EXPANDED COLOR BASIC

for the (Disk) Extended Color BASIC TRS-80 Color Computer

CONTENTS

SECTION 0 - INTRODUCTION

Chapter 1. A Short Summary

Expanded Color BASIC is a language designed to be added on to Microsoft's Extended Color BASIC (or Disk BASIC if you have disk). Although Extended BASIC is a pretty powerful language, it is deficient in a number of respects; Expanded BASIC makes good these deficiencies. The new features of Expanded BASIC include:

Printing text in all graphic modes and colors,
A special 51x24 sized-screen for PMODE4,
Scrolling of any section of any screen in any direction,
Extra colors in a new mode,
Borders for the text screen,
Extra graphic pages,
Sound effects,
REPEAT...UNTIL loops,
Multiline IF...THEN...ELSE statements,
Procedures as a better alternative to subroutines,
Local variables,
Full ON-ERROR implementation,
Break disabling,
Autoline numbering,
On-screen copying,
User-definable printer widths,
Single key entry of most BASIC words,
Function keys and finally,
The ability to execute strings as commands.

Chapter 2. About this Manual

The purpose of this manual is to guide you in the application of Expanded Color BASIC. It assumes a knowledge of Microsoft's BASIC . If you are not familiar with that, you are first advised to read the Radio Shack Manuals, "Getting Started with Color BASIC" and "Going Ahead with Extended BASIC", before attempting to start on Expanded BASIC.

The following terms will be used throughout this manual:
1) Command. This instructs the computer to perform something.
e .g. PRINT, INPUT, SOUND, GOTO etc. are all commands.
2) Function . This is much like a command except that the computer supplies an answer.
e .g. SIN, LOG, MID$, LEN, RND etc. are all functions.
3) Number. This is simply a number!
e.g. 1, 1000, -.734 etc. are all numbers.
4) Numeric variable. This uses a group of letters to represent a number.
e.g. X, ZZ, ACTIVITY etc. are all variables.
5) Numeric function. This returns a result which is a number.
e.g. SIN, ATN, INT, INSTR, LEN etc. are all numeric functions.
6) Operator. This is one of the set +, -, *, /, ^, AND, OR and NOT.
7) Numeric expression . This is either a number, numeric variable, numeric function or a mixture thereof.
e.g. 3, X+Y, SIN(2)-INT(F/2)*LOG(T^5), LEN(A$) etc. are all numeric expressions.
8) String constant. This is a sequence of characters within quotation marks.
e.g. "HELP", "This is a constant" are string constants.
9) String variable. This a group of letters (suffixed by $) which represents a string.
e.g. A$, L1$, GRAPH$ etc. are all string variables.

10) String function. This is a function that returns a string result.
e.g. MID$, LEFT$, RIGHT$, INKEY$, HEX$, STR$ etc . are string functions.
11) String expression. This is either a string constant, string variable, string function or a mixture thereof.
e.g. "HELLO", MID$(A$,2,3)+LEFT$(EX$,INT(S/2)) are string expressions.

The following abbreviations recur throughout :
1> EXP. Expanded Color BASIC.
2> EXPTAPE. Expanded Color BASIC system tape.
3> EXPDISK. Expanded Color BASIC system disk.
4> CoCo#1. TRS-80 Color Computer 1.
5> CoCo#2. TRS-80 Color Computer 2.

SECTION 1 - GRAPHIC/TEXT COMMANDS

Chapter 3 - Meet Expanded BASIC

To load EXP, firstly switch on your computer. IF IT IS ALREADY ON , SWITCH IT OFF AND TURN IT ON AGAIN. Obey the following instructions exactly as described:

If you have disk
---
1) Insert EXPDISK into drive 0.
2) If you have a CoCo#1 then skip to step 4.
3) Enter RUN"R". When "R" is fully run, the following message should appear on the screen:
DISK EXTENDED COLOR BASIC 1.0
COPYRIGHT (C) 1980 BY TANDY
UNDER LICENSE FROM MICROSOFT
4) Enter RUN"G" . When the program has fully run the following message should appear on the PMODE4 graphic screen:
DISK EXPANDED COLOR BASIC 1.3
COPYRIGHT 1984 BY T.DELBOURGO
INCLUDES GRAPHIC/TEXT COMMANDS

If you have tape
---
1) Put EXPTAPE into tape recorder.
2) Position tape to beginning of "R" program. (Look at the counter numbers on the label).
3) Enter CLOAD and press <PLAY>. When the 'OK' prompt reappears press <STOP>.
4) Enter RUN and press <PLAY>. When the following message appears, press <STOP>:
EXTENDED COLOR BASIC 1.0
COPYRIGHT (C) 1980 BY TANDY
UNDER LICENSE FROM MICROSOFT
5) Position tape to beginning of "G" program.
6) Enter CLOAD and press <PLAY>. When the 'OK' prompt appears press <STOP>, enter RUN and press <PLAY> again.
7) After the following message appears, press <STOP>:
EXPANDED COLOR BASIC 1.3
COPYRIGHT 1984 BY T.DELBOURGO
INCLUDES GRAPHIC/TEXT COMMANDS

EXP is now loaded. At this point, stored in memory are: Color BASIC 1.1, Extended Color BASIC 1.0 and, if you have disk, Disk Extended Color BASIC 1.0. This applies regardless of what ROM memory your computer began with.

Text is being written on the graphic screen at this stage. Enter TEXTON and you will find yourself back on the text screen. (Note that TEXTON clears the text screen). To get back to the graphic screen enter TEXTOFF. (TEXTOFF clears the graphic screen) . Enter CLS and you will find that the graphic screen goes clear. Enter PCLS and the graphic screen will also clear , but , as well, the cursor will stay put. Press <SHIFT>/<0> to go into the lowercase mode and type in a few characters. Notice that true lower-case appears -- not reverse video upper-case!

Press <SHIFT>/<Ø> again to return to the uppercase mode.

Text in different COLORs and PMODEs.

Go into PMODE1 by entering PMODE1. You may change the color of the text by using the COLOR command. Enter COLOR4,2 and you will see red text on a yellow background. Go into color-set 1 by entering SCREEN1,1 . Now you will find orange text on a cyan background. Enter COLOR5,8 to get white text on an orange background . Type CLS and the screen will clear to orange (the background color). Enter CLS7 and the screen will clear to magenta. Practise changing color-sets, modes and colors.

You will soon notice that in different modes, the size of the text varies. In fact, in PMODEs Ø and 1 the screen size for text is 16 across and 8 down. In PMODEs 2 and 3 the screen size is 16 by 16. In PMODE4 it is 32 by 16. PRINT@ on the graphic screen is fully implemented in EXP. Look at the PRINT@ sheets in Appendix A to work out the PRINT@ positions.

We haven't yet changed graphic pages. Try doing so by entering PMODEØ ,2 . Type anything and press <ENTER>. Then enter PMODEØ,1 . Enter PMODEØ,2 again and you will find that your text is still on that page.

Small text

Get back into PMODE4 ,1 by entering PMODE4 ,1 . The PMODE4 graphic screen is very special because, with EXP, one can change the screen size from 32 by 16 to 51 by 24. Obey the following instructions to do this:

If you have disk
---
1) Insert EXPDISK into drive Ø.
2) Enter: *SIZE(51X24)

If you have tape
---
1) Put EXPTAPE in your recorder.
2) Position tape to beginning of "SIZ51X24" program.
3) Enter CLOAD. When the 'OK' prompt reappears press <STOP> , enter RUN and press <PLAY> again.
4) When the 'OK' prompt appears (it will be very small) press <STOP>. Note that the current program in memory will be erased when "SIZ51X24" is run.

Type whatever you fancy on the screen. You will observe that the characters being typed in are diminutive. You can change color-sets , colors and graphic pages in the same way as you did with the 32 by 16 screen. However, when you are in the '51X24 state' you can only print text on the PMODE4 graphic screen or the text screen. Enter PMODE3 and you will find yourself back on the text screen because you cannot have PMODE3 text in the '51X24 state'. Enter PMODE4 to get back into PMODE4.

To return to the '32X16 state' obey the following instructions:

If you have disk
---
1) Insert EXPDISK into drive Ø.
2) Enter: *SIZE(32X16)

If you have tape
---
1) Put EXPTAPE in your recorder.
2) Position tape to the start of "SIZ32X16" program.
3) Enter CLOAD. When the 'OK' prompt reappears press <STOP> , enter RUN and press <PLAY> again.
4) When the 'OK' prompt appears press <STOP>.
Warning: the current program in memory will be erased!
IMPORTANT REMARK : You cannot use filenames after the CLOAD and SKIPF commands when you are in the "TEXTOFF" state.

KEYWORDS

(This is unavailable if you are in the '51X24 state')

All of the grey keys of the keyboard have been defined to represent two BASIC words. To find a list of these words look at Appendix B. Find the letter Q in Appendix B. You will see that under the column headed 'Right Arrow', the key stands for the word FOR. This means that whenever you press <Q>, while holding right arrow down, the word FOR will become part of the line that you are typing in. Look under the column headed 'Down Arrow' and you will see the word TO listed. This means that Q, pressed in conjunction with the down arrow key, represents TO . Try the two above examples and see for yourself what happens. You will find your programs much easier to type in using this method ! A cut-out keyboard overlay is provided. You might like to cut it out now?

THE FOLLOWING CHAPTERS

For the remainder of this section, it will be assumed that you have already loaded EXP, are in PMODE4 and in the 'TEXTOFF state'. If not, please use the loading instructions that are described at the beginning of this chapter.

