are obtained. Unfortunately this cannot be shown in printed form, as the single byte thus stored in the program creates havoc when received by printers - usually a graphics symbol is printed, but anything goes. For this reason, these commands are shown in full as PRINT CHR$(129), etc. Readers may make a considerable saving in typing time and effort by substituting:

CHR$129	SHIFT-f1	(Read text)
CHR$130	SHIFT-f2	(Green text)
CHR$131	SHIFT-f3	(Yellow text)
CHR$132	SHIFT-f4	(Blue text)
CHR$133	SHIFT-f5	(Magenta text)
CHR$134	SHIFT-f6	(Cyan text)
CHR$135	SHIFT-f7	(White text)
CHR$136	SHIFT-f8	(Flashing text)
CHR$137	SHIFT-f9	(Steady text)

Another very welcome feature of the BBC machine is that it supports procedures. This not only encourages good programming practice, but enables sections of the program to be tested and proved in isolation. As an example of this, on page 4 are listed a number of interesting and useful procedures that are used many times in this book. It is recommended that the reader types and saves them once only - on disk or tape - and reloads them when starting to copy a new program. The method is:

(a) Copy the procedures exactly as printed
(b) Type *SPOOL PROCS (Return)
(c) If using tape, start recording
(d) Type LIST (Return)
(e) When list is finished, type *SPOOL (Return).

The procedures are then stored on tape or disk in ASCII format (not BASIC). When starting the copy of a new listing from this book, rewind the tape and type:

*EXEC PROCS (Return)

Start the tape and the procedures will be loaded in. If desired, the procedures could be SAVEd and LOADed as a BASIC program, but the advantage of doing it in the way outlined is that they can be added to an existing program at any time, without destroying what is already in memory.

PROCTITLE
The procedure prints a title in double height, with a decorative design above and below. The design is in blue, with the title yellow. The only parameter that needs passing to it is the title X$, with the procedure automatically placing it in the centre. Please note that the procedure works in Mode 7 only; indeed, the calling program might include a MODE 7 command, which could clear the screen and prepare for the title.

PROCDBL
This procedure is in the User Guide and I have altered it slightly so that the position on the screen is determinable - the X% and Y% values passed - as is the colour C%. The text to be printed in double height is passed in X$.

PROCBOX
A longish procedure, but enormously useful. It will print a box of any size, in any colour, in any position on the screen, outlining it with a strong line. In addition, it will if desired print a number of white dots within the box to indicate the maximum number of input characters allowable, and accept input, overprinting the dots. Deletion is allowed up to the time when RETURN is pressed, while if the user attempts to enter more characters than the program allows, the ultimate character is altered. In other words, the user cannot overflow the box.

Mode 7 graphics are used. The parameters passed are:

X% Horizontal tab position of left edge
Y% Vertical tab of bottom edge minus 1
L% The length: the number of character spaces allowed inside
H% The height: the number of lines allowed inside
C% Colour: from 145 to 151 for white to read
F% Flag, where 0 = input desired, 1 = no input