Chapter 4. Improved loops

REPEAT...UNTIL LOOPS

The only loop available in ordinary BASIC is the FOR .. .NEXT loop . EXP introduces a new kind of loop, called a REPEAT... UNTIL loop. (These loops figure prominently in many high level languages like Pascal). An example of a REPEAT...UNTIL loop in a program is the following:

```
10 PMODE4,1
20 COLOR1,0
30 CLS
40 REPEAT
50 PSET(RND(256)-1,RND(192)-1,1)
60 UNTILPPOINT(128,96)=1
70 END
```

The above program sets random green points on a black background . It repeats this process until the point at the centre of the screen (co-ordinates 128,96) is green. Therefore , by deduction , a REPEAT ...UNTIL loop keeps repeating what is in-between the REPEAT and UNTIL commands, until a certain condition is true . Anything can be placed between the REPEAT and UNTIL commands, including other REPEAT .. .UNTIL loops and FOR...NEXT loops.

The REPEAT command may only be placed at the beginning of a program line (only spaces can precede it). An ?RG error will result if you attempt to execute an UNTIL command before the corresponding REPEAT command is executed.

MULTILINE IF...THEN...ELSEs

The IF...THEN...ELSE statement of Color BASIC is probably one of the most vital commands of the vocabulary. It has, however, one major drawback: the IF,THEN and ELSE commands must all be placed on the same program line. In EXP, using the multiline IF. ..THEN...ELSE feature, one can introduce an IF .. .THEN .. .ELSE statement to take up as many lines as one chooses. An example of multiline IF...THEN...ELSEs is given below:

```
10 REPEAT
20 INPUT"Number";n
30 IFn<0THEN
40 PRINT"Negative"
50 ELSE
60 PRINT"Zero or positive"
70 ENDIF
80 UNTIL0=1
```

The above program includes a few things that you have not encountered before . Firstly , variables in EXP can either be uppercase or lowercase. The first two characters of the variable name are the only two that are recognized by the computer. Lowercase variables are separate and distinct from uppercase variables , i.e. the variable A is different from the variable a.

Secondly, the REPEAT...UNTIL 0=1 which surrounds the main body of the program means 'repeat forever'. The only way to get out of the program is to press (BREAK). Line 30 to 70 of the program make up one big IF...THEN...ELSE statement. The ENDIF in line 70 signifies the end of the IF...THEN...ELSE statement.

How does the computer know whether you intend to use a normal IF . . .THEN ...ELSE statement or a multiline one? The answer is very simple. If nothing is placed after the THEN command (spaces ,REM commands and colons count as something in this context!) then a multiline IF .. .THEN .. .ELSE statement is assumed . Also , you must be careful not to put anything after the ELSE and ENDIF commands -- only spaces are permitted before them . Like a normal IF...THEN...ELSE statement the ELSE part is optional. The ENDIF command is mandatory. Between the THEN and the ELSE , and the ELSE and the ENDIF anything can be written (including REPEAT...UNTIL loops, normal and multiline IF...THEN. ..ELSE statements).

INDENTATION

Indenting programs can make programs easier to understand. To indent programs use the EDIT command to insert the required number of spaces. Everything between REPEAT and UNTIL commands, FOR and NEXT commands, and THEN and ENDIF commands (apart from the ELSE command) should be indented three of four spaces further to the right. Consider the layout of the above program when it is indented:

```
10 REPEAT
20     INPUT"Number";n
30     IFn<0THEN
40         PRINT"Positive"
50     ELSE
60         PRINT"Zero or negative"
70     ENDIF
80 UNTIL0=1
```

This facility makes the program much easier to read than one lacking indentation. All programs in this manual will be indented from now on.

Chapter 5. Procedures

Procedures are very similar, but more versatile, than subroutines. Every procedure has its own name that can be a maximum of 30 characters long. To execute a procedure you must place the PROC command in front of the procedure's name. Information can be passed to the procedure. Each item of information must be separated by a comma and brackets must enclose the whole list of information items.

A procedure is defined using the DEFPROC command. If the procedure requires information to be passed to it, then a list of variables must be placed inside brackets (each variable being separated by a comma). When the procedure is executed each variable is made equal to the corresponding item of information. At the end of the definition an ENDPROC command must be placed. This tells the computer that the procedure has ended and the computer must now execute the BASIC command that is after the PROC command. In many ways, PROC is like GOSUB, and ENDPROC is like RETURN. The following program illustrates the use of procedures:

```
 10 PMODE3,1
 20 COLOR6,5
 30 CLS
 40 SCREEN1,1
 50 PROChome
 60 REPEAT
 70     PRINT@0,"";
 80     INPUTa$
 90     IFLEFT$(a$,2)="UP" THEN PROCup(VAL(MID$(a$,3)))
100     IFLEFT$(a$,4)="DOWN" THEN PROCdown(VAL(MID$(a$,5)))
110     IFLEFT$(a$,4)="LEFT" THEN PROCleft(VAL(MID$(a$,5)))
120     IFLEFT$(a$,5)="RIGHT" THEN PROCright(VAL(MID$(a$,6)))
130     IFa$="CLS" THEN CLS
140     IFa$="HOME" THEN PROChome
150 UNTILa$="END"
160 END
165 '**************************
170 DEFPROChome
180 x=128
190 y=96
200 ENDPROC
210 DEFPROCup(n)
220 y=y-n
230 LINE-(x,y),PSET
240 ENDPROC
250 DEFPROCdown(n)
260 y=y+n
270 LINE-(x,y),PSET
280 ENDPROC
290 DEFPROCleft(n)
300 x=x-n
310 LINE-(x,y),PSET
320 ENDPROC
330 DEFPROCright(n)
340 x=x+n
350 LINE-(x,y),PSET
```

360 ENDPROC

Lines 10 to 40 of this drawing-board program simply change modes, color-sets and colors. Line 50 executes a procedure called "home" which 'homes' the x and y co-ordinates of the screen, i.e. makes them equal to the centre of the screen. The REPEAT...UNTILa$="END" loop keeps repeating until your input is "END" (see Line 80). Line 70 makes sure that the cursor is at the top of the screen when you input.

If you input "UP35" then the computer will draw a line upwards that is 35 units long. The same applies to DOWN, LEFT and RIGHT. If you enter "CLS" then the computer will clear the screen, and if "HOME" is entered then it will move the x and y co-ordinates to the screen's centre (128,96).

Lines 90 to 120 are the most difficult to comprehend. In plain English, Line 90 reads: 'If the first two characters of a$ are "UP" then execute a procedure with the name of "up". The information to be passed to the procedure is the value of a$ from the third character onwards. The variable n is made equal to that value.' A similar concept applies to Lines 100 to 120. (Note the END command in Line 160. It does not really need to be there since the computer will never execute Line 160). The rest of the program consists of the definitions of the procedures. Save the program. You will be needing it in the next chapter.

If you are writing a program in which a string constant is an item of information that is passed to the procedure then you must put a ' ""+ ' (null string and then a plus sign) in front of the string constant. The DEFPROC command should only be placed at the beginning of a program line and you should put nothing before it (not even spaces). The names of procedures do not need to be in lower-case, but they are certainly easier to read that way. The names can be anything you choose.

An ?RG error will occur if you try to ENDPROC without executing a procedure. A ?UL error will result if the procedure's name does not exist anywhere in the program. You will get an ?SN error or ?TM error if there is an error in a PROC command line or DEFPROC command line. The error will always be registered in the PROC line. A ?SN ERROR will also arise if you try to execute a procedure without using the PROC command.

DELPROC

Every time a procedure is executed, the computer must remember where it must return to so that when the ENDPROC command is executed it will know where to continue your program. When it does return, the computer has no further need of remembering the 'return line' and so the computer will forget. By using the DELPROC command you can make the computer forget the last return line that it remembered, without having to return from the procedure! By using DELPROCØ, the computer will forget ALL return lines . The syntaxes for DELPROC and DELPROCØ are simply DELPROC and DELPROCØ! You'll only very occasionally want to use DELPROC and DELPROCØ.

STRUCTURED PROGRAMMING

Programming in a structured way is possible in EXP. Structured programming is a necessity in many other computer languages . In EXPANDED BASIC it involves:

1) Indenting programs.
2) Not using GOTO unless absolutely necessary.
3) Using procedures , REPEAT...UNTIL loops and multiline IF... THEN...ELSE statements.
4) Hardly ever using colons.
5) Using REM statements to describe the functioning of your program.
6) Using lower-case variables and procedure names.

Structured programs are far easier to follow that unstructured ones.

Chapter 6. Error Trapping

In EXP, one can trap errors. 'Trapping errors' means going to a certain section of a program (called the 'error trap routine') when an error occurs. Normally, in Extended Color BASIC, an error message (e.g. ?SN ERROR) will be printed and the computer will stop running your program. By using the 'ONERROR :GOTO' command we can instead go to a certain part of the program when an error happens.

Reload the program you typed in in Chapter 5. Let us alter the program so that when an error occurs, the computer will go into PMODE4, color-set Ø before the error message is printed. Add the following lines:

```
5 ONERROR:GOTO1000
1000 PMODE4,1
1010 CLS
1020 SCREEN1,0
1030 ERRORERR
```

Line 5 tells the computer to go to line 1000 when an error occurs. Lines 1000 to 1020 go into PMODE4, color-set Ø and clear the screen . EXP has a special command, ERROR, which causes an error message to be displayed. For example enter ERRORØ and the computer will respond with an ?NF error message. This is because the code for NF error is Ø. Enter ERROR4 and you will get an ?FC error (FC error's code is 4). A function called ERR tells us the code of the error that just occurred . Enter PRINTERR and you should get 4 (4 was the code of the last error). Therefore, in line 1030, ERRORERR means: 'print the error message of the error that has just occurred'. (The ERROR command also causes the computer to stop running your program) . EXP has another function called ERL. ERL tells you the line in which the error occurred. There is a list of all error codes in Appendix C.

How can an error occur in the program? Well, suppose you were to input "LEFT1000". Since the grid size of the graphic screen for graphics is 256 by 192 , going LEFT1000 would obviously cause an ?FC error. Run the program and see for yourself what happens.

EXP has three other commands that can be used in error trapping. They are CONTON, CONTOFF and CONTERROR. These commands affect the operation of the <BREAK> key. CONTOFF disables the <BREAK> key (i.e. <BREAK> now acts like any other key of the keyboard) . CONTON re-enables the <BREAK> key . CONTERROR makes <BREAK> cause an error of code 127 everytime it is pressed. Now enter ERROR127 . You should get the break message, but in fact you do not! Instead, enter STOP. This time the message 'BREAK' does appear ! this shows that to BREAK the program you STOP rather than invoke ERROR127.This is how you do it in practice.

Add line 7 to your program:

```
7 CONTERROR
```

Delete line 1030 and add:

```
1030 IFERR=127THEN
1040     STOP
1050 ELSE
1060     ERRORERR
1070 ENDIF
```

Now run the program. The break message should be displayed on the PMODE4 screen when you press <BREAK>. Next change line 7 to:

```
7 CONTOFF
```

and run the program. Try pressing <BREAK>. You will find that this will no longer let you 'break out' of the program. Type "DOWN20" but do not press <ENTER>. Instead press <BREAK>. Nothing should happen. When <BREAK> is in the 'CONTOFF state' and is pressed when you are inputting, <BREAK> will act like <ENTER> except that it causes the computer to ignore what you have just typed in. Press RESET to get back into the 'OK' mode. Change line 7 back to:

```
7 CONTERROR
```

Delete line 1030 and add these lines:

```
1030 DELPROC0
1040 PRINT"Error"
1050 GOTO70
```

Now run the program. You will discover that the computer prints "Error" everytime you press <BREAK> or cause an error. You should always use DELPROC0 in the error trapping routine, if the program contains prcedures. Remember that 'ONERROR :GOTO' closes all open files and FOR...NEXT loops and also causes the computer to forget where it must return to after a GOSUB command.

The command ERROROFF is the opposite of 'ONERROR:GOTO' and allows the computer resume the normal way of printing error messages and breaking out of programs when an error occurs.

RUN and CLEAR

Here are a few facts about the RUN and CLEAR commands. Whenever a program is run, the computer closes all files, clears all variables, etc. In fact, the only thing that the computer remembers is the actual program. The same applies to Color BASIC's CLEAR command. When the CLEAR command is entered everything is forgotten except the actual program and the number of string storage bytes that should be reserved.

Chapter 7. New sounds and characters

The tones available to you in Extended Color BASIC come via the PLAY and SOUND commands. Their quality is governed by the 'square wave' shape of the note. With EXP, you can alter the 'quality' of the tone and obtain some really bizarre sound effects by changing the shape of the wave patterns. The method is to define an envelope (wave) of sound with the ENVELOPE command and to play it using the BEEP command. The ENVELOPE command lets you increment the volume of the note played in each of four successive stages. The syntaxes for ENVELOPE and BEEP are:

```
ENVELOPE n;pi1,vi1,s1;pi2,vi2,s2;pi3,vi3,s3;pi4,vi4,s4
```

where n is the envelope number, pi is the pitch increment, vi is the volume increment and s is the number of increments. n, pi, vi and s are numeric expressions.

```
BEEP n;sp,sv,t
```

where n is the envelope number, sp is the starting pitch and sv is the starting volume. t is the number of times the envelope is beeped.

You may define up to 16 envelopes. Therefore the values for n must go from 1 to 16. pi,vi,s,sp,sv,t all range from 0 to 255.

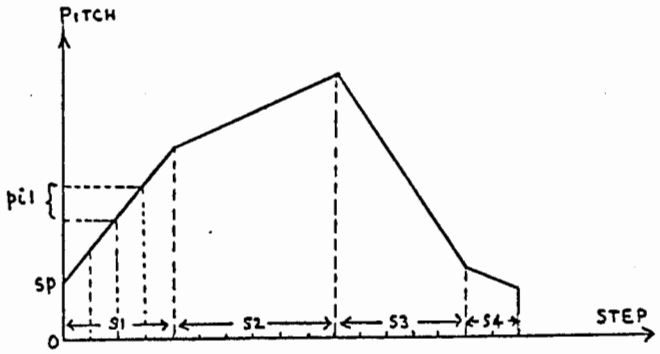


FIGURE P

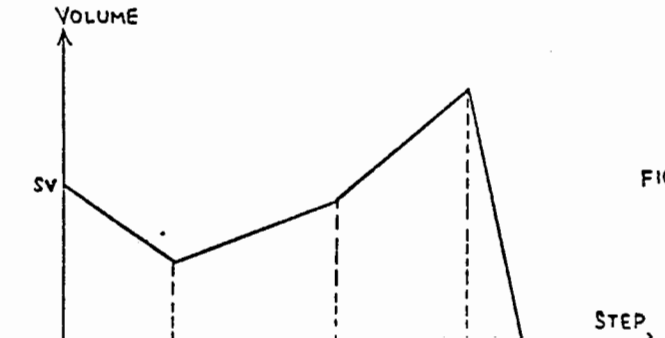


FIGURE V

Each of the 4 parts of the envelope instructs the computer to increase the pitch (pi) and volume (vi) for a specified number of steps. See the Figures p and v to see what happens typically. (If you wish to silence the computer in a particular part of the envelope make pi,vi and s all equal Ø). The starting pitch and volume of each sound effect is determined by the sp and sv values in the BEEP command. t is the number of times that the computer must 'beep' the envelope.

Although vi and pi range from Ø to 255 , if vi and pi are made to exceed this range (by incrementing) then the numbers restart from Ø (i .e . the computer interprets the number 257 as 1). This means that you can decrease vi or pi by one by adding 255 ; similarly for other decrements.

An example of the BEEP and ENVELOPE commands in a program is as follows:

```
1Ø ENVELOPE1;1Ø,7,5;Ø,Ø,Ø;Ø,Ø,Ø;Ø,Ø,Ø
2Ø FORp=ØTO2ØØSTEP1Ø
3Ø    BEEP1;p,1ØØ,3Ø
4Ø NEXTp
5Ø END
```

Line 1Ø defines only part 1 of envelope 1. (The other parts are silent) . The envelope increases the pitch by 1Ø and the volume by 16 five times. Line 3Ø beeps envelope 1 with the starting pitch as the variable p , the starting volume as 1ØØ, for 3Ø times. Run the program and hear the sound effect. Note that the higher the pitch, the faster the beep.

Some useful sound envelopes are listed in Appendix D. You are urged to experiment and develop your own.

REDEFINING CHARACTERS

Using EXP, one can redefine a character. The syntax for redefining characters is:

```
CHR$(c)=d1,d2,d3,d4,d5,d6,d7,d8,d9,d1Ø,d11,d12
```

where c is a numeric expression ranging from 32 to 127 and the 'd's are numeric expressions ranging from Ø to 255.

For example, to redefine CHR$(95) (produced when you press <SHIFT>/<up arrow>) from a back arrow to a dagger, you would first have to draw up the new character on a sheet of paper . You would write a '1' to represent the foreground color and a 'Ø' for the background color. There are 8 dot positions across and 12 down for each character. The dagger, drawn in a series of Ø's and 1's would look like:

```
ØØØØØØØØ   =Ø
ØØØØØØØØ   =Ø
ØØØ1ØØØØ   =16
Ø1111ØØØ   =124
Ø1111ØØØ   =124
ØØØ1ØØØØ   =16
ØØØ1ØØØØ   =16
ØØØ1ØØØØ   =16
ØØØ1ØØØØ   =16
ØØØ1ØØØØ   =16
ØØØØØØØØ   =Ø
ØØØØØØØØ   =Ø
```

The next step is to translate each row into a number. Each row represents one byte in binary. To work out the conversion to decimal look up pages 184 to 189 of "Going Ahead With Extended Color BASIC" . There you will see a list of base conversions. Look under the column headed binary and find the appropriate pattern of Ø's and 1's that corresponds exactly to the pattern of the row. In the example above, these numbers are written after the equal sign. Thus you would then enter:

```
CHR$(95)=Ø,Ø,16,124,124,16,16,16,16,16,Ø,Ø
```

Press <SHIFT>/<up arrow> and you will discover a dagger instead of a back arrow.

You may also redefine characters when in the '51X24 state' . However , the 'd's must go from Ø to 15 and there must be only eight 'd's listed (instead of 12) since a '51X24' character is 4 dots across and 8 dots down.