On exit, X$ holds the input line, if any, and this will be tested by the calling routine.

```
10000 DEFPROCTITLE(X$)
10010 PRINTCHR$132;STRING$(19,"Oo")
10020 PROCDBL((36-LEN(X$))/2,3,131,X$)
10030 PRINT'CHR$132;STRING$(19,"Oo")
10040 ENDPROC
10050
10060 DEFPROCDBL(X%,Y%,C%,X$)
10070 PRINTTAB(X%,Y%);CHR$141;CHR$C%;X$
10080 PRINTTAB(X%,Y%+1);CHR$141;CHR$C%;X$
10090 ENDPROC
10100
10110 DEFPROCBOX(X%,Y%,L%,H%,C%,F%)
10120 LOCALV%,W%,I%,J%:REM - MODE7 ONLY
10130 PRINTTAB(X%,Y%);
10140 V%=VPOS:W%=POS:PRINTTAB(W%,V%-H%);CHR$C%;"7";
10150 FORI%=0TOL%+1:PRINT"£";:NEXT:PRINT"k"
10160 PRINTTAB(W%,V%+1);CHR$C%;"u";
10170 FORI%=0TOL%+1:PRINT"p";:NEXT:PRINT"z"
10180 FORJ%=V%-H%+1TOV%:PRINTTAB(W%,J%);CHR$C%;"5":NEXT
10190 FORJ%=V%-H%+1TOV%
10200 PRINTTAB(W%+L%+3,J%);CHR$C%;"j":NEXT
10210 IF F%GOTO10320
10220 PRINTTAB(W%+2,V%);CHR$135;
10230 FORI%=1TOL%:PRINT".";:NEXT
10240 PRINTTAB(W%+3,V%);:X$=""
10250 G$=GET$:IF ASCG$=13 GOTO10320
10260 IF LENX$=L%OR ASCG$=127 GOTO10280
10270 PRINT G$;:X$=X$+G$:GOTO10250
10280 IF X$="" GOTO10250
10290 X$=LEFT$(X$,LENX$-1):PRINTCHR$8;
10300 IF ASCG$<>127 GOTO10270
10310 PRINT".";CHR$8;:GOTO10250
10320 ENDPROC
10330
10340 DEFPROCRET
10350 PRINTTAB(5,19);CHR$131;"Press";
10360 PRINTCHR$132;CHR$157;CHR$129;"RETURN ";CHR$156;
10370 G$=GET$:CLS:ENDPROC
10380
10390 DEFPROCWARBLE
10400 FORS%=1TO20:SOUND1,-12,30,1
10410 SOUND1,-12,100,1:NEXT:ENDPROC
10420
10430 DEFPROCBOING
10440 SOUND 0,-15,80,2:FOR S%=-15 TO 0
10450 SOUND1,S%,20+S%,2:NEXT:ENDPROC
```

PROCRET

The instruction to the user, 'Press RETURN' is so often used that it is well worth while having a little procedure to give it and to wait until a key is pressed. This procedure prints a very attractive and eye-

catching instruction, with ' Press'in yellow, and ' RETURN'in red on a dark blue background. No parameters are passed to the procedure, but the TAB position may require alteration to suit different circumstances.

PROCWARBLE
A little sound routine used as an audible signal that the user has won a game or done something good.

PROCBOING
Another little sound for the opposite effect. It sounds rather like a recalcitrant sofa spring and nobody can doubt its message!

Close examination of the above procedures in various parts of the book will reveal small but significant differences here and there. This is because I have taken the view that readers interested in only one or two programs will not wish to duplicate these procedures as outlined above and so will not need some of their features; PROCBOX is a case in point. In the individual listings, therefore, I have included only those features needed by that particular program. If you copy the listings as shown you may be confident that they will run correctly. On the other hand, if you follow the method outlined above, re-using the saved procedures, the programs will still run correctly, although features will be included in a particular program that may not be used within it.

A good illustration of this is PROCBOX, which in its complete form allows the user to type an input within the box, the length of input being indicated by the number of white dots. Many of the listed programs do not require this facility, which (together with the necessary ability to delete or correct an input before RETURN) makes PROCBOX rather long. The listings not requiring an input of this nature therefore do not show this part of PROCBOX.

In the pursuit of readability, I have spread out the listings a little by the insertion of empty lines. These show a line number only. Empty lines may cause a little bewilderment to newcomers, because of course if you type a line number only, followed by RETURN, nothing is recorded. Moreover, if there was a line with that number, it is now erased! In fact, the empty lines are easy to achieve: if you wish to incorporate them into your own programs for the purpose of breaking up the listings and making them easier to read, the trick is to type the line number and then one or more spaces, before pressing RETURN.

One other trick to aid readability - which unfortunately we cannot show in printed form - is to incorporate colour into REM statements. If you do this, note that the colour byte mentioned earlier MUST be preceded by inverted commas:

```
250 REM " This remark is in colour
```

where the CHR$131 byte (or whatever) follows the inverted commas. It is also worth noting that, if a line is never processed during run time, the REM is not needed, so that

```
250 " This remark can be in colour
260 " and so can this, as long as they are not processed.
```

are perfectly OK and will always show in colour when the program is listed. Such lines are very usefully placed before procedures in long programs and allow you to scan a listing swiftly. For obvious reasons, these tricks have not been incorporated in the current listings.

Tired of all that typing? All of the programs listed in this book are available on cassette from:

Leasalink Viewdata Ltd
Electron House
Bridge Street
Sandiacre
Nottingham NG10 5BA

Leasalink are also main distributors for the BBC computer.