Chapter 8 - Scrolling

SCROLLING ON THE TEXT SCREEN

Type in the following program:

```
10 TEXTON
20 CLS8
30 PRINT@230,"THIS IS AN EXAMPLE";
40 PRINT@265,"OF SCROLLING";
50 REPEAT
60     SCR(6,5)-(23,10),150,L
70     FORdelay=1TO400
80     NEXTdelay
90 UNTILØ=1
```

and run it. Press <BREAK> to get back into the 'OK' mode. The program is an example of scrolling. Scrolling, put simply, is moving the screen display within a section of the screen. In the above program , the message "THIS IS AN EXAMPLE OF SCROLLING" moves slowly left across a certain scetion of the text screen. We'll call a section of a screen a 'window'.

To scroll on the text screen use the SCR command. Its syntax is:

```
SCR(a,b)-(c,d),e,direction
```

where (a,b) is the top left corner of the window and (c ,d) is the bottom right corner of the window. e is the alphanumeric code of the column or row of characters that is inserted (in the case of the program above: the column is inserted at the far right of the window). Direction is U for up, D for down, L for left and R for right. Obviously a must be less than c, and b<d. Since the text screen is 32 across by 16 down, a and c are numeric expressions ranging from Ø to 31, and b and d range from Ø to 15. In line 60 the SCR command line was:
SCR(6,5)-(23,10),150,L
Therefore the top left corner of the window has a x coordinate of 6 and a y coordinate of 5. The bottom right corner of the window has coordinates 23,10. The code for a 'checkerboard' yellow graphic character is 150 , so a column of these characters is inserted at the far right of the window. See Appendix E for a list of the alphanumeric code.

Change Line 60 so it reads:

```
60 SCR(6,5)-(23,10),150,U
```

When you run the program you will notice the window scrolling up. Change Line 60 again to:

```
60 SCR(6,5)-(23,10),150,R
```

This time running the program will cause the window to scroll right . Yet again, change Line 60 to the following and run the program:


```
60 SCR(6,5)-(23,10),Ø,R
```

In this example e is Ø and therefore NO column of characters is inserted . Change the size of the window by altering Line 60 to the following and running the program:

```
60 SCR(Ø,Ø)-(31,15),150,L
```

Here the window occupies the whole screen so the whole screen will move left. Finally, change Line 60 to:

```
60 SCR(Ø,Ø)-(31,15),255,L
```

and run the program. Only the message will appear to move; this is because the character column that is inserted is full orange which is the same color as the background color (Line 20 of the program).

SCROLLING ON THE GRAPHIC SCREEN

Type in the following program:

```
10 TEXTOFF
20 PMODE3,1
30 COLOR3,2
40 CLS
50 CIRCLE(138,96),90
60 PRINT@132,"Scrolling";
70 REPEAT
80     PSCR(Ø,Ø)-(31,47),L
90 UNTILØ=1
```

and run it. A circle will scroll to the left. The syntax for PSCR is very similar to SCR, and reads:

```
PSCR(a,b)-(c,d),direction
```

Notice that there is no e in the syntax of PSCR. e is assumed to be the current background color. In PMODEs Ø and 2, a and c go from Ø to 15. In PMODEs 1,3 and 4, a and c range from Ø to 31. In PMODEØ, b and d go from Ø to 11. In PMODEs 1 and 2, b and d go from Ø to 23. In PMODEs 3 and 4, b and d range from Ø to 47. Hence the window mentioned in Line 80 of the above program occupies the entire graphic screen . In the above program, e was 2 because the background color was 2. If the background color is the same as the foreground color then no new color column or row is inserted.

Chapter 9. A Miscellany

BORDER

The BORDER command will draw a border on the text screen. In a way , BORDER is very similar to the high resolution command 'LINE...,B' of Extended Color BASIC. BORDER's syntax is:

```
BORDER(a,b)-(c,d),e
```

where (a,b) are the co-ordinates of the upper left corner of the rectangle in which the border will be drawn, and (c,d) are the co-ordinates of the bottom right corner. Obviously a<c, b<d. a and c are numeric expressions that go from Ø to 31, b and d go from Ø to 15. For example, to draw a border in a checkered yellow character , just inside the edge of the text screen, one would type in the following program:

```
1Ø TEXTON
2Ø CLS
3Ø BORDER(Ø,Ø)-(31,15),15Ø
4Ø REPEAT
5Ø UNTILØ=1
```

The REPEAT...UNTILØ=1 loop of course means 'repeat forever' and because it is placed at the end of the program, it acts like an infinite loop. Typing

```
4Ø GOTO4Ø
```

would achieve the same effect. However, if you decide to write programs in a structured way, then you should use as few GOTO's as possible. Change Line 3Ø to:

```
3Ø BORDER(Ø,Ø)-(31,15),15Ø,F
```

and run the program. This time the whole screen is 'checkerboard yellow' . This is because the ',F' at the end of the BORDER command line tells the computer to fill the area of the screen surrounded by the border with the same character that the border has been drawn in.

EXTRA GRAPHIC PAGES

Enter TEXTOFF. Then enter NEW12. Finally enter PMODEØ ,12. You might imagine that entering PMODEØ ,12 will generate an ?FC error, but this is not the case. The reason is that the NEW command with a number following it tells the computer to not only erase the program currently in memory, but to PCLEAR that number of graphic pages. Normally one can PCLEAR only 8 graphic pages, but with NEW the highest number of pages that can be reserved is 18. Therefore when you entered NEW12, the computer NEWed your program and PCLEARed 12 graphic pages. It was then possible, of course, for the computer to go into PMODEØ,12 .

FILL

In EXP , you can fill sections of memory with a particular number. The syntax for the FILL command is:

```
FILL s,e,n
```

where s is the start address of the section of memory, e is the end address of the section of memory and n is the number with which the section of memory will be filled. s and e are numeric expressions going from Ø to 65535 , and n is a numeric expression with a range of Ø to 255. Enter TEXTON (unless you are already in the 'TEXTON state') and enter FILL1Ø24,1535,255. The text screen should clear to orange, because the text screen goes from byte 1Ø24 to byte 1535 in memory and 255 is the alphanumeric code for the full orange character.

MCOPY

The MCOPY command copies a section of memory to another section of memory. Its syntax is:

```
MCOPY f,t,n
```

where f is the start address of the section of memory that is being copied and t is the start address of the section of memory that is receiving the copy. n is the number of bytes that will be copied.

REVERSE VIDEO

REV 'reverses' the color of the text characters on the text screen. Enter the following program to see REV in action:

```
1Ø CLS
2Ø PRINT"PRESS ANY KEY TO REVERSE THE COLOR"
3Ø REPEAT
4Ø    REPEAT
5Ø    UNTILINKEY$<>""
6Ø    REV
7Ø UNTILØ=1
```

Press <BREAK> to get back into the 'OK' mode.

GOTO, GOSUB VARIABLE LINE NUMBERS

In EXP , one can put variables after GOTO and GOSUB commands. Type in the following:

```
10 h=10
20 REV
30 GOTOh
```

and run it to understand its significance. (You can use this method instead of Extended's ON ...GOTO.., or ON...GOSUB...) Press <BREAK> to escape from the program.

SECTION 2 - EDIT/HELPER COMMANDS
****

Chapter 10. Keys

Due to memory limitations, all of the features of EXP cannot exist in memory at once. This section of the manual describes the editing and helper features of EXP. If you have these features in memory then you cannot have the graphic screen features as well. The graphic screen features introduced in Section 1 were:
printing text on the graphic screen,
scrolling the graphic screen,
PCLEARing more than 8 graphic pages
using the NEW command, and
switching from text to graphic screen and vice versa using the TEXTON and TEXTOFF commands.

Therefore a separate EXP program was made that has the editing and helper features but not the graphic screen features. To load that program first switch off your computer. If it is already on, switch it off and turn it on again.

If you have disk and a CoCo #1
---

1) Load EXP as described in Chapter 3.
2) Put the EXPDISK in drive 0.
3) Enter: *COM(E)

If you have disk and a CoCo #2
---

1) Insert EXPDISK into drive 0.
2) Enter RUN"R" . When "R" is fully run, the following message should appear on the screen:

```
DISK EXTENDED COLOR BASIC 1.0
COPYRIGHT (C) 1980 BY TANDY
UNDER LICENSE FROM MICROSOFT
```

3) Enter RUN"E". When the program has fully run, the following message should appear on the PMODE4 graphic screen:

```
DISK EXPANDED COLOR BASIC 1.3
COPYRIGHT 1984 BY T.DELBOURGO
INCLUDES EDIT/HELPER COMMANDS
```

If you have tape
---

1) Put EXPTAPE into tape recorder.
2) Position tape to beginning of "R" program.
3) Enter CLOAD and press <PLAY>. When the 'OK' prompt reappears press <STOP>.
4) Enter RUN and press <PLAY>. When the following message appears, press <STOP>:

```
EXTENDED COLOR BASIC 1.0
COPYRIGHT (C) 1980 BY TANDY
UNDER LICENSE FROM MICROSOFT
```

5) Position tape to beginning of "E" program.
6) Enter CLOAD and press <PLAY>. When the 'OK' prompt appears press <STOP>, enter RUN and press <PLAY> again.
7) After the following message appears, press <STOP>:

EXPANDED COLOR BASIC 1.3
COPYRIGHT 1984 BY T.DELBOURGO
INCLUDES EDIT/HELPER COMMANDS

The editing and helper features now reside in your computer's memory.

ON-SCREEN COPYING

Type anything you like and press <ENTER>. Now press the <CLEAR> key. While holding the <CLEAR> key down, press the up arrow key. You will see a reverse video cursor (which we shall call the 'second cursor') above the normal cursor. By holding down the <CLEAR> key and simultaneously pressing the arrow keys, one can move the second cursor around the screen. Note that the <CLEAR> key will no longer clear the screen. Position the second cursor over a character on the screen. Press <CLEAR>/<@> and the computer will now copy the character and make that character part of the line that you are typing in. Observe that the computer also moves the cursor one position to the right when you press <CLEAR>/<@>. When you have finished copying, by pressing <ENTER> the second cursor vanishes and you will have to press <CLEAR>/<arrow key> to bring it back into operation.

This method of moving a second cursor about and copying characters on the screen offers a new way to edit programs and information. On-screen copying also works with the LINEINPUT and INPUT commands.

FUNCTION KEYS

You can define any of the number keys (Ø to 9) to stand for (up to) 15 characters. For example, you could define <Ø> to represent LIST . You activate this by using the KEY command. In this case you would enter:

```
KEYØ"LIST"
```

Now, whenever <CLEAR>/<Ø> is pressed, LIST will become part of the line that you are currently typing in. You can also make a key represent an enter, for instance. To make <Ø> represent LIST <ENTER> you would enter:

```
KEYØ"LIST"+CHR$(13)
```

CHR$(13) is the character that represents the enter key. Now all you need do is press <CLEAR>/<Ø> and your program is listed.

AUTOKEY REPEAT

Is it not a nuisance that if you wish to type 32 spaces, you must press and release the <SPACEBAR> 32 times? With 'autokey repeat' you need only press the key down once and hold it until 32 spaces appear on the screen. To enable autokey repeat use the KEYSCR command. Its syntax is:

```
KEYSCR a,b
```

where a is the number of sixtieths of a second before the key 'repeats' and b is the number of sixtieths of a second between each repeat . a and b are numeric expressions ranging from Ø to 255.

Enter KEYSCRØ,3 . Press the <SPACEBAR> and hold it down . Notice how the key autorepeats . There is one problem with autorepeating: when entering BASIC words using the right and down arrow keys , as described in Chapter 3, the key that you are pressing down autorepeats as well as the BASIC word! This is unavoidable.

To disable autokey repeat, enter:

```
KEYSCR255,255
```

Chapter 11. Printing Aids

AUTOLINE NUMBERING

The computer can supply you automatically with line numbers as you type in your programs . For the computer to do this, you must use the AUTO command. Its syntax is:

AUTO s,i

where s is the start line and i is the increment. Both s and i are numeric expressions ranging from Ø to 63999.

Enter AUTO2Ø ,5 . Type in the following program. With AUTO, of course, do not type in the line numbers. Here is the program:

```
2Ø PRINT"THIS IS A PROGRAM"
25 PRINT"THAT SHOULD BE TYPED IN"
3Ø PRINT"USING THE EXPANDED COLOR BASIC"
35 PRINT"AUTO COMMAND."
4Ø END
```

To get out of the 'AUTO state', press <BREAK>. Now list the program. It should appear as above.

ADJUSTABLE PRINTER WIDTH

You can change the width of your printer using the WIDTH command. Its syntax is:

WIDTH p

where p is a numeric expression representing the new printer width.

Enter WIDTH32 and LLIST the program . You will see that no LLISTed program line is longer that 32 characters, and this is exactly how it appears on the video screen.

SECTION 3 - Q-SCREEN COMMANDS

Chapter 12. The Q-screen

There is a third EXP program in which you are not able to use the graphic screen features , nor the editing and helper features. Instead you have available the 'Q-screen features' . These are associated with the Semigraphics-24 mode. In that mode, the resolution of the screen is 32 across by 192 down , and in EIGHT colors. Because of the poor quality of most TV's, these colors when blended in with one another, often produce new color mixtures , that are distinct colors. The color mixing is achieved by putting two colors (which may be different) on alternate horizontal lines.

To load the 'Q commands' first switch on your computer. If it is already on then switch it off and turn it on again. Obey the following loading instructions:

If you have disk and a CoCo#1

1) Put EXPDISK into drive Ø and load EXP.
2) Enter: *COM(Q)

If you have disk and a CoCo#2

1) Insert EXPDISK into drive Ø.
2) Enter RUN"R" . When "R" is fully run, the following message should appear on the screen:
```
DISK EXTENDED COLOR BASIC 1.Ø
COPYRIGHT (C) 1981 BY TANDY
UNDER LICENSE FROM MICROSOFT
```
3) Enter RUN"G". When the program has fully run the following message should appear on the screen:
```
DISK EXPANDED COLOR BASIC 1.3
COPYRIGHT 1984 BY T.DELBOURGO
INCLUDES Q-SCREEN COMMANDS
```

If you have tape

1) Put EXPTAPE into tape recorder.
2) Position tape to beginning of "R" program.
3) Enter CLOAD and press <PLAY>. When the 'OK' prompt reappears press <STOP>.
4) Enter RUN and press <PLAY>. When the following message appears press <STOP>:
```
EXTENDED COLOR BASIC 1.Ø
COPYRIGHT (C) 198Ø BY TANDY
UNDER LICENSE FROM MICROSOFT
```
5) Position tape to beginning of "Q" program.
6) Enter CLOAD and press <PLAY>. When the 'OK' prompt appears press <STOP>, enter RUN and press <PLAY> again.
7) After the following message appears, press <STOP>:
```
EXPANDED COLOR BASIC 1.3
COPYRIGHT 1984 BY T.DELBOURGO
INCLUDES Q-SCREEN COMMANDS
```

Enter PMODE4,1 and the Q commands wil become fully operational . It is necessary to enter PMODE4,1 because the Q-screen can only be applied in that PMODE.

Type in the following program:

```
10 QON
65 REPEAT
70 UNTILØ=1
```

and then run it. The Q-screen will be displayed. Press <BREAK> to get back into the 'OK' mode. The QON command, in Line 10 of the program, instructs the computer to show the Q-screen on the TV . The opposite of QON command is QOFF which tells the computer to go back to the text screen.

CLEARING THE Q-SCREEN

Add this line to your program:

```
15 QCLS(3,4)
```

and run it again. A new color, mauve, will fill the entire screen . The command, QCLS, clears the Q-screen to a color mix of two colors. In the example above, the colors were blue and red . This then means that all even horizontal lines will be blue, and all odd ones will be red.

Change Line 15 to:

```
15 QCLS(Ø,Ø)
```

Now run the program. This time the screen will be cleared to a solid black color.

SETTING AND RESETTING POINTS

Add these lines:

```
20 FORx=ØTO63
25    QSET(x,96,2)
30 NEXTx
35 FORy=ØTO191
40    QSET(32,y,7)
45 NEXTy
```

and run the program . On the black screen there will appear a yellow horizontal line that is bisected by a magenta vertical line . You may deduce that the command, QSET, sets a point on the Q-screen. Its syntax is:

```
QSET (x,y,c)
```

where x is the x coordinate and ranges from Ø to 63, y is the y coordinate and goes from Ø to 191. c is the color and ranges from 1 to 8. x, y and c are all numeric expressions.

To reset a point on the Q-screen (set it to black), we use the QRESET command. Its syntax is:

```
QRESET (x,y)
```

where x is the x coordinate and y is the y coordinate. x and y are again numeric expressions.
Add these lines to the program and then run it:

```
50 FORx=ØTO63
55    QRESET(x,96)
60 NEXTx
```

The original yellow line will be reset.

FINDING OUT THE COLOR OF POINTS

We can probe the color of any location with the 'POINT(Q ;' function . Type in the following program to see 'POINT(Q;' in action:

```
10 QON
20 QCLS(Ø,Ø)
30 REPEAT
40    QSET(RND(64)-1,RND(32)-1,RND(8))
50 UNTILPOINT(Q;32,96)<>Ø
```

The program will set randomly colored dots at random positions on the Q-screen until the dot at the centre of the screen (coordinates 32,96) is not black. The 'POINT(Q ;' function in Line 50 returns the color of the point at the specified x and y coordinates.

Chapter 13. Q printing and scrolling

PRINTING ON THE Q-SCREEN

Here is a program which demonstrates how to print on the Q-screen:

```
10 QON
20 QCLS(7,8)
30 QPRINT@231,"THIS IS AN EXAMPLE"
40 QPRINT@266,"OF PRINTING"
50 REPEAT
60 UNTILQ=1
```

Run it . Press <BREAK> to return to the 'OK' mode. You will notice a new command in Lines 30 and 40. Its syntax is:

```
QPRINT m$
```

where m$ is a string expression.

QPRINT@ is also implemented. The '@' co-ordinate is the same as for the TEXT screen. Three things to note:
1) QPRINT will only print strings, not numbers. To convert a number into a string, use Color BASIC's STR$ function.
2) QPRINT will only print one string expression.
3) QPRINT does not advance to the next line, unlike PRINT. The screen does not scroll up when the printing position reaches the far bottom right of the screen. Notice that no semi-colon was placed at the end of the QPRINT command line in Lines 30 and 40. NEVER put a semi-colon at the end of a QPRINT command line . If you want to advance to the next line then put a '+CHR$(13)' at the end of the string expression.

SCROLLING THE Q-SCREEN

You can scroll the Q-screen using the QSCR command. QSCR's syntax is:

```
QSCR(a ,b)-(c ,d) ,e ,f ,direction   if a column or row is to be inserted
```

OR

```
QSCR (a,b)-(c,d),,direction   if no column or row is to be inserted.
```


(a ,b) are the coordinates of the top left corner of the window, (c;d) are the coordinates of bottom right corner of the window . The grid size of the Q-screen for scrolling is the same as for the text screen. Therefore a and c go from 0 to 31, b and d go from 0 to 15. e and f are the two colors that make up the color mix of the column or row that is inserted.

The next progrsm illustrates scrolling on the Q-screen:

```
10 QON
20 QCLS(3,4)
30 QPRINT@7,"THIS IS AN EXAMPLE"
40 QPRINT@42,"OF SCROLLING"
50 REPEAT
60    QSCR(7,0)-(24,15),7,8,D
70 UNTILQ=1
```

Type in and run the program. The message 'THIS IS AN EXAMPLE OF PRINTING' will scroll down the screen. A color mix row of 7 and 8 is inserted. Press <BREAK> to get out of the program. By changing Line 60 to:

```
60 QSCR(7,0)-(24,15),,D
```

no new row will be inserted.

IMPORTANT NOTE

NEVER USE A VARIABLE STARTING WITH THE LETTER Q WITH THE Q COMMANDS IN MEMORY.

THE STOP COMMAND HAS BEEN RENAMED QUIT.

SECTION 4 - EXTRA COMMANDS AND SAMPLE PROGRAMS
****

Chapter 14. Extra commands

With any of the three EXP programs, you may also have the 'extra commands' in memory . These commands will, however, consume 1.25K of User RAM. To load the extra commands first load any of the three EXP programs (the loading instructions for the EXP programs are mentioned in Chapters 3, 10 and 12) . Then:

If you have disk
---
1) Put the EXPDISK into drive 0.
2) Enter: *EXTRA

If you have tape
---
1) Put EXPTAPE into recorder.
2) Position tape to start of "EXTRA" program.
3) Enter CLOAD and press <PLAY>. When the 'OK' prompt reappears, press <STOP>.
4) Enter RUN and press <PLAY>. When the 'OK' prompt appears press <STOP>.

The extra commands are now resident in memory . All extra commands begin with a '.' (full stop symbol).

LOCAL VARIABLES

With the extra commands, you can make variables in procedures 'local' . When a variable is 'localized', the computer remembers its original value. Thus when the procedure is finished and the computer returns from the procedure, it restores the variable to its original value. The following program is an example of the use of local variables:

```
5 .DELLOC
7 DIM[](1)
10 INPUT"CHARACTER OF LINE";c
20 PROCprint asterisks(5)
30 PRINTSTRING$(32,c)
40 END
60 DEFPROCprint asterisks(n)
70 .LOCc
80 FORc=1TOn
90    PRINT"*"
100 NEXTc
110 .ENDPROC
```


Type in the program and run it. To type in the '[' character press <SHIFT>/<down arrow> and to type in ']' press <SHIFT>/<right arrow> . Enter any number from 128 to 255 in answer to the prompt: "CHARACTER OF LINE". What this program does is to print 5 asterisks vertically and then a line of the graphics character of the code that you entered. (If the graphic screen features happen to be in resident in memory, then enter TEXTON before running the program).

After the program has run, enter: PRINTc . The computer will respond with the same number as the number that you inputted. You would be correct in thinking that the value of c changed when the procedure was executed. However, before the value of c changed, we localized the variable. Hence, when the computer returned from the procedure, it restored the original value of c.

Now, let's go through the difficult lines of the program. Line 5 tells the computer to forget any previous local variables. You must always put '.DELLOC' at the beginning of a program that requires local variables. (You should also put '.DELLOC' in an error trapping routine). Line 7 tells the computer that only one variable will be made local and that it shall be a number. The computer can localize a maximum of 255 string variables and 255 numeric variables . In order to tell the computer that a maximum of 250 string variables and 31 numeric variables will be localized you would need to include the following line as part of your program:

```
DIM[](31),[]$(250)
```

If you are not sure how many variables will be made local then make a reasonable estimate . You can always begin with dimensioning [] and []$ larger than they need be. Line 70 localizes the variable c. If you want to localize more than one variable, then separate each variable name by a comma. The '.ENDPROC' command in Line 110 delocalizes the variable(s) and the returns from the procedure. Whenever a procedure localizes variables , you must always put an '.ENDPROC' rather than just 'ENDPROC' as the procedure's last line.

A ?BS error will result if you try and localize more variables than you have dimensioned [] and []$.

EXECUTING STRINGS AS COMMANDS

With the extra commands, strings can be executed as commands. Simply place a '.' (full stop symbol) in front of the string and you're in business.
For example, enter:

```
."CLS"
```

and the screen will clear. Enter:

```
a$="CLS"
.a$
```

and the screen will again clear . This is not possible in Extended Color BASIC! Therefore, the syntax for executing strings as commands is:

```
.c$
```

where c$ is a string expression . If the string expression begins with a string function then you must put a '""+' in front of the string expresion.

```
e.g. .LEFT$(a$,3) is incorrect, but
     .""+LEFT$(a$,3) is correct.
```

When equating string variables to be equal to string constants, using the 'executing a string as a command' feature, you must put a '""+' in front of the string constant. Also, your string cannot contain multicommands (commands separated by a colon) and cannot exceed 200 characters.

WARNING : You cannot execute strings as commands if the strings contain the words FOR, NEXT, REPEAT, UNTIL, GOTO, GOSUB, a multiline IF..THEN..ELSE statement or an extra command. Neither can you use the abbreviations question mark or apostrophe for the PRINT and REM commands in your string.

Chapter 15. Sample programs

CONGRATULATIONS !!!

If you have ploughed through the manual and understood all the concepts, then you have mastered Expanded Color BASIC. At first, you may find it necessary to reread the whole manual to fully absorb it.

Some sample programs are provided. It is time to run and study them now. Use the graphic screen features for sample program filenames beginning with the letter 'G' , and the Q-screen features whenever the filenames start with a 'Q'. To load and run the programs, obey the following instructions:

If you have disk

1) Put EXPDISK into drive Ø.
2) Enter RUN"filename" (where "filename" is the filename of the sample program).

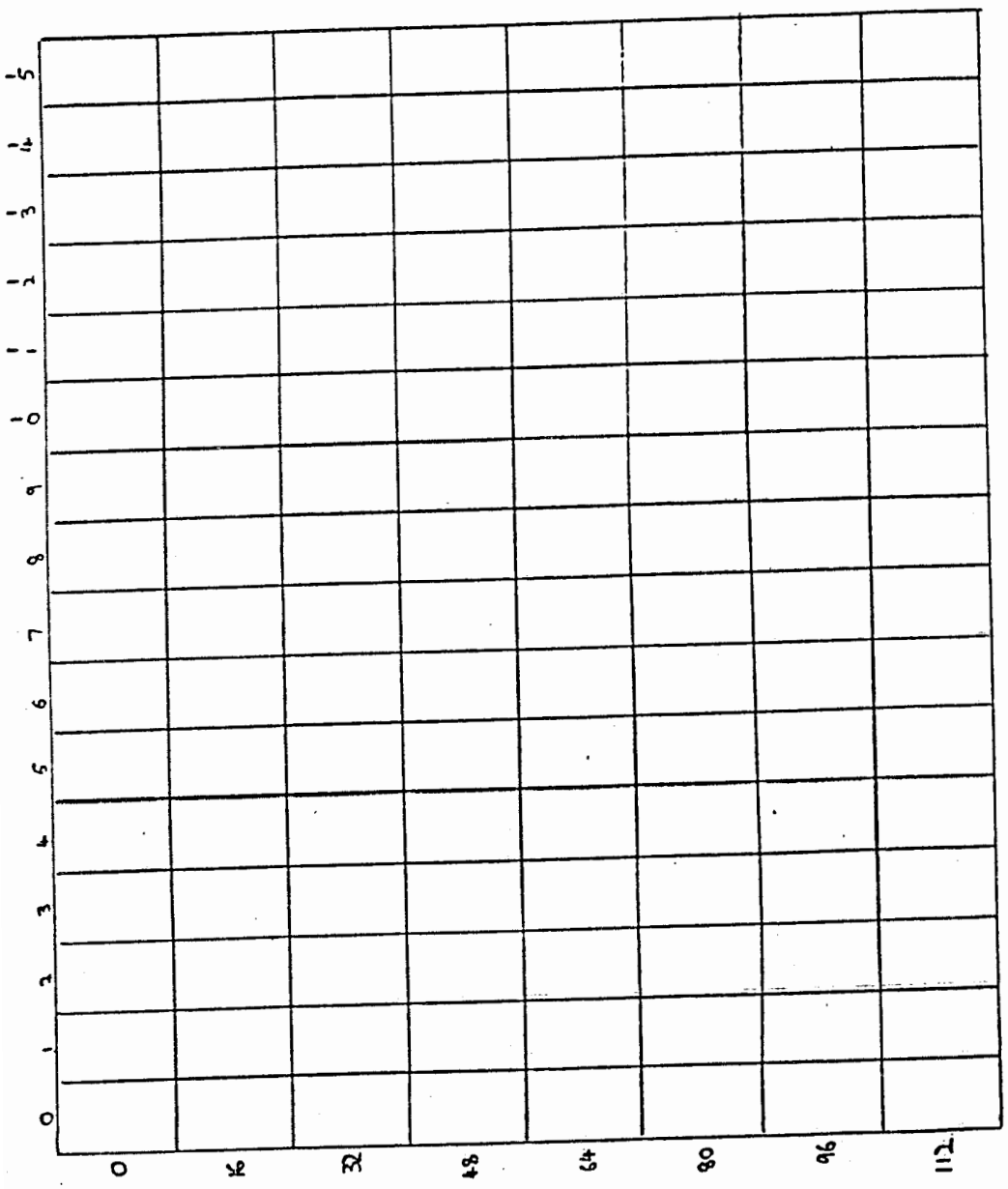
If you have tape

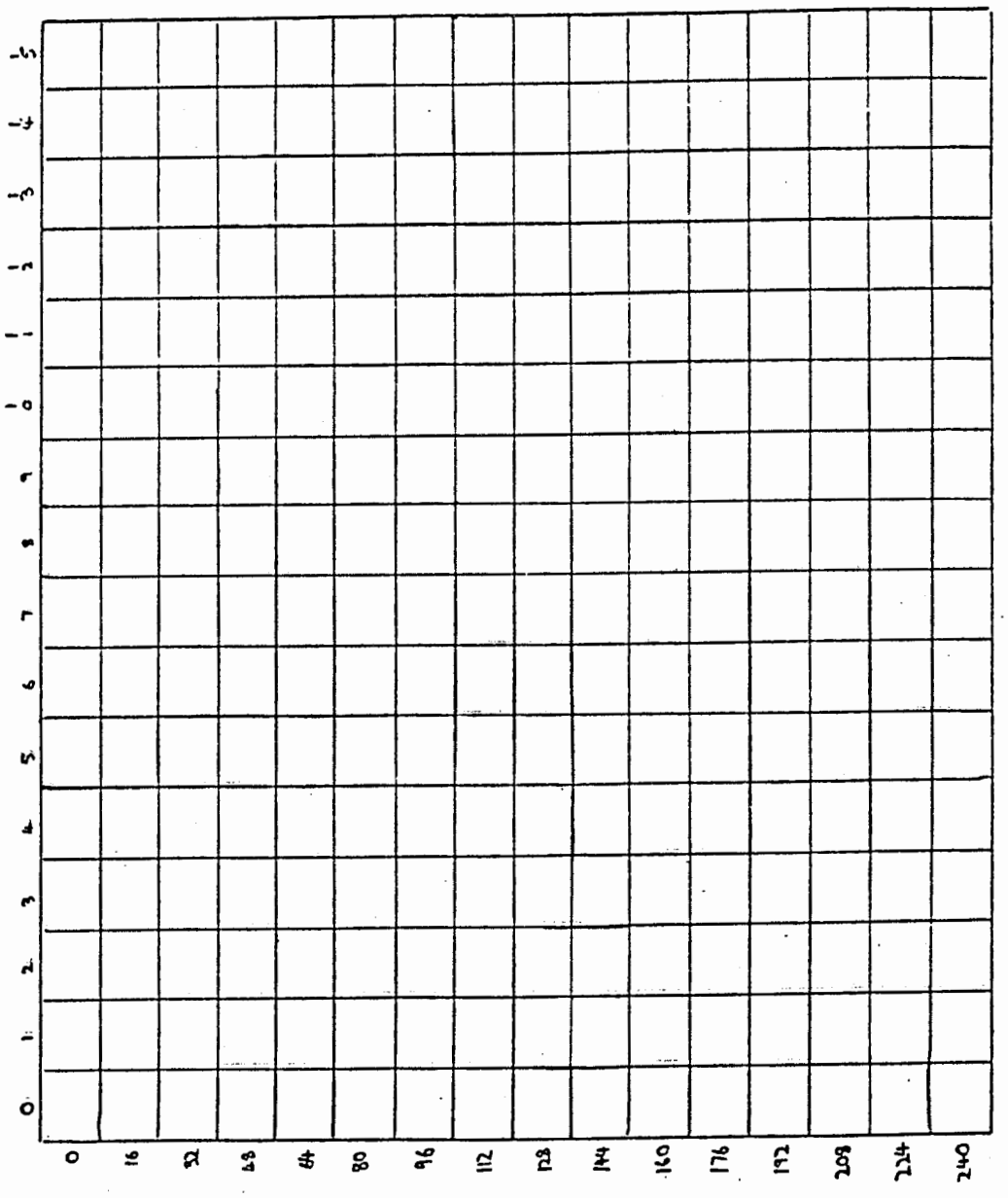
1) Put EXPTAPE into tape recorder.
2) Position tape to beginning of sample program.
3) Enter CLOAD, and press <PLAY>.
4) When the 'OK' prompt appears, press <STOP>.
5) Enter RUN

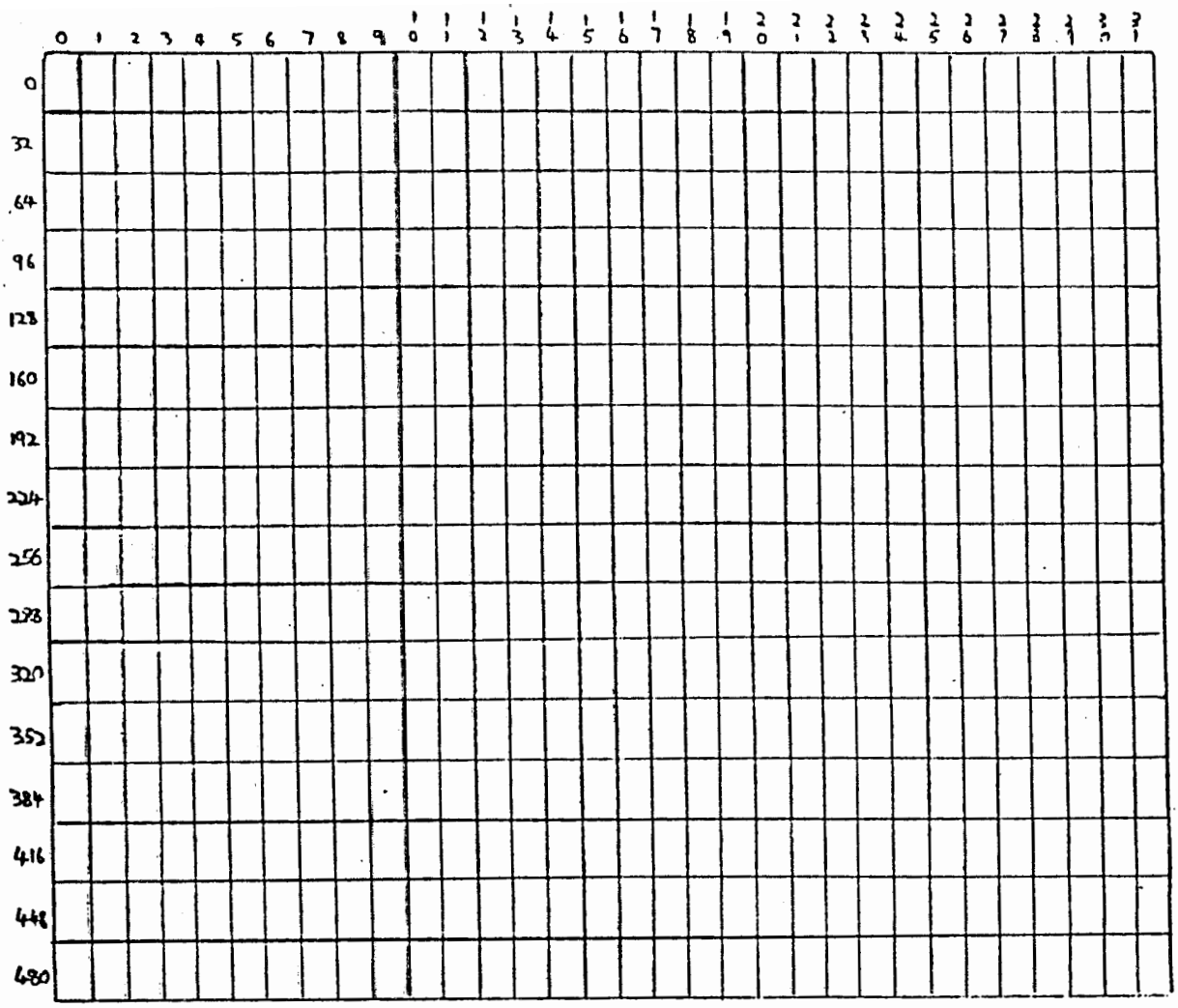
I hope that you will have fun exploiting the power of Expanded Color BASIC in your own programs. Appendices are supplied for ready reference.

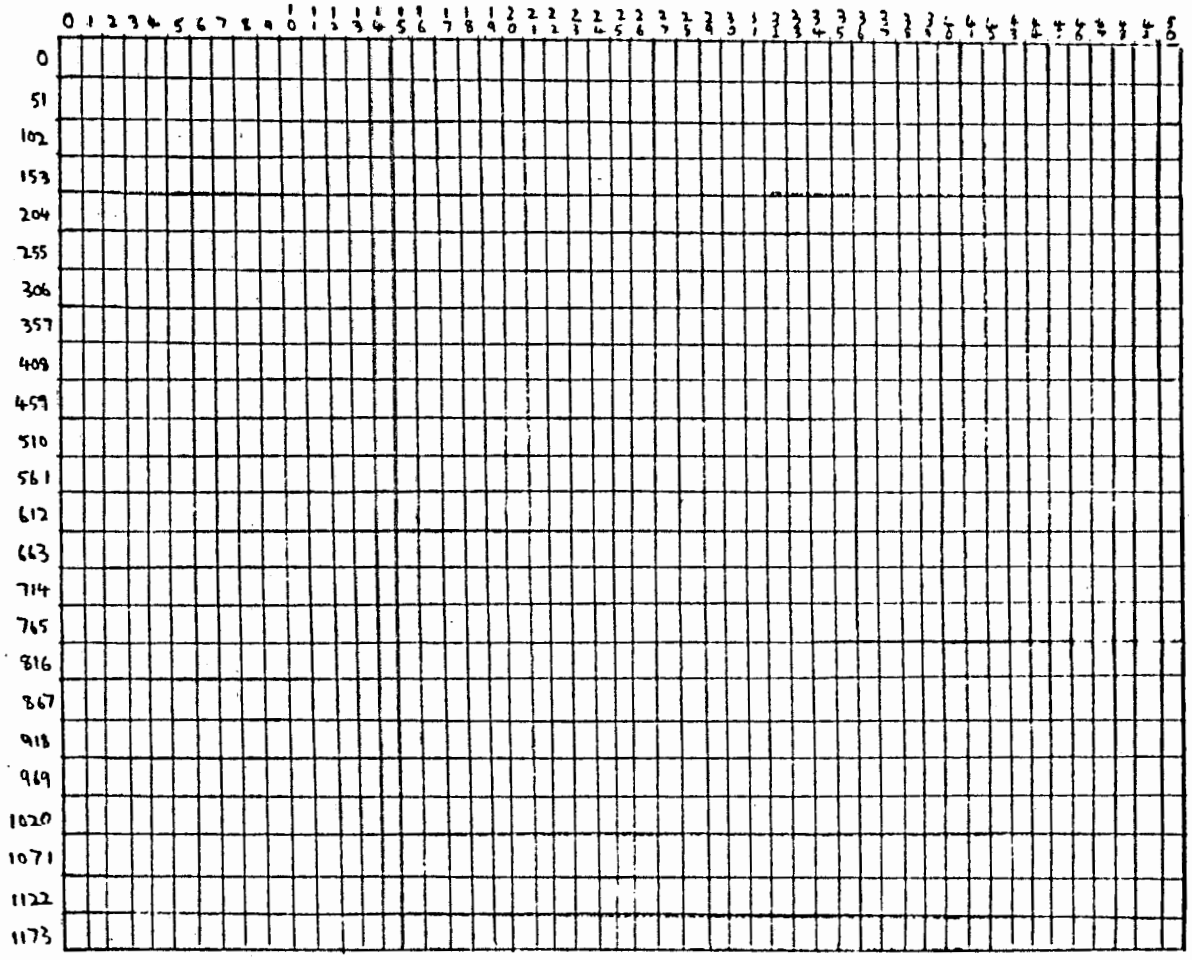
Again, happy EXProgramming.

APPENDIX A - PRINT@ POSITIONS






0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0
32
64
96
128
160
192
224
256
288
320
352
384
416
448
480


0
51
102
153
204
255
306
357
408
459
510
561
612
663
714
765
816
867
918
969
1020
1071
1122
1173

APPENDIX B - KEYWORDS

KEY	> ARROW	V ARROW	WORD	KEY	WORD	KEY
	GOTO	GOSUB	AND	6 >	LINE	N >
,	DRAW	DRIVE	ASC(	U V	LIST	1 >
-	EDIT	AUDIO	AUDIO	- V	LOAD	C V
.	GET	UNLOAD	AUTO	1 V	MERGE	N V
/	PUT	BACKUP	BACKUP	/ V	MID$(	J V
Ø	CLS	PAINT(	BEEP	G >	MOTOR	K >
1	LIST	AUTO	CHR$(	F V	NEXT	W >
2	RUN	KEY	CIRCLE(	M >	ON	; >
3	CONT	FREE(	CLEAR	: >	OPEN	4 V
4	REPEAT	OPEN	CLOSE	5 V	OFF	L >
5	UNTIL	CLOSE	CLS	Ø >	PAINT(	Ø V
6	AND	EOF	COLOR	X >	PEEK(	P V
7	END	COPY	CONT	3 >	PLAY	J >
8	DEF	RENUM	COPY	7 V	PMODE	Z >
9	PROC	SCR	DATA	U >	POKE	P >
:	CLEAR	DIM	DEF	8 >	PRINT	A >
;	ON	USING	DIM	: V	PROC	9 >
A	PRINT	STRING$(	DIR	Z V	PUT	/ >
B	RESET	RENAME	DRAW	, >	READ	I >
C	SCREEN	LOAD	DRIVE	, V	RENAME	B V
D	INPUT	INKEY$	DSKI$	R V	RENUM	8 V
E	STEP	DSKO$	DSKO$	E V	REPEAT	4 >
F	ENVELOPE	CHR$(	EDIT	- >	RESET	B >
G	BEEP	LEFT$(	ELSE	Y >	RESTORE	O >
H	SOUND	RIGHT$(	END	7 >	RIGHT$(	H V
I	READ	JOYSTK(	ENVELOPE	F >	RND(	W V
J	PLAY	MID$(	EOF	D V	RUN	2 >
K	MOTOR	HEX$(	FOR	Q >	SAVE	X V
L	OFF	INSTR(	FREE(	3 V	SCR	9 V
M	CIRCLE(	VERIFY	GET	. >	SCREEN	C >
N	LINE	MERGE	GOSUB	V	SET	V >
O	RESTORE	INT(	GOTO	>	SOUND	H >
P	POKE	PEEK(	HEX$(	K V	STEP	E >
Q	FOR	TO	IF	R >	STR$(	S V
R	IF	DSKI$	INKEY$	D V	STRING$(	A V
S	WRITE	STR$(	INPUT	D >	THEN	T >
T	THEN	LEN(	INSTR(	L V	TO	Q V
U	DATA	ASC(	INT(	O V	UNLOAD	. V
V	SET	KILL	JOYSTK(	I V	UNTIL	5 >
W	NEXT	RND(	KEY	2 V	USING	; V
X	COLOR	SAVE	KILL	V V	VAL(	Y V
Y	ELSE	VAL(	LEFT$(	G V	VERIFY	M V
Z	PMODE	DIR	LEN(	T V	WRITE	S >

APPENDIX C - ERROR CODES

CODE	ERROR	DESCRIPTION
Ø	NF	"NEXT" prior to FOR
1	SN	Syntax
2	RG	"RETURN" before "GOSUB"
3	OD	Out of data
4	FC	Invalid parameter
5	OV	Overflow as # is too big
6	OM	Out of memory
7	UL	Undefined line #
8	BS	Invalid subscript in array
9	DD	Redimensioned array
10	/Ø	Invalid division by Ø
11	ID	Invalid command
12	TM	Type mismatch
13	OS	Out of string space
14	LS	Overlong string
15	ST	Overcomplex $ operation
16	CN	Cannot continue
17	FD	Data out of sequence
18	AO	File already opened
19	DN	Wrong device #
20	IO	Input/Output error
21	FM	File mismatch
22	NO	File not opened
23	IE	Reading past file end
24	DS	Missing Line #
25	UF	Undefined function
26	NE	Nonexistent file
27	BR	Bad record #
28	DF	Disk is full
29	OB	Out of buffer space
30	WP	Disk is write protected
31	FN	Unacceptable file name
32	FS	Bad file structure
33	AE	Already existing file
34	FO	Field length overflow
35	SE	String not fielded
36	VF	Verification error
37	ER	Past end of record

APPENDIX D - SAMPLE DEFINING ENVELOPE PROCEDURE

****

You are invited to include one or other of these envelopes to produce certain sound effects in your programs:

```
60000 DEFPROCdefine envelopes
60009 'Horn
60010 ENVELOPE1;1,50,1;1,206,1;0,0,0;0,0,0
60019 'Alarm
60020 ENVELOPE2;1,50,1;0,0,4;0,0,4;1,206,1
60029 'Hooter
60030 ENVELOPE3;0,200,1;0,206,4;0,0,0;0,0,0
60039 'Harpsichord
60040 ENVELOPE4;0,255,255;0,0,0;0,0,0;0,0,0
60049 'Violin
60050 ENVELOPE5;0,1,255;0,0,255;0,0,0;0,0,0
60059 'Siren
60060 ENVELOPE6;1,0,255;255,0,255;0,0,0;0,0,0
60069 'Trill
60070 ENVELOPE7;0,0,9;10,0,2;246,0,2;0,0,10
60079 'Foghorn
60080 ENVELOPE8;40,200,3;80,129,3;1,1,2;236,100,3
60090 ENDPROC
```


APPENDIX E - ALPHANUMERIC CODE

****

The alphanumeric code of graphic characters is the same as the ASCII code -- see page 276 of "Getting Started with Color BASIC" in order to work out the code of a graphic character.

The following list gives the codes for reverse video characters. Incidentally, one may derive the normal character codes by adding 64 to the numbers appearing below:

CHAR.	CODE
@	0
A	1
B	2
C	3
D	4
E	5
F	6
G	7
H	8
I	9
J	10
K	11
L	12
M	13
N	14
O	15
P	16
Q	17
R	18
S	19
T	20
U	21
V	22
W	23
X	24
Y	25
Z	26
[	27
\	28
]	29
^	30
<	31

CHAR.	CODE
	32
!	33
"	34
#	35
$	36
%	37
&	38
'	39
(	40
)	41
*	42
+	43
,	44
-	45
.	46
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
:	58
;	59
<	60
=	61
>	62
?	63

F - SYNTAX OF EXPANDED BASIC

Syntax words

Extra commands

Star commands

STAR COMMANDS ARE COMMANDS THAT ARE STORED ON DISK.